Visvesvaraya Technological University

Belagavi, Karnataka – 590018

Project report on

DRAWING APPLICATION

Submitted in partial fulfillment of the requirements for the course

MOBILE APPLICATION DEVELOPMENT
(18CSMP68)

Submitted by

DEEPAK NAIDU	**1JS19CS049**
GIRISH KUMAR DV	**1JS19CS057**

Under the guidance of

Smt. Rashmi B N
Assistant professor, Dept. of CSE

Smt. Namitha SJ
Assistant professor, Dept. of CSE

JSS Academy of Technical Education, Bengaluru

JSS Academy of Technical Education,

Bengaluru – 560060,

Department of Computer Science and Engineering

2021-2022

JSS Mahavidyapeetha, Mysuru

JSS Academy of Technical Education

JSS Campus, Uttarahalli – Kengeri Main Road, Bengaluru – 560060

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work entitled "Alarm Application" is a bona fide work carried out by **Mr. DEEPAK NAIDU(1JS19CS049) and Mr. GIRISH KUMAR DV(1JS19CS057)** in partial fulfillment of the requirements for the course Mobile Application Development of 6th semester, Bachelor of engineering in Computer Science and engineering of the Visvesvaraya Technological University, Belagavi, during the academic year 2021 – 2022. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said degree.

Smt. Rashmi B N	**Smt. Namitha SJ**
Assistant professor, Dept. of CSE	Assistant professor, Dept. of CSE

Dr. Naveen N.C.

Professor and Head, Dept. of CSE

JSSATE, Bengaluru

Name of the examiners   Signature with date

1.___________________

2.___________________

ACKNOWLEDGEMENTS

I express my humble greetings to his holiness **Jagadguru Sri Shivarathri Deshikendra Mahaswamijigalavaru** who has showered their blessings on us for framing our career successfully.

The completion of any project involves the efforts of many people. I've been lucky to have received a lot of help and support from seniors and peers, during the making of this project. I take this opportunity to acknowledge all those, whose guidance andencouragement helped me emerge successful.

I am thankful to the resourceful guidance, timely assistance and graceful gesture of my guide **Smt. Rashmi BN and Smt. Namitha SJ**, Assistant professor, Department of Computer Science and engineering, who helped me in every aspect of my project work.

I am also indebted to **Dr. Naveen N.C.**, Professor and head of department of Computer Science and engineering for the facilities and support extended towards us.

I express our sincere thanks to our beloved principal, **Dr. Bhimasen Soragaon** for having supported us in our academic endeavors.

Last but not the least, I am pleased to express our heart full thanks to all the teaching and nonteaching staff of department of Computer and Science engineering and my friends who have rendered their help, motivation and support.

DEEPAK NAIDU (1JS19CS049)

GIRISH KUMAR DV (1JS19CS057)

ABSTRACT

Drawing Application is an android project that is implemented in Android Studio and it is developed to fulfil the Part-B laboratory requirements for Mobile Application Development (18CSMP68).

Compose Drawing is a fun application developed to allow user to draw/doodle/sketch on a mobile phone or a tablet. A canvas is provided as workspace for the user. A wide range of options are provided to the user in a toolbar where they can choose brush size, stroke width, stroke cap (Butt, Round, Square), Stroke Join (Miter, Round, Bevel), Eraser, Stroke Color which user can chose by adjusting RGBA values provided. Undo and Redo options are also provided to take care of any mistakes done while drawing.

The application is best suited for digital artists and kids as a learner platform. We aim to promote creativity and hence the product i.e. this application.

CONTENTS

LIST OF FIGURES

Chapter 1
INTRODUCTION

1.1 Overview

Ever wanted to note down something or scribble something as a reminder or want to let it out but couldn't find a paper and a pen?

Well, our drawing app lets you write down your thoughts, draw your imaginations and also let you doodle and scribble your thoughts away for any purpose.

This is simple drawing application which lets you draw with different strokes and widths and colors.

1.2 Motivation

In today's digital world it's difficult to keep up with traditional books and pens because they are not always available at our disposal. Keeping that in mind, incase a designer or a planner or an artist comes across an interesting idea in their minds, they can draw out their phones and immediately make a rough copy using our application. It helps in solving the immediate purpose which is the motive of building the application.

