EXP:11	IMPLEMENT AN APPLICATION THAT WRITES DATA TO THE SD CARD

Aim:

To develop an Android application that writes data to the **SD Card** (external storage) while handling runtime permissions and ensuring proper file operations.

Algorithm:

1. **Check & Request Permissions**
   - Verify if the app has WRITE_EXTERNAL_STORAGE permission (for older Android versions).
   - For Android 10 (API 29+) or later, use MANAGE_EXTERNAL_STORAGE if needed (scoped storage).
2. **Verify SD Card Availability**
   - Check if the external storage (SD Card) is **available** and **writable**.
3. **Create a File on SD Card**
   - Define a file path (e.g., /storage/emulated/0/MyApp/data.txt).
   - Use FileOutputStream to write data to the file.
4. **Write Data**
   - Open the file in write mode.
   - Write sample text (e.g., "Hello, SD Card!").
   - Close the file stream.
5. **Display Success/Failure**
   - Show a **Toast** message confirming successful write or error.

Code:

MainActivity.kt:

```
package com.example.sdcard;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.Settings;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_CODE = 100;
    private EditText inputText;
    private Button writeButton,clearButton;

    private static final int MANAGE_STORAGE_REQUEST_CODE = 101;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        inputText = findViewById(R.id.input_text);
        writeButton = findViewById(R.id.button_write);
        clearButton = findViewById(R.id.button_clear);

        // Check and request permissions based on Android version
        checkAndRequestPermissions();

        writeButton.setOnClickListener(v -> {
            String data = inputText.getText().toString();
            if (!data.isEmpty()) {
                if (isStoragePermissionGranted()) {
                    writeToFile(data);
                    inputText.setText(""); // Clear after writing
                } else {
                    Toast.makeText(MainActivity.this, "Storage
permission required", Toast.LENGTH_SHORT).show();
                    checkAndRequestPermissions();
                }
            } else {
                Toast.makeText(MainActivity.this, "Please enter some
text", Toast.LENGTH_SHORT).show();
            }
        });

        // Set up the Clear Button to clear the EditText
        clearButton.setOnClickListener(v -> inputText.setText(""));  //
Clears the EditText field
    }

    private void checkAndRequestPermissions() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            // Android 11 (API 30) and above
            if (!Environment.isExternalStorageManager()) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                builder.setTitle(R.string.permission_required_title);

builder.setMessage(R.string.permission_required_message);
                builder.setPositiveButton(R.string.dialog_button_ok,
(dialog, which) -> {
                    Intent intent = new
Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION);
                    Uri uri = Uri.fromParts("package", getPackageName(),
null);
                    intent.setData(uri);
                    startActivityForResult(intent,
MANAGE_STORAGE_REQUEST_CODE);
                });
                builder.setNegativeButton(android.R.string.cancel,
(dialog, which) ->
                    Toast.makeText(this,
R.string.permission_denied_message, Toast.LENGTH_LONG).show());
                builder.show();
            }
        } else {
            // Android 10 (API 29) and below
            if (ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(
                        this,
                        new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                        REQUEST_CODE
                );
            }
        }
    }

    private boolean isStoragePermissionGranted() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            return Environment.isExternalStorageManager();
        } else {
            return ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    == PackageManager.PERMISSION_GRANTED;
        }
    }

    private void writeToFile(String text) {
        if
(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)
) {
            File file;

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
                // For Android 11+, we'll use the app-specific directory
                file = new File(getExternalFilesDir(null),
"output.txt");
            } else {
                // For older versions, we can use the public directory
                File directory =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCU
MENTS);
                if (!directory.exists()) {
                    directory.mkdirs();
                }
                file = new File(directory, "output.txt");
            }

            try (FileOutputStream output = new FileOutputStream(file,
true)) {
                output.write((text + "\n").getBytes());
                // Show a more user-friendly success message with file
path
                showSuccessDialog(file.getAbsolutePath());
            } catch (IOException e) {
                Toast.makeText(this, "Failed to write: " +
e.getMessage(), Toast.LENGTH_LONG).show();
            }
        } else {
            Toast.makeText(this, "External storage not available",
Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == MANAGE_STORAGE_REQUEST_CODE) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
                if (Environment.isExternalStorageManager()) {
                    Toast.makeText(this,
R.string.permission_granted_message, Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(this,
R.string.permission_denied_message, Toast.LENGTH_SHORT).show();
                }
            }
        }
    }

    /**
     * Shows a dialog with the file path where data was saved
     * @param filePath The absolute path to the saved file
     */
    private void showSuccessDialog(String filePath) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.dialog_success_title);

        // Create a simplified path for display
        String displayPath = filePath;
        try {
            // Try to make the path more readable
            if (filePath.contains("/Android/data/")) {
                displayPath = "App Storage: " +
filePath.substring(filePath.lastIndexOf("/") + 1);
            } else if (filePath.contains("/Documents/")) {
                displayPath = "Documents: " +
filePath.substring(filePath.lastIndexOf("/") + 1);
            }
        } catch (Exception e) {
            // If any error in string manipulation, use the full path
            displayPath = filePath;
        }

        builder.setMessage(getString(R.string.dialog_success_message) +
"\n" + displayPath);
        builder.setPositiveButton(R.string.dialog_button_ok, null);

        // Add a button to show the full path
        builder.setNeutralButton(R.string.dialog_button_show_path,
(dialog, which) -> {
```

```
            AlertDialog.Builder pathBuilder = new
AlertDialog.Builder(this);
            pathBuilder.setTitle(R.string.dialog_full_path_title);
            pathBuilder.setMessage(filePath);
            pathBuilder.setPositiveButton(R.string.dialog_button_ok,
null);
            pathBuilder.show();
        });

        builder.show();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == REQUEST_CODE && grantResults.length > 0) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "Permission granted",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Permission denied",
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

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"
    android:background="@color/accent"
    tools:context=".MainActivity">

    <androidx.cardview.widget.CardView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        app:cardCornerRadius="16dp"
        app:cardElevation="8dp"
        app:cardUseCompatPadding="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.4">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
```

