Exercise 6
Mobile Device Development
File IO Operations
Due: 10/27/2020@11:59pm

Reading and Writing files in Android will be helpful if you want to read data from sdcard, store your application data as a text file. In this exercise, we are going to Write data in a text file in the internal SD card and Read data from a text file from internal SD card.

Creating Project

Create a new Android Application project with package as **edu.sjsu.android.externalstorage**. Create the main layout as activity_main and main Activity as MainActivity.

Creating Layout

Our main layout consists of three EditText widgets to get file name, file content and name of the file which should be read as input. It also has a TextView widget to display the file contents which is read from the file from internal SD card. It has two button widgets to perform read and write operation.

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_gravity="center"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textAlignment="center"
        android:text="Android Read/Write File" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/fname"
        android:hint="File Name" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="100px"
        android:id="@+id/ftext"
        android:hint="File Text" />
    <Button
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/btnwrite"
        android:text="Write File" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/fnameread"
        android:hint="File Name" />
       <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnread"
        android:text="Read File" />
       <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/filecon" />

</LinearLayout>
```

Creating Activity

Before proceeding to MainActivity, you need to create **FileOperations** class which has functions to perform Read and Write operations. For Write operation if the file does not exists, a new file will be created and contents are written. The text contents are written using BufferedWriter. For Read operation BufferedReader is used to read the contents of the file.

FileOperations.java

```
package edu.sjsu.android.externalstorage;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import android.util.Log;
import android.os.Environment;

public class FileOperations {
  private String path;

  public FileOperations(String path) {
    this.path = path;
  }
  public Boolean write(String fname, String fcontent){
    try {
      File file = new File(path, fname);

      // If file does not exists, then create it
      if (!file.exists()) {
        file.createNewFile();
      }
```

```
        FileWriter fw = new FileWriter(file.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(fcontent);
        bw.close();

        Log.d("Suceess","Sucess");
            return true;
        } catch (IOException e) {
          e.printStackTrace();
          return false;
        }
      }
    public String read(String fname){

      BufferedReader br = null;
      String response = null;

      try {
        StringBuffer output = new StringBuffer();
        File file = new File(path, fname);

        br = new BufferedReader(new FileReader(file));
        String line = "";
        while ((line = br.readLine()) != null) {
          output.append(line +"\n");
        }
        response = output.toString();
      } catch (IOException e) {
        e.printStackTrace();
        return null;
      }
      return response;
    }
  }
```

When the write button is pressed the write function is invoked from FileOperations class and the file is written in your internal SD card. When Read button is pressed the read function is invoked and the text is displayed in Textview.

Main Activity

```
package edu.sjsu.android.externalstorage;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
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;
public class MainActivity extends AppCompatActivity {
```

```
  private static final int PERMISSION_REQUEST_CODE = 100;
  EditText fname,fcontent,fnameread;
  Button write,read;
  TextView filecon;
  String path;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    fname = (EditText)findViewById(R.id.fname);
    fcontent = (EditText)findViewById(R.id.ftext);
    fnameread = (EditText)findViewById(R.id.fnameread);
    write = (Button)findViewById(R.id.btnwrite);
    read = (Button)findViewById(R.id.btnread);
    filecon = (TextView)findViewById(R.id.filecon);
    path = getExternalFilesDir(null).getPath();

    write.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View arg0) {
        String filename = fname.getText().toString();
        String filecontent = fcontent.getText().toString();
        if (checkPermission()) {
          FileOperations fop = new FileOperations(path);
          fop.write(filename, filecontent);
          if(fop.write(filename, filecontent)){
            Toast.makeText(getApplicationContext(), filename+"created",
                                                          Toast.LENGTH_SHORT).show();
          }else{
            Toast.makeText(getApplicationContext(), "I/O error",
                                                          Toast.LENGTH_SHORT).show();
          }
        } else {
          requestPermission(); // Code for permission
        }
      }
    });
    read.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View arg0) {
        String readfilename = fnameread.getText().toString();
        FileOperations fop = new FileOperations(path);
        String text = fop.read(readfilename);
        if(text != null){
          filecon.setText(text);
        }
        else {
          Toast.makeText(getApplicationContext(), "File not Found",
                                                          Toast.LENGTH_SHORT).show();
          filecon.setText(null);
        }
      }
    });
  }

