AIM: TO BUILD AN APPLICATION USING KOTLIN TO WRITE INTO SD.

CODE:

activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <EditText  
 android:id="@+id/editText"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="16dp"  
 android:layout\_marginTop="200dp"  
 android:hint="Type something to save"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <Button  
 android:id="@+id/saveButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="168dp"  
 android:text="Save to File"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/editText" />  
  
 <TextView  
 android:id="@+id/statusTextView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text=""  
 app:layout\_constraintTop\_toBottomOf="@id/saveButton"  
 android:textColor="@android:color/black"  
 android:textSize="16sp"  
 tools:layout\_editor\_absoluteX="37dp"  
 tools:layout\_editor\_absoluteY="340dp" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

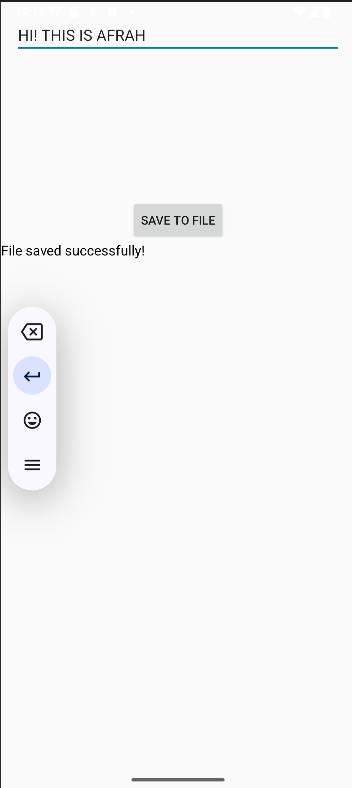
AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.sd">  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">  
  
 <activity android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
</manifest>

MainActivity.kt

package com.example.sd  
  
import android.Manifest  
import android.content.ContentValues  
import android.content.pm.PackageManager  
import android.os.Build  
import android.os.Bundle  
import android.os.Environment  
import android.provider.MediaStore  
import android.widget.Button  
import android.widget.EditText  
import android.widget.TextView  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.app.ActivityCompat  
import androidx.core.content.ContextCompat  
import java.io.OutputStream  
  
class MainActivity : AppCompatActivity() {  
  
 private val REQUEST\_CODE = 101  
 private lateinit var statusTextView: TextView // Declare a reference to the TextView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 val editText = findViewById<EditText>(R.id.*editText*)  
 val saveButton = findViewById<Button>(R.id.*saveButton*)  
 statusTextView = findViewById(R.id.*statusTextView*) // Initialize the TextView  
  
 saveButton.setOnClickListener **{** val text = editText.*text*.toString()  
  
 if (checkPermission()) {  
 saveToFile(text) // Save to shared storage using MediaStore  
 } else {  
 requestPermission()  
 }  
 **}** }  
  
 private fun saveToFile(data: String) {  
 // Prepare the content values for the file to be created  
 val contentValues = ContentValues().*apply* **{** put(MediaStore.MediaColumns.*DISPLAY\_NAME*, "myfile.txt") // File name  
 put(MediaStore.MediaColumns.*MIME\_TYPE*, "text/plain")  
 put(MediaStore.MediaColumns.*RELATIVE\_PATH*, Environment.*DIRECTORY\_DOCUMENTS*) // or Environment.DIRECTORY\_DOWNLOADS for Downloads folder  
 **}** // Get content resolver and insert into MediaStore  
 val resolver = *contentResolver* val uri = resolver.insert(MediaStore.Files.getContentUri("external"), contentValues)  
  
 uri?.*let* **{** try {  
 // Open an output stream to write the data  
 val outputStream: OutputStream? = resolver.openOutputStream(**it**)  
 outputStream?.write(data.*toByteArray*())  
 outputStream?.close()  
  
 // Show success in the TextView  
 statusTextView.*text* = "File saved successfully!" // Update TextView  
 Toast.makeText(this, "File saved successfully!", Toast.*LENGTH\_LONG*).show() // Optional Toast  
 } catch (e: Exception) {  
 // Handle any errors  
 statusTextView.*text* = "Error: ${e.message}" // Update TextView with error message  
 Toast.makeText(this, "Error: ${e.message}", Toast.*LENGTH\_LONG*).show()  
 }  
 **}** ?: *run* **{** // Handle failure if URI is null  
 statusTextView.*text* = "Failed to create file" // Update TextView with failure message  
 Toast.makeText(this, "Failed to create file", Toast.*LENGTH\_LONG*).show()  
 **}** }  
  
 private fun checkPermission(): Boolean {  
 return if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 // No need to request storage permission on Android 10+ for app-specific storage  
 true  
 } else {  
 // For Android versions below API 29 (Android 9 and below), check WRITE\_EXTERNAL\_STORAGE permission  
 val result = ContextCompat.checkSelfPermission(  
 this, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE* )  
 result == PackageManager.*PERMISSION\_GRANTED* }  
 }  
  
 private fun requestPermission() {  
 // Request storage permission if needed (for Android versions below 10)  
 ActivityCompat.requestPermissions(  
 this,  
 *arrayOf*(Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*),  
 REQUEST\_CODE  
 )  
 }  
  
 override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String>, grantResults: IntArray) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
 if (requestCode == REQUEST\_CODE) {  
 if (grantResults.*isNotEmpty*() && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 Toast.makeText(this, "Permission Granted", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Permission Denied", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
}

OUTPUT:



RESULT: APP BUILD SUCCESSFULLY.