CHAPTER 1

INTRODUCTION

1.1 Android

Android is open source code mobile phone operating system that comes out by Google. Music player in this project is application software based on Google Android. Music is one of the best ways to relieve pressure in stressful modern society life. The purpose of this project is to develop a player which can play the mainstream file format. To browse and query the storage space as well as operation of playing can be realised. Meanwhile, this software can play, pause and select songs with latest button and next button.

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touch screen mobile devices such as smart phones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019.

Android Architecture: -

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram. GLUT gives you the ability to create a window, handle input and render to the screen without being Operating System dependent.

Android Architecture: -

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram. GLUT gives you the ability to create a window, handle input and render to the screen without being Operating System dependents

Libraries

On top of Linux kernel there is a set of libraries including open -source Web browser engine Web Kit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multithreading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

Android UI

An Android application user interface is everything that the user can see and interact with Installation steps of the developing environment

- Step 1: install the Java virtual machine JDK version -7
- Step 2: install the Android SDK: first download the Android SDK
- Download address: http://developer-android-com/sdk/index-html
- Input SDK tools path in the SDK location: D: \ android \ software \ android SDK– Windows and click OK
- The Android environment is set up successfully.

1.2 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

• A flexible Gradle-based build system
• A fast and feature-rich emulator
• A unified environment where you can develop for all Android devices
• Apply Changes to push code and resource changes to your running app without restarting your app
• Code templates and GitHub integration to help you build common app features and import sample code
• Extensive testing tools and frameworks
• Lint tools to catch performance, usability, version compatibility, and other problems
• C++ and NDK support
• Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

• Android app modules
• Library modules
• Google App Engine modules

By default, Android Studio displays your project files in the Android project view, This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

• manifests: Contains the AndroidManifest.xml file.
• java: Contains the Java source code files, including JUnit test code.
• res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select Project from the Project dropdown.

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the Problems view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file

1. The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

2. The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

3. The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4. The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5. The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6. The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window.

This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Tool windows

Instead of using pre-set perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

• To expand or collapse a tool window, click the tool's name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.

• To return to the current default tool window layout, click Window > Restore Default Layout or customize your default layout by clicking Window > Store Current Layout as Default.

• To show or hide the entire tool window bar, click the window icon in the bottom left hand corner of the Android Studio window.

• To locate a specific tool window, hover over the window icon and select the tool window from the menu.

You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows.

If you want to hide all toolbars, tool windows, and editor tabs, click View > Enter Distraction Free Mode. This enables Distraction Free Mode. To exit Distraction Free Mode, click View > Exit Distraction Free Mode.

You can use Speed Search to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

Code completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

CHAPTER 2

REQUIREMENT SPECIFICATION

The requirement specification is a comprehensive description of the software and the hardware requirements required to run the project successfully.

2.1 Hardware Requirements:

• Processor: intel/AMD processor.

• RAM: 8GB.

• Input: Keyboard/mouse

• Display: Monitor

• Memory: 4 GB

2.2 Software Requirements:

• Operating system: WINDOWS 10

• Language used: Xml and Java.

• Software: Android Studio.

REQUIREMENT ANALYSIS OF SYSTEM

The feasibility analysis:

This section verified that it is feasible to add paint application on the Android from the aspects of economic, technical and social feasibility.

Economic feasibility: To design Android mobile phone paint application as long as a computer has the Android development and the application development of Android is free.. The information that which functions are necessary form all the consumers , which functions are needed for some people, and which features are seldom to use is easy to understand. And a lot of research is eliminated, thus saved the spending. Therefore, the whole process of development doesn't need to spend any money that is economic feasibility.

Technical Feasibility Study

To design a music player which meets the basic requirements, a deep understand of JAVA language, the Android system architecture, application of framework and other technical knowledge are needed.(framework is the core of the application, and rules that all the programmers participating in the development must abide by).