  private boolean checkPermission() {
```

```
    int result = ContextCompat.checkSelfPermission(
                    MainActivity.this, android.Manifest.permission.WRITE_EXTERNAL_STORAGE);
    if (result == PackageManager.PERMISSION_GRANTED) {
      return true;
    } else {
      return false;
    }
  }

  private void requestPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(
                MainActivity.this, android.Manifest.permission.WRITE_EXTERNAL_STORAGE)) {
      Toast.makeText(MainActivity.this, "Write External Storage permission",
                                                              Toast.LENGTH_LONG).show();
    } else {
      ActivityCompat.requestPermissions(
      MainActivity.this, new String[]{android.Manifest.permission.WRITE_EXTERNAL_STORAGE},
                                                               PERMISSION_REQUEST_CODE);
    }
  }
  @Override
  public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
    switch (requestCode) {
      case PERMISSION_REQUEST_CODE:
        if (grantResults.length>0 && grantResults[0]==PackageManager.PERMISSION_GRANTED) {
          Log.e("value", "Permission Granted, Now you can use local drive .");
        } else {
          Log.e("value", "Permission Denied, You cannot use local drive .");
        }
        break;
    }
  }
}
```

Creating Manifest

Add the following Permissions to the AndroidManifest.xml

```
<uses-permission android:name=”android.permission.WRITE_EXTERNAL_STORAGE” />
<uses-permission android:name=”android.permission.READ_EXTERNAL_STORAGE” />
```

Runtime Permissions

Android defines basically three types of permissions:

- Normal Permissions
- Signature Permissions
- Dangerous Permissions

Both Normal and Dangerous permissions must be defined in the Manifest file. But only Dangerous permissions are checked at runtime, Normal permissions are not.

Normal Permissions

Some permissions are automatically granted to the application. Just we need to declare those permissions in AndroidManifest.xml and it will work fine. Those permissions are called Normal Permissions. An example of a normal permission is INTERNET.