1.3 Technologies

This whole project has only one concept, to give a free hand to the user to draw out their thoughts and put it to a soft copy all in one application. This project is designed using Android Studio. The project uses Kotlin programming language for field validations. Gradle's and wrappers are used for assigning the SDKs and other runtimes platforms. This project also can be implemented in different gadgets like mobile phones. The whole project has a simple-looking UI design.

1.4 Mobile Application Development Front-End

The mobile front-end is the visual and interactive part of the application the user experiences. It usually resides on the device, or there is at least an icon representing the app that is visible on the home screen or is pinned in the application catalog of the device.

The application can be downloaded from the platform app store, side-loaded directly onto the device, or can be reached through the device's browser, as in the case for PWAs.

1.5 Mobile Application Development Back-End

Regardless of what front-end platform or development methodology is being used, delivering high-quality mobile applications that delight and retain users requires reliable back-end services. Given the critical importance of back-end services for the success of the mobile application, the developers have several important architectural decisions that they must consider. These decisions include which services should they build themselves and which third party services should they leverage, and then should they run and maintain their own services or should they take advantage of 3rd party services. The answer is increasingly clear; to improve developer productivity and efficiency, mobile app programmers should only build their own services if they are highly specific to the domain of the application and embody unique intellectual property. Also, even for the services they build themselves, they should almost always leverage cloud- based services to build and maintain their backend infrastructure.

1.6 Features Of Mobile Application Development

1. **Simplicity:** Of course, simplicity is the key to good performing applications. UI/UX design (User Interface/ User Experience) is a crucial factor that should be implemented in every application to draw the attention of audiences. Simple UI designs with simple or no login applications will catch the user's attention.
2. **Best performance:** The best performance is decided by the loading speed of the applications. Also, security plays a major role here. Most of the mobile apps are secured and have better user performance.
3. **Different modes of work:** Most of the mobile apps are offline and some of the apps are online. But both will have a particular traffic base to access them. Thus, users can change the app mode based on their network.
4. **Customization:** When comes to organize the applications, the users may have the choice to design their UI according to their wishes. The mobile app is having more customization features like a web app.
5. **Notifications:** Another main feature of mobile app development is, it is having Pop up notifications. Any new messages from the app or regarding the app will appear in front

and the user can easily remember the new features or messages.

6. **Branding:** A mobile app is a wide platform for marketing your business easily. Most of the apps are generating ads which result in branding for a particular business.
7. **Real-time uses:** A mobile app is popular because it is useful for real-time activities. Games, Chat applications makes the audience more engaging with it. To get a data intensive application, it should be fun, engaging at the same time, it should be useful because users are giving their valuable time to access your apps. Hence the app should worth the time of using it

Chapter 2

SYSTEM REQUIREMENTS

2.1 Hardware and Software Requirements

Software:

- Windows 7 Operating System.
- Android Studio Version 2021.2.1.15
- JSON
- XML
- JavaScript

Hardware:

- Dual Core Processor
- 2 GB RAM
- 2 GB Hard Disk Space
- Smart Phone

Chapter 3
SYSTEM DESIGN

3.1 Proposed System

In this paper, we create a drawing application in Android Studio. The system is designed so that user can draw or doodle or sketch out their thoughts on screen by touch. This application allows users to make use of various functionalities provided to get their desired results. The application aims to utmost use for designers, planner, digital artists and kids.

3.2 Features of System

The simple Drawing Android app we will be building in this tutorial will have the following features:

- User can draw on the canvas with different stroke widths of different brushes
- Different colors can be chosen
- User can undo or redo a given stroke
- User can move canvas without making a stroke
- User can erase strokes made on canvas
- User can select different stroke joins and stroke caps

3.3 Design

The first activity which will be used as the main screen for the Android app. It contains a plain white canvas which the user can use to draw any desired shapes or text. A toolbar is provided at the bottom of the screen with various useful tools for the user.

