This successfully implemented calculator app handles all four basic operations (+, -, ×, ÷) with a clean interface. It successfully implemented real-time calculations, error handling for division by zero, and a clear function. The responsive design successfully implemented smooth operation with instant results.

Ex No: 10
Date:

DYNAMIC TEXT STYLING INTERFACE

AIM:

To develop an Android application that allows users to change the font style and colour of displayed text, and shows a toast notification when a button is pressed.

ALGORITHM:

1. **Setup Project**:
    - Create a new Android Studio project with Empty Activity
    - Set minimum SDK version to support most devices
2. **Design UI Layout** (activity_main.xml):
    - Add a textView to display the text to be styled
    - Add buttons for different font styles (e.g., normal, bold, italic)
    - Add buttons for different text colours (e.g., red, blue, green)
    - Add a submit button that triggers the toast message
3. **Implement Functionality** (MainActivity.java):
    - Initialize UI components using findViewById()
    - Set on Click listeners for all buttons
    - For font style buttons:
        - Change TextView's typeface using setTypeface()
    - For colour buttons:
        - Change TextView's text colour using setTextColor()
    - For submit button:
        - Display toast message using Toast.makeText()
        - Show confirmation of text styling changes
4. **Test Application**:

- Run on emulator/real device
- Verify all buttons change text style/colour as expected
- Confirm toast appears when submit button is pressed

CODE:

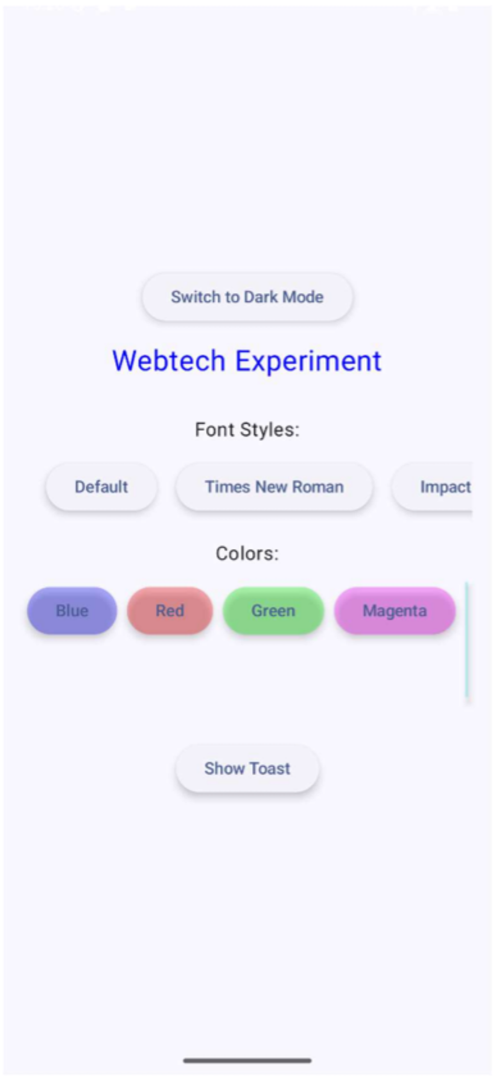
KOTLIN:

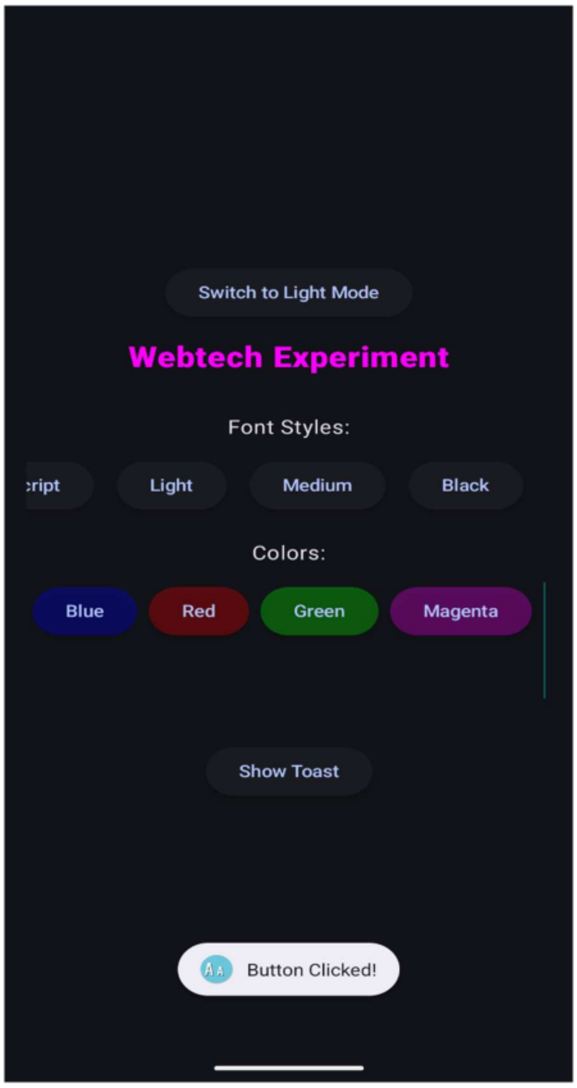
```
package com.example.webtech

import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.webtech.ui.theme.WebtechTheme


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            var isDarkMode by remember { mutableStateOf(false)
}

            WebtechTheme(darkTheme = isDarkMode) {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color =
MaterialTheme.colorScheme.background
                ) {
                    MainScreen(isDarkMode) { newMode ->
                        isDarkMode = newMode
                    }
                    }
                }
            }
        }
    }

    @Composable
    fun MainScreen(isDarkMode: Boolean, onThemeChange: (Boolean) -
    > Unit) {

        var textColor by remember { mutableStateOf(Color.Blue) }
        var currentFontFamily by remember {
    mutableStateOf(FontFamily.Default) }
        var currentFontWeight by remember {
    mutableStateOf(FontWeight.Normal) }
        val context = LocalContext.current

        val fontStyles = listOf(
            Triple(FontFamily.Default, FontWeight.Normal,
    "Default"),
            Triple(FontFamily.Serif, FontWeight.Normal, "Times New
    Roman")
        )


        val colors = listOf(
            Color.Blue to "Blue",
            Color.Red to "Red"
        )


        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {

            ElevatedButton(
                onClick = { onThemeChange(!isDarkMode) },
                modifier = Modifier.padding(bottom = 16.dp)
            ) {
                Text(if (isDarkMode) "Switch to Light Mode" else
    "Switch to Dark Mode")
            }


            Text(
                text = "Webtech Experiment",
                fontSize = 24.sp,
                color = textColor,
            fontFamily = currentFontFamily,
            fontWeight = currentFontWeight,
            modifier = Modifier.padding(bottom = 32.dp)
        )


        ElevatedButton(
            onClick = {
                Toast.makeText(context, "Button Clicked!",
Toast.LENGTH_SHORT).show()
            }
        ) {
            Text("Show Toast")
        }
    }
}
```

XML:

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

    <application
        android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Webtech"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.Webtech">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

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

OUTPUT:





RESULT:

The Android app successfully lets users change text fonts and colours instantly displaying modifications. A confirmation toast appears when applying changes, and the intuitive interface works smoothly across Android devices. Thus, the app is implemented successfully with all required functionalities working as intended.