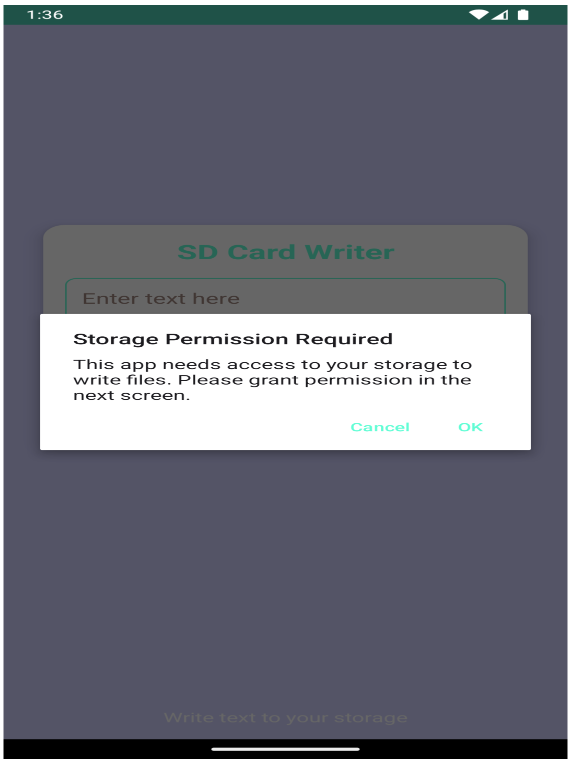
```xml
            android:padding="16dp">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="@string/app_name"
                android:textColor="@color/primary"
                android:textSize="24sp"
                android:textStyle="bold"
                android:layout_marginBottom="16dp" />

            <EditText
                android:id="@+id/input_text"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="@drawable/edit_text_background"
                android:hint="@string/enter_text_here"
                android:inputType="textMultiLine"
                android:minHeight="100dp"
                android:padding="12dp"
                android:textColorHint="@color/black"
                android:textColor="@color/black"
                android:textSize="18sp"
                android:gravity="top|start"
                android:layout_marginBottom="16dp"
                tools:ignore="Autofill" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:gravity="center">

                <com.google.android.material.button.MaterialButton
                    android:id="@+id/button_write"
                    style="@style/Widget.MaterialComponents.Button"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_margin="8dp"
                    android:layout_weight="1"
                    android:backgroundTint="@color/primary"
                    android:padding="12dp"
                    android:text="@string/write"
                    android:textColor="@color/white"
                    app:cornerRadius="24dp"
                    app:icon="@drawable/download_24"
                    app:iconGravity="textStart"
                    app:iconTint="@color/white" />

                <com.google.android.material.button.MaterialButton
                    android:id="@+id/button_clear"

      style="@style/Widget.MaterialComponents.Button.OutlinedButton"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_margin="8dp"
                    android:layout_weight="1"
                    android:padding="12dp"
                    android:text="@string/clear"
                    android:textColor="@color/primary"
                    app:cornerRadius="24dp"
                    app:icon="@drawable/delete_24"
                    app:iconGravity="textStart"
                    app:iconTint="@color/primary"
                    app:strokeColor="@color/primary"
                    app:strokeWidth="2dp" />
            </LinearLayout>
        </LinearLayout>
    </androidx.cardview.widget.CardView>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/footer_text"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

AndroidManifest.xml:

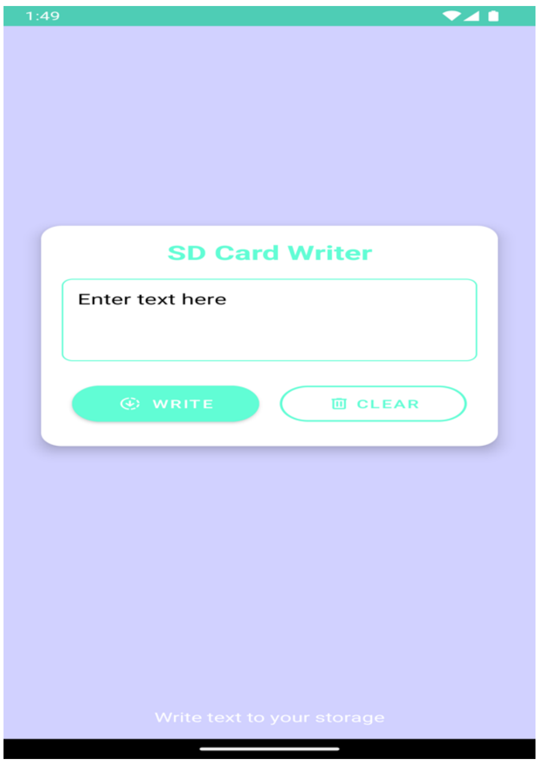
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <!-- Storage permissions -->
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"
                     android:maxSdkVersion="29" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"
                     android:maxSdkVersion="32" />
    <!-- For Android 11+ -->
    <uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
                     tools:ignore="ScopedStorage" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:hardwareAccelerated="true"
        tools:ignore="HardwareIds"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.sdcard"
        tools:targetApi="31">
        <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>
```