Chapter 4
IMPLEMENTATION

4.1 Main Activity Code

```
package com.smarttoolfactory.composedrawingapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Expand
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.smarttoolfactory.composedrawingapp.ui.theme.ComposeDrawingAppTheme
import com.smarttoolfactory.composedrawingapp.ui.theme.backgroundColor
import kotlinx.coroutines.launch

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ComposeDrawingAppTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    Scaffold(

                        topBar = {
                            TopAppBar(
                                elevation = 2.dp,
                                backgroundColor = MaterialTheme.colors.surface,
                                contentColor = MaterialTheme.colors.onSurface,
                                title = {
                                    Text("Drawing App")
                                },

                                actions = {}
                            )
                        }
                    ) { paddingValues: PaddingValues ->
                        DrawingApp(paddingValues)
                    }
                }
            }
        }
    }
    }
}

@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    ComposeDrawingAppTheme {
        Greeting("Android")
    }
}
```

4.2 Menu.kt file Code

```
package com.smarttoolfactory.composedrawingapp.ui.menu

import androidx.compose.foundation.Canvas
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Brush
import androidx.compose.material.icons.filled.Redo
import androidx.compose.material.icons.filled.TouchApp
import androidx.compose.material.icons.filled.Undo
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.shadow
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.Path
import androidx.compose.ui.graphics.StrokeCap
import androidx.compose.ui.graphics.StrokeJoin
import androidx.compose.ui.graphics.drawscope.Stroke
import androidx.compose.ui.platform.LocalDensity
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.window.Dialog
import com.smarttoolfactory.composedrawingapp.*
import com.smarttoolfactory.composedrawingapp.R
import com.smarttoolfactory.composedrawingapp.model.PathProperties
import com.smarttoolfactory.composedrawingapp.ui.ColorSlider
import com.smarttoolfactory.composedrawingapp.ui.ColorWheel
import com.smarttoolfactory.composedrawingapp.ui.theme.Blue400
import kotlin.math.roundToInt

@Composable
fun DrawingPropertiesMenu(
    modifier: Modifier = Modifier,
    pathProperties: PathProperties,
    drawMode: DrawMode,
    onUndo: () -> Unit,
    onRedo: () -> Unit,
    onPathPropertiesChange: (PathProperties) -> Unit,
    onDrawModeChanged: (DrawMode) -> Unit
) {

    val properties by rememberUpdatedState(newValue = pathProperties)
```

```
var showColorDialog by remember { mutableStateOf(false) }
   var showPropertiesDialog by remember { mutableStateOf(false) }
   var currentDrawMode = drawMode

   Row(
      modifier = modifier
//          .background(getRandomColor())
      ,
      verticalAlignment = Alignment.CenterVertically,
      horizontalArrangement = Arrangement.SpaceEvenly
   ) {
      IconButton(
         onClick = {
            currentDrawMode = if (currentDrawMode == DrawMode.Touch) {
               DrawMode.Draw
            } else {
               DrawMode.Touch
            }
            onDrawModeChanged(currentDrawMode)
         }
      ) {
         Icon(
            Icons.Filled.TouchApp,
            contentDescription = null,
            tint = if (currentDrawMode == DrawMode.Touch) Color.Black else Color.LightGray
         )
      }
      IconButton(
         onClick = {
            currentDrawMode = if (currentDrawMode == DrawMode.Erase) {
               DrawMode.Draw
            } else {
               DrawMode.Erase
            }
            onDrawModeChanged(currentDrawMode)
         }
      ) {
         Icon(
            painter = painterResource(id = R.drawable.ic_eraser_black_24dp),
            contentDescription = null,
            tint = if (currentDrawMode == DrawMode.Erase) Color.Black else Color.LightGray
         )
      }


      IconButton(onClick = { showColorDialog = !showColorDialog }) {
         ColorWheel(modifier = Modifier.size(24.dp))
      }
```

```
  IconButton(onClick = { showPropertiesDialog = !showPropertiesDialog }) {
            Icon(Icons.Filled.Brush, contentDescription = null, tint = Color.LightGray)
        }

        IconButton(onClick = {
            onUndo()
        }) {
            Icon(Icons.Filled.Undo, contentDescription = null, tint = Color.LightGray)
        }

        IconButton(onClick = {
            onRedo()
        }) {
            Icon(Icons.Filled.Redo, contentDescription = null, tint = Color.LightGray)
        }
    }

    if (showColorDialog) {
        ColorSelectionDialog(
            properties.color,