Normal Permissions are automatically granted to the application.

Signature Permissions

A permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.

Dangerous Permissions

Some permissions may affect users private info, or could potentially affect their data or the operation of other application are called Dangerous Permissions. For example, the ability to read the user's contacts is a dangerous permission. Some other examples are CONTACTS, CAMERA, CALENDAR, LOCATION , PHONE , STORAGE , SENSORS , MICROPHONE, etc.

Dangerous permissions must be granted by the user at runtime to the app.

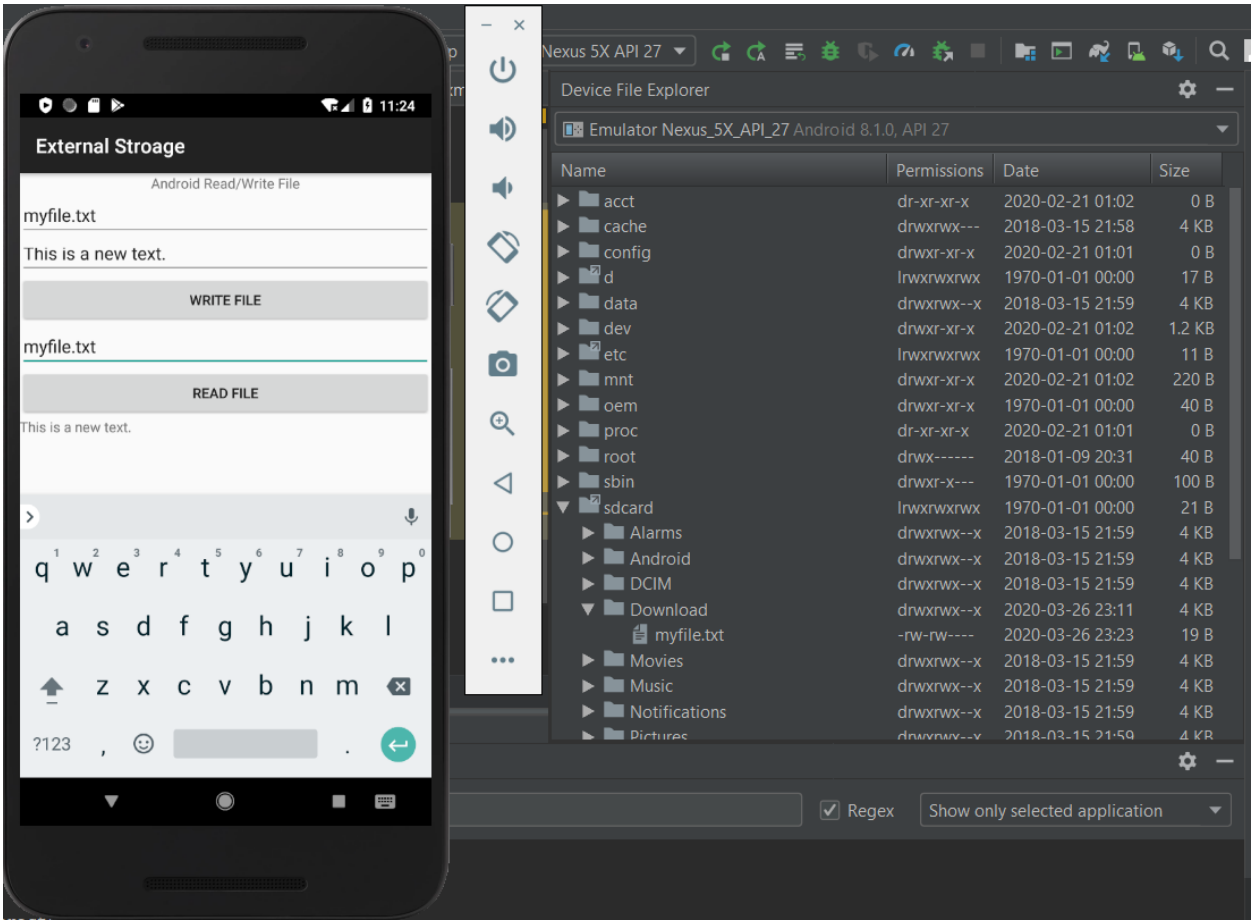
Dangerous permissions are grouped into categories that make it easier for the user to understand what they are allowing the application to do. If any permission in a Permission Group is granted. Another permission in the same group will be automatically granted as well. For example , once WRITE_CONTACTS is granted, application will also grant READ_CONTACTS and GET_ACCOUNTS as all three permissions in the same group.

In this exercise, we use the Android Support Library to check for and request permissions. Using the Android support library makes it easier to provide compatibility with older versions of Android.

Device File Explorer

The Device File Explorer allows you to view, copy, and delete files on an Android device. This is useful when examining files that are created by your app or if you want to transfer files to and from a device.

1. Open file system explorer **:View > Tool Windows > Device File Explorer**
2. In **Device File Explorer** : Click right button , you will see **Upload** and **Save as**



Submission

1. Push your project directory along with the source to remote bitbucket repository by the due date.
2. Invite and share your project repository the Grader (yan.chen01@sjsu.edu) and Instructor (ramin.moazeni@sjsu.edu).
3. Submit a Readme.pdf to Canvas including your name, repository access link, instructions to run your program (if any), snapshot of your running program
4. Your project directory will be graded according to the state your project directory was in at due time when fetched.