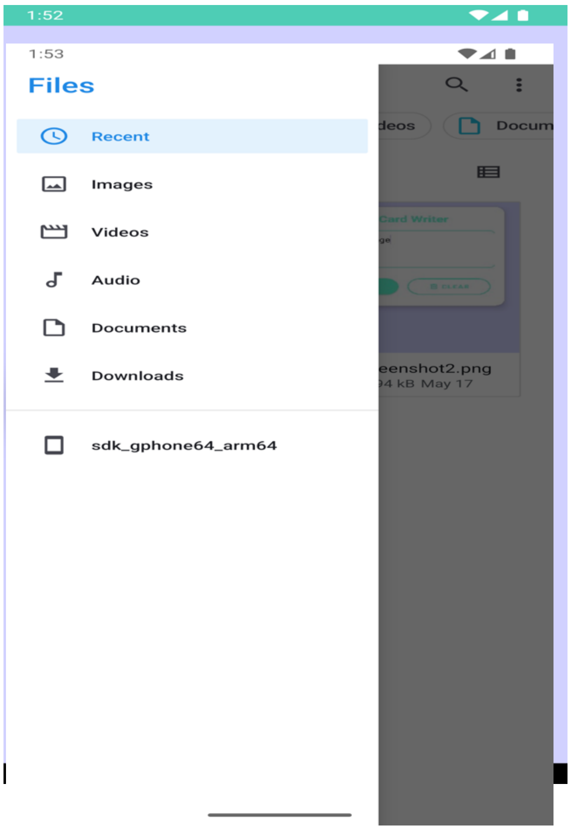
Output:

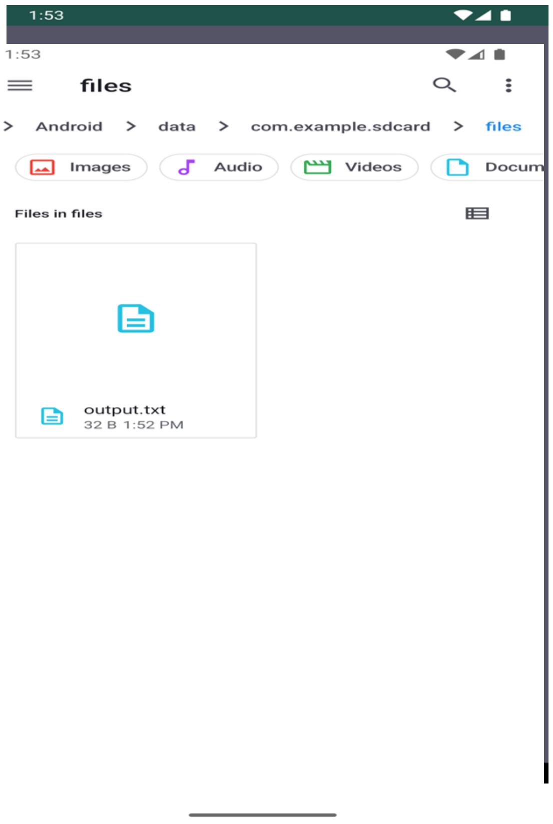


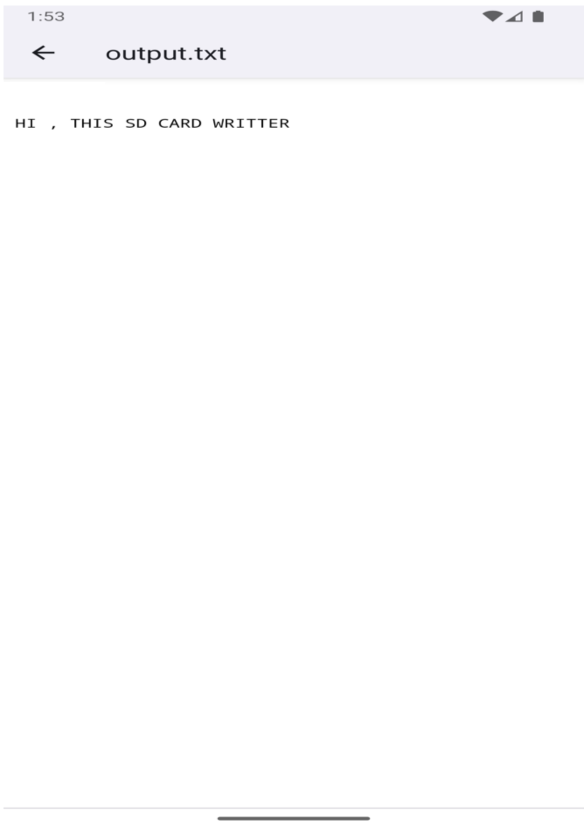
1:36
SD Card Writer
Enter text here
Storage Permission Required
This app needs access to your storage to write files. Please grant permission in the next screen.
Cancel
OK
Write text to your storage



1:49
SD Card Writer
Enter text here
WRITE
CLEAR
Write text to your storage







```
HI , THIS SD CARD WRITTER
```

Result:

Thus the give program is executed successfully.