            onDismiss = { showColorDialog = !showColorDialog },
            onNegativeClick = { showColorDialog = !showColorDialog },
            onPositiveClick = { color: Color ->
                showColorDialog = !showColorDialog
                properties.color = color
            }
        )
    }

    if (showPropertiesDialog) {
        PropertiesMenuDialog(properties) {
            showPropertiesDialog = !showPropertiesDialog
        }
    }
}

@Composable
fun PropertiesMenuDialog(pathOption: PathProperties, onDismiss: () -> Unit) {

    var strokeWidth by remember { mutableStateOf(pathOption.strokeWidth) }
    var strokeCap by remember { mutableStateOf(pathOption.strokeCap) }
    var strokeJoin by remember { mutableStateOf(pathOption.strokeJoin) }

    Dialog(onDismissRequest = onDismiss) {

        Card(
            elevation = 2.dp,
            shape = RoundedCornerShape(8.dp),
            modifier = Modifier.padding(vertical = 8.dp)
        ) {
```

```
Column(modifier = Modifier.padding(8.dp)) {

        Text(
            text = "Properties",
            color = Blue400,
            fontSize = 18.sp,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.padding(start = 12.dp, top = 12.dp)
        )

        Canvas(
            modifier = Modifier
                .padding(horizontal = 24.dp, vertical = 20.dp)
                .height(40.dp)
                .fillMaxWidth()
        ) {
            val path = Path()
            path.moveTo(0f, size.height / 2)
            path.lineTo(size.width, size.height / 2)

            drawPath(
                color = pathOption.color,
                path = path,
                style = Stroke(
                    width = strokeWidth,
                    cap = strokeCap,
                    join = strokeJoin
                )
            )
        }

        Text(
            text = "Stroke Width ${strokeWidth.toInt()}",
            fontSize = 16.sp,
            modifier = Modifier.padding(horizontal = 12.dp)
        )

        Slider(
            value = strokeWidth,
            onValueChange = {
                strokeWidth = it
                pathOption.strokeWidth = strokeWidth
            },
            valueRange = 1f..100f,
            onValueChangeFinished = {}
        )


        ExposedSelectionMenu(title = "Stroke Cap",
            index = when (strokeCap) {
                  StrokeCap.Butt -> 0
                  StrokeCap.Round -> 1
                  else -> 2
                },
                options = listOf("Butt", "Round", "Square"),
                onSelected = {
                  println("STOKE CAP $it")
                  strokeCap = when (it) {
                    0 -> StrokeCap.Butt
                    1 -> StrokeCap.Round
                    else -> StrokeCap.Square
                  }

                  pathOption.strokeCap = strokeCap

                }
              )

              ExposedSelectionMenu(title = "Stroke Join",
                index = when (strokeJoin) {
                  StrokeJoin.Miter -> 0
                  StrokeJoin.Round -> 1
                  else -> 2
                },
                options = listOf("Miter", "Round", "Bevel"),
                onSelected = {
                  println("STOKE JOIN $it")

                  strokeJoin = when (it) {
                    0 -> StrokeJoin.Miter
                    1 -> StrokeJoin.Round
                    else -> StrokeJoin.Bevel
                  }

                  pathOption.strokeJoin = strokeJoin
                }
              )
            }
          }
        }
      }


@Composable
fun ColorSelectionDialog(
    initialColor: Color,
    onDismiss: () -> Unit,
    onNegativeClick: () -> Unit,
    onPositiveClick: (Color) -> Unit
```

```
){
   var red by remember { mutableStateOf(initialColor.red * 255) }
   var green by remember { mutableStateOf(initialColor.green * 255) }
   var blue by remember { mutableStateOf(initialColor.blue * 255) }
   var alpha by remember { mutableStateOf(initialColor.alpha * 255) }

   val color = Color(
      red = red.roundToInt(),
      green = green.roundToInt(),
      blue = blue.roundToInt(),
      alpha = alpha.roundToInt()
   )

   Dialog(onDismissRequest = onDismiss) {

      BoxWithConstraints(
         Modifier
            .shadow(1.dp, RoundedCornerShape(8.dp))
            .background(Color.White)
      ) {

         val widthInDp = LocalDensity.current.run { maxWidth }


         Column(horizontalAlignment = Alignment.CenterHorizontally) {

            Text(
               text = "Color",
               color = Blue400,
               fontSize = 18.sp,
               fontWeight = FontWeight.Bold,
               modifier = Modifier.padding(top = 12.dp)
            )

            // Initial and Current Colors
            Row(
               modifier = Modifier
                  .fillMaxWidth()