CHAPTER 3

IMPLEMENTATION

Implementation is the stage where all planned activities are put into action. Before the implementation of a project, the implementors (spearheaded by the project committee or executive) should identify their strength and weaknesses (internal forces), opportunities and threats (external forces).

Create a simple PAINT APP in Android Studio

Casual painting is a part of everyone's life. Whenever a person is stressed out from his/her hectic life, the "Paint app" helps the individual for some stress free time. So, through this article, we will try to build an Paint application using Android Studio.

About the PAINT APP

This paint app will allow the users to draw any free hand structures, save it. The saved image will be in the Gallery app. The free hand structures includes different drawing with specific colours and brushes.

Features of the PAINT APP

1 It contains the multiple colour feature.
2 This app can save the art in Gallery.
3 We can share the art through Social Media
4 We can Undo and Redo while performing operations.
5 There is a clear all function, which can.

3.1 Flow of Application

1. **Workspace:** This is the part of application where the casual drawing or the actual work is done. The workspace is the most important part of this application.

2. **Sharing Screen:** This part of the application mainly deals with the sharing of the document. This can be shared through the messaging app in the operating system or through the social media platform installed in the operating system.

FLOWCHART

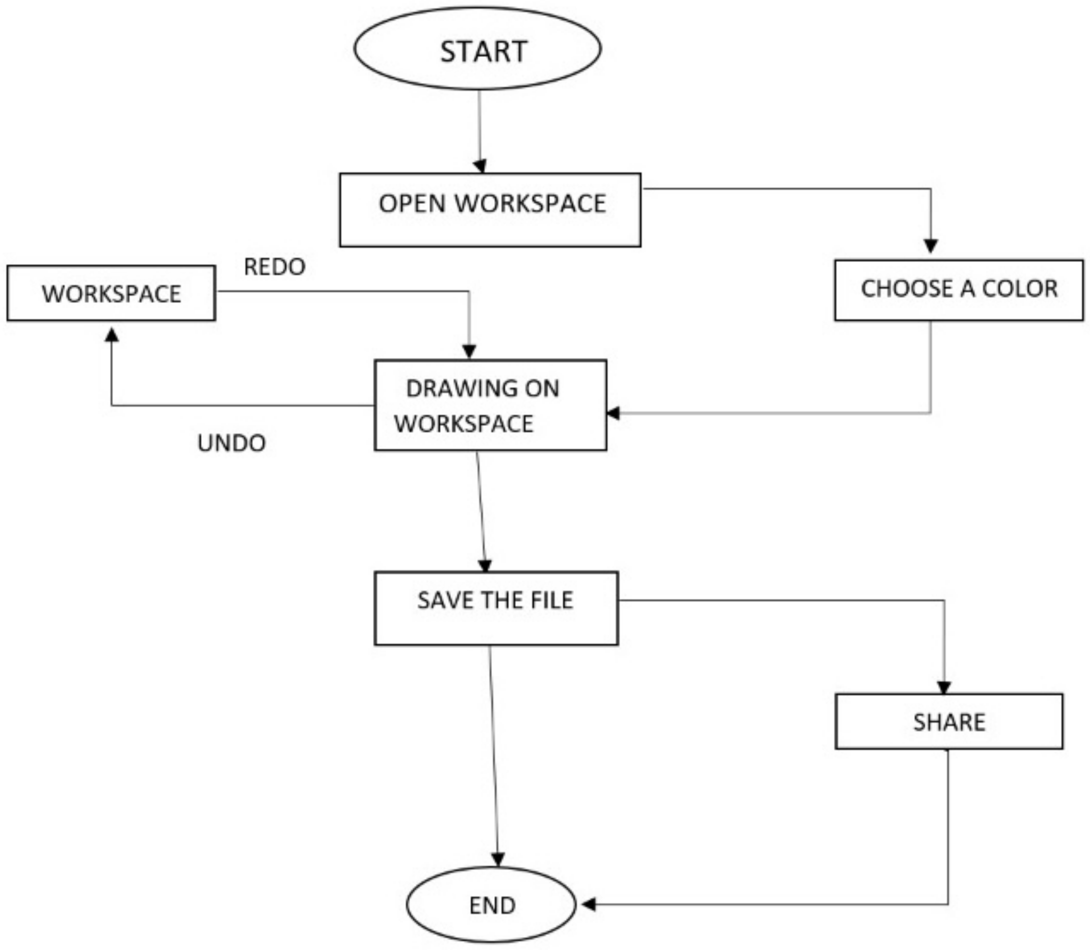

Fig. 3.1.1 - Flowchart

3.2 <u>Source Code</u>

<u>Java File</u>:

```
package com.example.simplepaintapp;
import android.content.Context;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Resources;
import android.graphics.drawable.GradientDrawable;
import android.media.MediaScannerConnection;
import android.net.Uri;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.ImageView;
import java.io.File;
public class MainActivity extends AppCompatActivity implements View.OnClickListener
{
   private CanvasExporter canvasExporter;
   private CanvasView canvasView;
   private ScaleGestureDetector scaleGestureDetector;
   @SuppressLint("ClickableViewAccessibility")
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      hideUINavigation();
      canvasExporter = new CanvasExporter();
      canvasView = findViewById(R.id.canvasView);
      ScaleHandler scaleHandler = createScaleHandler();
      scaleGestureDetector = new ScaleGestureDetector(MainActivity.this, scaleHandler);
      DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
      canvasView.initialise(displayMetrics.widthPixels, displayMetrics.heightPixels);
      canvasView.setOnTouchListener(new View.OnTouchListener()
      {
         @Override
         public boolean onTouch(View v, MotionEvent event)
```

```
        {
            scaleGestureDetector.onTouchEvent(event);
            switch (event.getAction())
            {
                case MotionEvent.ACTION_DOWN:
                    handleUIElements(View.INVISIBLE);
                    break;
                case MotionEvent.ACTION_UP:
                    handleUIElements(View.VISIBLE);
                    break;
            }
            if (event.getPointerCount() == 1 && !scaleGestureDetector.isInProgress())
            {
                if (canvasView.getPreviousStrokeWidth() == canvasView.getStrokeWidth())
                {
                    canvasView.handleTouches(event.getX(), event.getY(), event.getAction());
                } else
                {
                    canvasView.undo();
                }
                canvasView.setPreviousStrokeWidth(canvasView.getStrokeWidth());
            }
            return true;
}
    });
```

```
        ImageButton clearButton = findViewById(R.id.clearButton);

        clearButton.setOnClickListener(this);

        ImageButton undoButton = findViewById(R.id.undoButton);

        undoButton.setOnClickListener(this);

        ImageButton redoButton = findViewById(R.id.redoButton);

        redoButton.setOnClickListener(this);

        ImageButton styleButton = findViewById(R.id.styleButton);

        styleButton.setOnClickListener(this);

        ImageButton saveButton = findViewById(R.id.saveButton);

        saveButton.setOnClickListener(this);

        ImageButton shareButton = findViewById(R.id.shareButton);

        shareButton.setOnClickListener(this);

    }

    private void hideUINavigation()

    {

        final View view = getWindow().getDecorView();

        final int flags = View.SYSTEM_UI_FLAG_HIDE_NAVIGATION

                | View.SYSTEM_UI_FLAG_LAYOUT_STABLE

                | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION

                | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN

                | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY

                | View.SYSTEM_UI_FLAG_FULLSCREEN;

        view.setSystemUiVisibility(flags);

        view.setOnSystemUiVisibilityChangeListener(new
View.OnSystemUiVisibilityChangeListener()

        {
            @Override
            public void onSystemUiVisibilityChange(int visibility)
            {
                if ((visibility & View.SYSTEM_UI_FLAG_FULLSCREEN) == 0)
                    view.setSystemUiVisibility(flags);
            }
        });
    }
    private ScaleHandler createScaleHandler()
    {
        ImageView penIcon = findViewById(R.id.penSizeIcon);
        ScaleHandler scaleHandler = new ScaleHandler();
        scaleHandler.setOnScaleChangedListener(new ScaleHandler.ScaleChangedListener()
        {
           @Override
            public GradientDrawable onScaleIconRequired()
            {
                GradientDrawable gradientDrawable = (GradientDrawable) penIcon.getBackground();
                gradientDrawable.setColor(canvasView.getColour());
                return gradientDrawable;
            }
            @Override
            public Context getContext()
            {
                return MainActivity.this;
            }