                  .padding(horizontal = 50.dp, vertical = 20.dp)
            ) {

               Box(
                  modifier = Modifier
                     .weight(1f)
                     .height(40.dp)
                     .background(
                        initialColor,
                        shape = RoundedCornerShape(topStart = 8.dp, bottomStart = 8.dp)
                     )
               )
```

```
    Box(
            modifier = Modifier
                .weight(1f)
                .height(40.dp)
                .background(
                    color,
                    shape = RoundedCornerShape(topEnd = 8.dp, bottomEnd = 8.dp)
                )
        )
    }

    ColorWheel(
        modifier = Modifier
            .width(widthInDp * .8f)
            .aspectRatio(1f)
    )

    Spacer(modifier = Modifier.height(16.dp))

    // Sliders
    ColorSlider(
        modifier = Modifier
            .padding(start = 12.dp, end = 12.dp)
            .fillMaxWidth(),
        title = "Red",
        titleColor = Color.Red,
        rgb = red,
        onColorChanged = {
            red = it
        }
    )
    Spacer(modifier = Modifier.height(4.dp))
    ColorSlider(
        modifier = Modifier
            .padding(start = 12.dp, end = 12.dp)
            .fillMaxWidth(),
        title = "Green",
        titleColor = Color.Green,
        rgb = green,
        onColorChanged = {
            green = it
        }
    )
    Spacer(modifier = Modifier.height(4.dp))

    ColorSlider(
        modifier = Modifier
            .padding(start = 12.dp, end = 12.dp)
            .fillMaxWidth(),
        title = "Blue",
  titleColor = Color.Blue,
            rgb = blue,
            onColorChanged = {
                blue = it
            }
        )

        Spacer(modifier = Modifier.height(4.dp))

        ColorSlider(
            modifier = Modifier
                .padding(start = 12.dp, end = 12.dp)
                .fillMaxWidth(),
            title = "Alpha",
            titleColor = Color.Black,
            rgb = alpha,
            onColorChanged = {
                alpha = it
            }
        )
        Spacer(modifier = Modifier.height(24.dp))

        // Buttons

        Row(
            modifier = Modifier
                .fillMaxWidth()
                .height(60.dp)
                .background(Color(0xffF3E5F5)),
            verticalAlignment = Alignment.CenterVertically

        ) {

            TextButton(
                onClick = onNegativeClick,
                modifier = Modifier
                    .weight(1f)
                    .fillMaxHeight()
            ) {
                Text(text = "CANCEL")
            }
            TextButton(
                modifier = Modifier
                    .weight(1f)
                    .fillMaxHeight(),
                onClick = {
                    onPositiveClick(color)
                },
            ) {
                Text(text = "OK")
                }
                    }
                }
            }
        }
}
@OptIn(ExperimentalMaterialApi::class)
@Composable
fun ExposedSelectionMenu(
    title: String,
    index: Int,
    options: List<String>,
    onSelected: (Int) -> Unit
) {

    var expanded by remember { mutableStateOf(false) }
    var selectedOptionText by remember { mutableStateOf(options[index]) }
    var selectedIndex = remember { index }

    ExposedDropdownMenuBox(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 4.dp),
        expanded = expanded,
        onExpandedChange = {
            expanded = !expanded
        }
    ) {
        TextField(
            modifier = Modifier.fillMaxWidth(),
            readOnly = true,
            value = selectedOptionText,
            onValueChange = { },
            label = { Text(title) },
            trailingIcon = {
                ExposedDropdownMenuDefaults.TrailingIcon(
                    expanded = expanded
                )
            },
            colors = ExposedDropdownMenuDefaults.textFieldColors(
                backgroundColor = Color.White,
                focusedIndicatorColor = Color.Transparent,
                unfocusedIndicatorColor = Color.Transparent,
                disabledIndicatorColor = Color.Transparent,
            )
        )
        ExposedDropdownMenu(
            modifier = Modifier.fillMaxWidth(),
            expanded = expanded,
            onDismissRequest = {
                expanded = false
    }
     ) {
        options.forEachIndexed { index: Int, selectionOption: String ->
            DropdownMenuItem(
                modifier = Modifier.fillMaxWidth(),
                onClick = {
                    selectedOptionText = selectionOption
                    expanded = false
                    selectedIndex = index
                    onSelected(selectedIndex)
                }
            ) {
                Text(text = selectionOption)
            }
        }
    }
  }
}
```