            @Override
            public Resources getContextResources()
            {
                return getResources();
            }
            @Override
            public void onScaleStarted()
            {
                penIcon.setVisibility(View.VISIBLE);
            }
            @Override
            public void onScaleChanged(float scaleFactor)
            {
                if (scaleFactor == ScaleHandler.MIN_WIDTH || scaleFactor ==
    ScaleHandler.MAX_WIDTH)
                    canvasView.setPreviousStrokeWidth(Math.round(scaleFactor) - 1);
                canvasView.setStrokeWidth(Math.round(scaleFactor));
                ViewGroup.LayoutParams params = penIcon.getLayoutParams();
                params.width = (int) scaleFactor;
                params.height = (int) scaleFactor;
                penIcon.setLayoutParams(params);
            }
            @Override
            public void onScaleEnded()
            {
                penIcon.setVisibility(View.GONE);
            }
        });
        return scaleHandler;
    }
    private void handleUIElements (int showType)
    {
        ViewGroup viewGroup = findViewById(R.id.container);
        for (int i = 0; i < viewGroup.getChildCount(); i++)
        {
            View view = viewGroup.getChildAt(i);
            if (view.getId() != R.id.canvasView && view.getId() != R.id.penSizeIcon)
                view.setVisibility(showType);
        }
    }
    @Override
    public void onClick(View v) {
        int viewID = v.getId();

        if (viewID == R.id.clearButton)
        {
            canvasView.clear();
        } else if (viewID == R.id.undoButton)
        {
            canvasView.undo();
        } else if (viewID == R.id.redoButton)

        {

            canvasView.redo();

        } else if (viewID == R.id.styleButton)

        {

            ColourPickerDialog dialog = new ColourPickerDialog(MainActivity.this,
canvasView.getColour());

            dialog.setOnDialogOptionSelectedListener(new
ColourPickerDialog.ColourPickerOptionSelectedListener(

                @Override

                public void onColourPickerOptionSelected(int colour)

                {

                    canvasView.setColour(colour);

                }

            });

            dialog.show();

        } else if (viewID == R.id.saveButton)

        {

            canvasExporter.setExportType(CanvasExporter.FLAG_SAVE);

            checkForPermissions();

        } else if (viewID == R.id.shareButton)

        {

            canvasExporter.setExportType(CanvasExporter.FLAG_SHARE);

            checkForPermissions();

        }

    }
    private void requestStoragePermission ()
    {
        String permission = Manifest.permission.WRITE_EXTERNAL_STORAGE;
        ActivityCompat.requestPermissions(this, new String[]{permission},
                CanvasExporter.PERMISSION_WRITE_EXTERNAL_STORAGE);
    }
    private void checkForPermissions()
    {
        int permission = ContextCompat.checkSelfPermission(
                this, Manifest.permission.WRITE_EXTERNAL_STORAGE);

        if (permission == PackageManager.PERMISSION_DENIED)
        {
            boolean shouldShowRationale =
ActivityCompat.shouldShowRequestPermissionRationale(
                    this, Manifest.permission.WRITE_EXTERNAL_STORAGE);

            if (shouldShowRationale)
            {
                StorageRationaleDialog dialog = new StorageRationaleDialog(MainActivity.this);
                dialog.setOnStorageRationaleOptionSelectedListener(new
StorageRationaleDialog.StorageRationaleOptionSelectedListener()
                {
                    @Override
                    public void onStorageRationaleOptionSelected(boolean allow)
                    {
```

```
                    if (allow)
                        requestStoragePermission();
                }
            });
            dialog.show();
        } else
        {
            requestStoragePermission();
        }
    } else
    {
        exportImage();
}
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults)
    {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == CanvasExporter.PERMISSION_WRITE_EXTERNAL_STORAGE)
        {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
            {
                exportImage();
            } else
        {
            findViewById(R.id.saveButton).setEnabled(false);

            findViewById(R.id.saveButton).setAlpha(0.5f);

            findViewById(R.id.shareButton).setEnabled(false);

            findViewById(R.id.shareButton).setAlpha(0.5f);

        }

    }

}

private void exportImage ()

{

    if (canvasExporter.getExportType() == CanvasExporter.FLAG_SAVE)

    {

        String fileName = canvasExporter.saveImage(canvasView.getBitmap());


        if (fileName != null)

        {

            MediaScannerConnection.scanFile(

                    MainActivity.this, new String[]{fileName}, null, null);

            Toast.makeText(MainActivity.this, "The image was saved successfully.",
Toast.LENGTH_SHORT).show();

        } else

        {

            Toast.makeText(MainActivity.this, "There was an error saving the image.",
Toast.LENGTH_SHORT).show();

        }

    } else if (canvasExporter.getExportType() == CanvasExporter.FLAG_SHARE)
```

```
        {
            shareImage();
        }
    }
    private void shareImage()
    {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        File image = canvasExporter.getImage(canvasView.getBitmap());

        if (image != null)
        {
            Uri uri = FileProvider.getUriForFile(
                    MainActivity.this,
                    MainActivity.this.getApplicationContext().getPackageName() +
                            ".provider", canvasExporter.getImage(canvasView.getBitmap()));
            intent.putExtra(Intent.EXTRA_STREAM, uri).setType("image/png");
            startActivity(Intent.createChooser(intent, "Share image via"));
        } else
        {
            Toast.makeText(MainActivity.this, "There was an error sharing the image.",
Toast.LENGTH_SHORT).show();
        }
    }
}
```