4.3 Android Manifest XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.smarttoolfactory.composedrawingapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ComposeDrawingApp">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.ComposeDrawingApp">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

4.4 Gradle Code

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-android-extensions'
}

android {
    compileSdk 31

    defaultConfig {
        applicationId "com.smarttoolfactory.composedrawingapp"
        minSdk 21
        targetSdk 31
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion compose_version
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'

    // Jetpack Compose
    implementation "androidx.compose.ui:ui:$compose_version"
    implementation "androidx.compose.ui:ui-tooling:$compose_version"
    implementation "androidx.compose.material:material:$compose_version"
    implementation "androidx.compose.material:material-icons-extended:$compose_version"
    implementation "androidx.compose.runtime:runtime:$compose_version"
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.4.1'
    implementation 'androidx.activity:activity-compose:1.4.0'

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_version"
}
```

Chapter 5

RESULTS

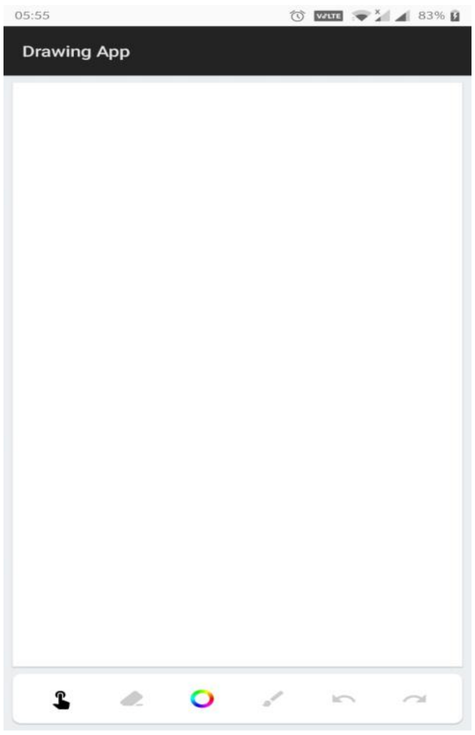

Fig 5.1: Blank Canvas on Startup

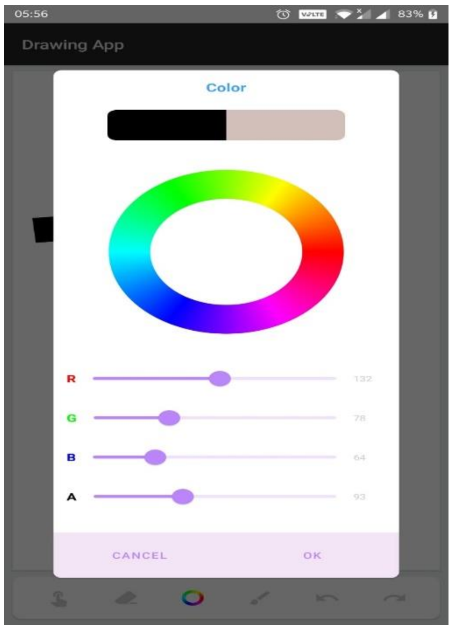

Fig 5.2: Color options to choose from for brush strokes