CHAPTER 4

SNAPSHOTS

1. **WORKSPACE**

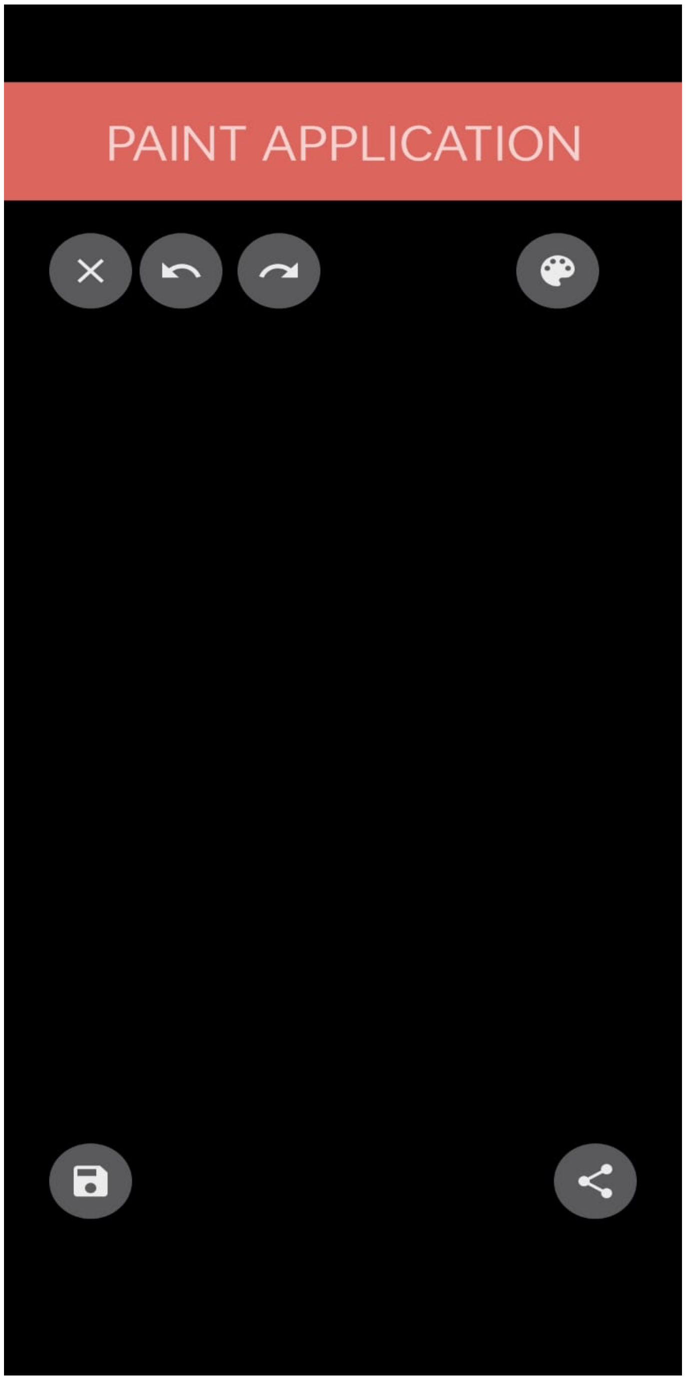

Fig 4.1 – Workspace where the free hand drawing is done

2. CHOOSING COLOR

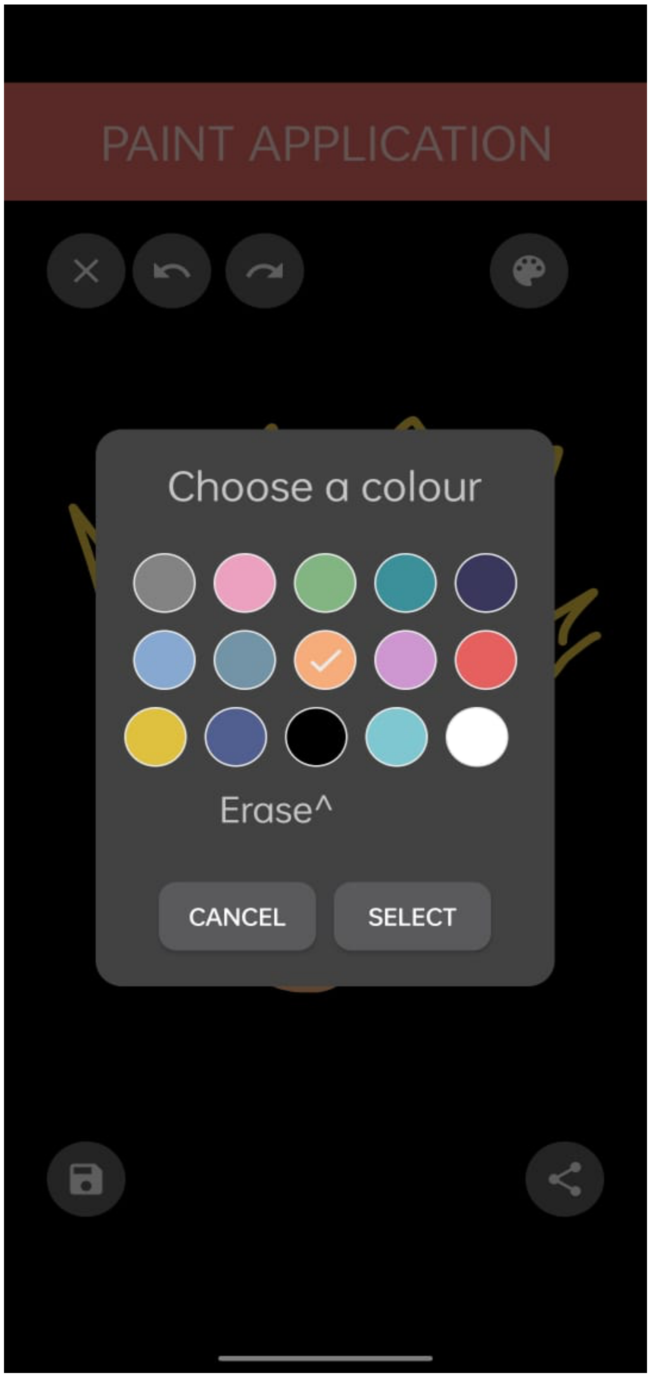

Fig 4.2 – Choosing a color from palette

3. DRAWING ON WORKSPACE

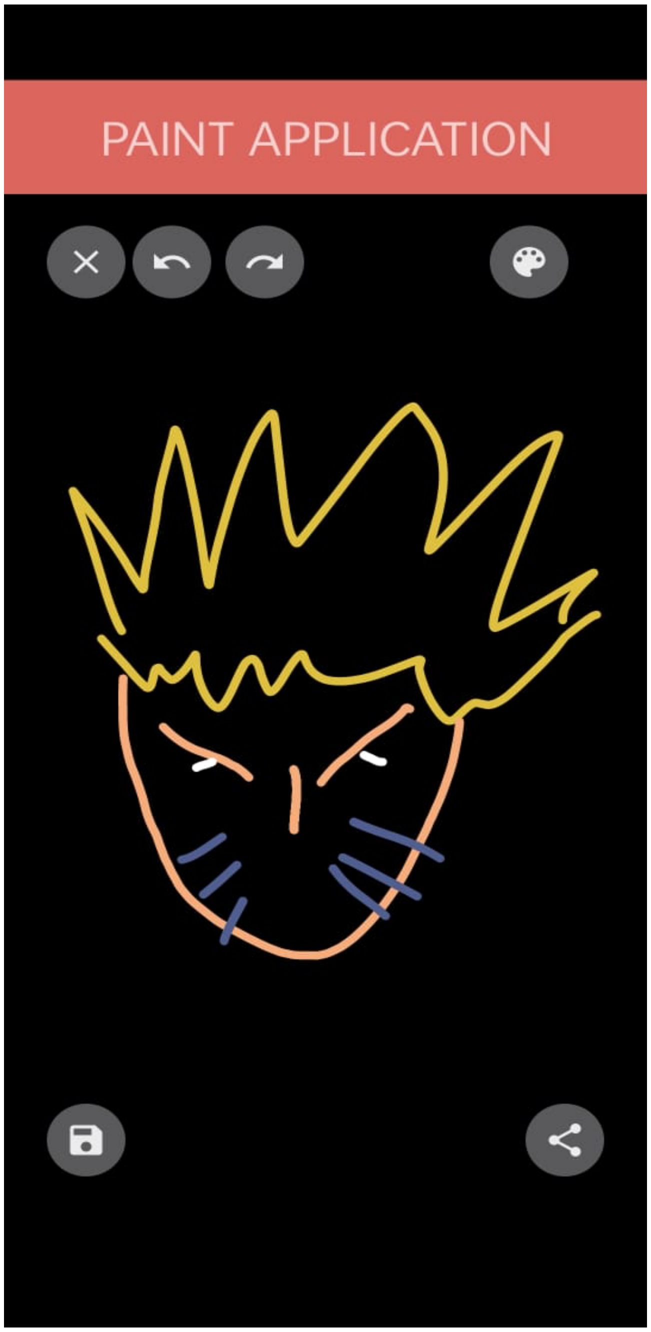

Fig 4.3 – Drawing on the workspace

4. SHARING PAGE

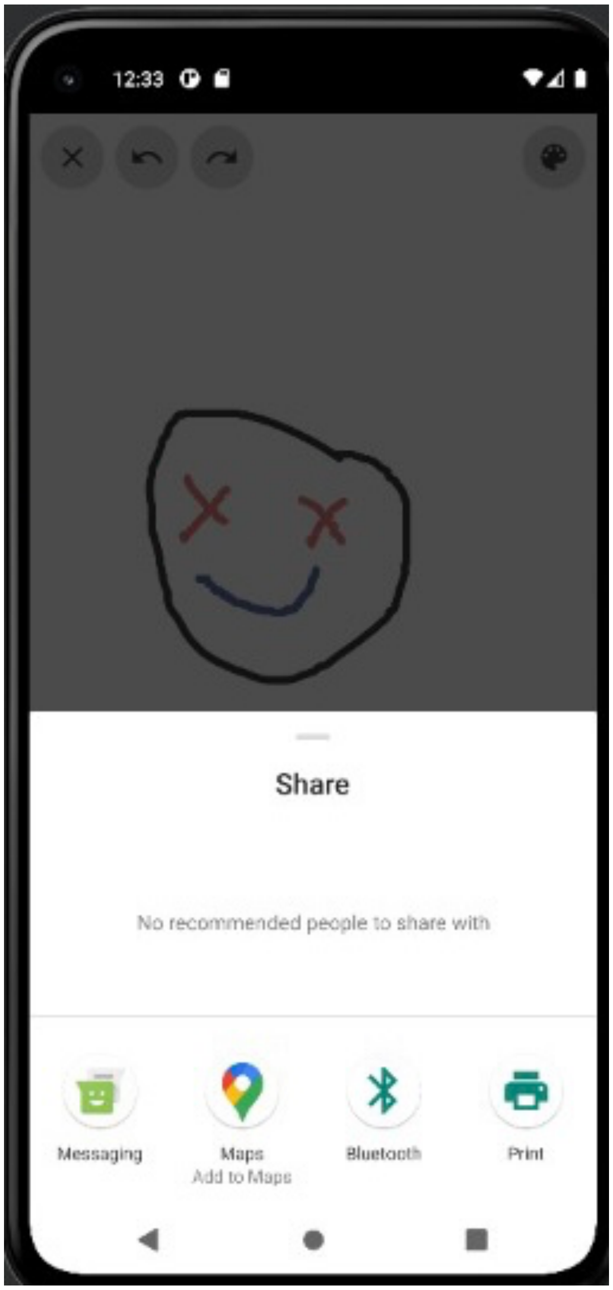

Fig 4.4 – Sharing and saving the drawing

CONCLUSION

Through the development of the PAINT APP in the Android Studio, we can conclude or understand the overall process of the system. With this application process, we mainly understand the process and the logic used in the Android Studio. We have many functionalities and implemented those in our application such as choosing colours, undo and redo functions and saving the file in the Gallery.

This PAINT APP has achieved the a lot of features and thus that is achieved with the combination of the java and xml files. The Android API is also responsible for saving the file in the system.

This design of the PAINT APP based on Android system requires elaborate design of the music player framework, by adopting ANDROID STUDIO 3.1.2 + Java language as technical support of this system, with the Android plug-in tools, and combination of Latest Android SDK version lead to the comprehensive and smoothly design and development of the mobile terminal.

REFERENCES:

[1] https://youtu.be/0zx_eFyHRU0

[2] [https://youtu.be/kMI2jy-WlGM]

[3] [https://youtu.be/fis26HvvDII]

[4] [https://youtu.be/A14BLDzNy2k]

[5] [https://github.com/topics/android-project]

[6] [https://www.amazon.com/kindle-dbs/hz/signup?ref_=assoc_tag_ph_1454291293420&_encoding=UTF8&camp=1789&creative=9325&linkCode=pf4&tag=uuid10-20&linkId=a1fe825323ace2514a09bd1765453d06]