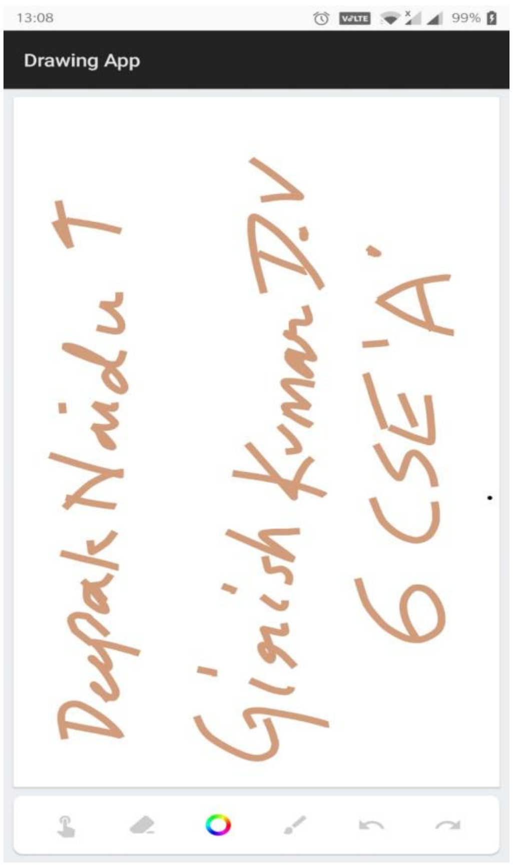

Fig 5.3: Drawing on the canvas

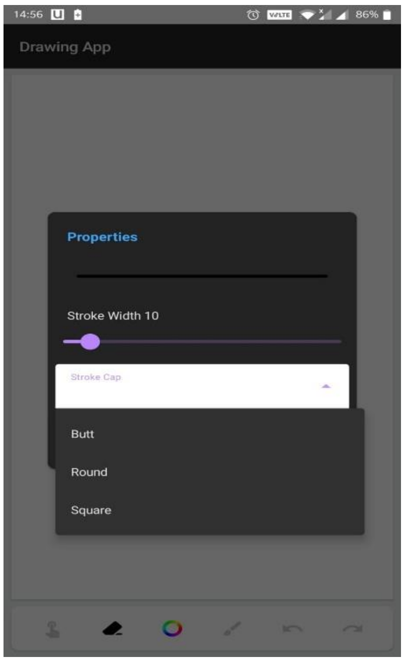

Fig 5.3: Brush options to choose from

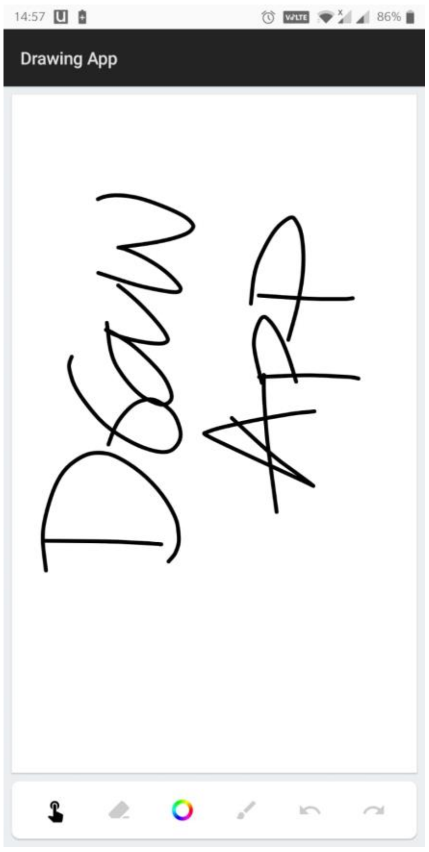

FIG 5.5: Output after changing brush stroke

Chapter 6

CONCLUSION

We develop and implement a drawing application in an event driven GUI software using the platform of the Android Studio. In this system, we create a drawing application in Android Studio. The system is designed so that user can draw any desired shapes and texts. And user can choose their choice of colors and brush strokes to draw. User can move canvas and also use the undo and redo functions provided to go back if any mistakes have been done.

Chapter 7

FUTURE ENHANCEMENT

The following section describes the future enhancements.

- There is more for the future work that can be done, can be implemented to other platform so the developers can aim to wider market, and can be done in various language so the application can suite better for each specific count.
- The application can be loaded with preset figures and shapes to make it easier for user to draw those shapes.
- Feature to store drawn image can be implemented

REFERENCES

[1] Introduction to LabVIEW– user manual - National Instruments http://www.ni.com/

[2] Udemy Android Development Course.

[3] Android studio documentation - https://developer.android.com/docs

[4] Kotlin Documentation - https://kotlinlang.org/

[5] Numerous queries from stack overflow.