DOP: / /2023 **DOS: / /2023**

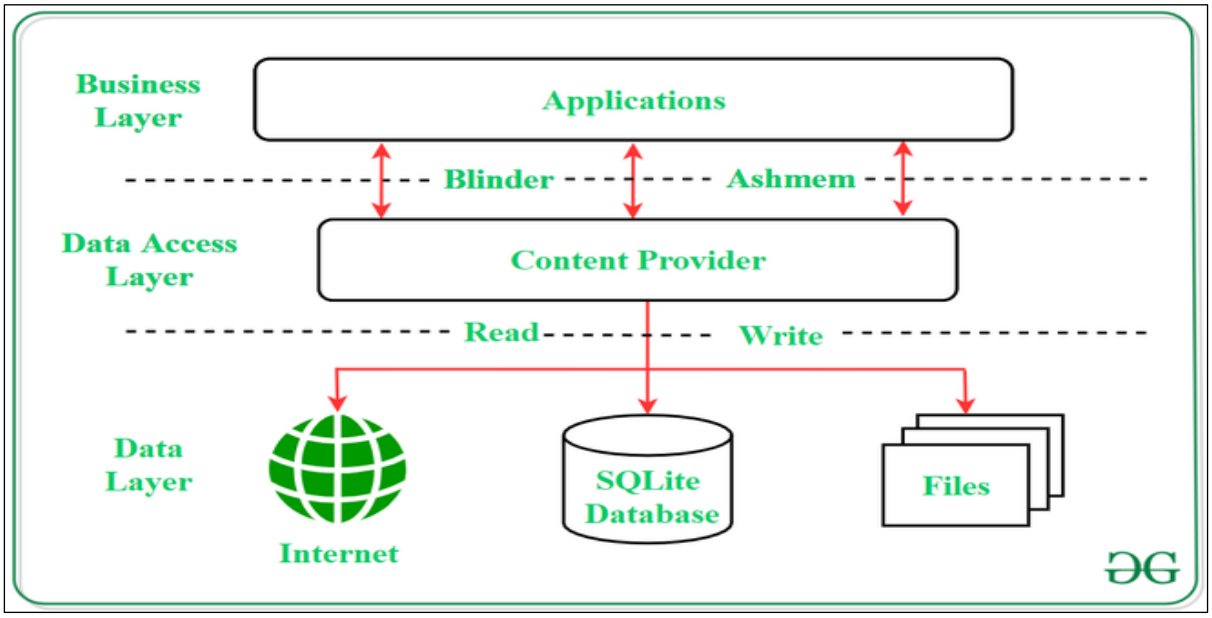
Experiment No:

Title: Developing Android applications using Receivers and Content Providers.

Theory:

Content Providers:

Content Providers are a very important component that serves the purpose of a relational database to store the data of applications. The role of the content provider in the android system is like a central repository in which data of the applications are stored, and it facilitates other applications to securely access and modifies that data based on the user requirements. Android system allows the content provider to store the application data in several ways. Users can manage to store the application data like images, audio, videos, and personal contact information by storing them in SQLite Database, in files, or even on a network. In order to share the data, content providers have certain permissions that are used to grant or restrict the rights to other applications to interfere with the data.



Content URI:

Content URI(Uniform Resource Identifier) is the key concept of Content providers. To access the data from a content provider, URI is used as a query string.

Structure of a Content URI: content://authority/optionalPath/optionalID

Operations in Content Provider:

Four fundamental operations are possible in Content Provider namely Create, Read, Update, and Delete. These operations are often termed as CRUD operations.

- Create: Operation to create data in a content provider.
- Read: Used to fetch data from a content provider.
- Update: To modify existing data.
- Delete: To remove existing data from the storage.

Input:

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#2EAFA3"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="                Content provider"
        android:textSize="25sp"
        android:textStyle="bold"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="126dp"
        tools:layout_editor_absoluteY="99dp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="133dp"
        android:layout_height="34dp"
        android:layout_marginEnd="32dp"
        android:hint="Name"
        android:text=""
        android:textAlignment="center"
        android:textSize="25sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.516"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.286" />
```

```
    <TextView
        android:id="@+id/textView2"
        android:layout_width="140dp"
        android:layout_height="34dp"
        android:layout_marginEnd="144dp"
        android:hint="Contact no"
        android:maxLength="13"
        android:text=""
        android:textAlignment="center"
        android:textSize="25sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.433" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.592" />

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

Javafile:

```
package com.example.content_provider;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.ContentResolver;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.util.Range;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    Button btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = findViewById(R.id.button);
        btn.setOnClickListener(this);
    }
    @Override
    public void onClick(View view) {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.READ_CONTACTS}, 100);
        } else {
            Intent openContactApp = new Intent(Intent.ACTION_PICK, ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
            startActivityForResult(openContactApp, 123);
        }
    }
    @Override
```

```
    @Override
    public void onClick(View view) {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.READ_CONTACTS}, 100);
        } else {
            Intent openContactApp = new Intent(Intent.ACTION_PICK, ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
            startActivityForResult(openContactApp, 123);
        }
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode==123&& resultCode == RESULT_OK) {

            String[] columns = new String[]{
                    ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
                    ContactsContract.CommonDataKinds.Phone.NUMBER
            };
            ContentResolver resolver = getContentResolver();
            Uri dataUri = data.getData();
            Cursor cursor = resolver.query(dataUri, columns, null, null, null);
            if (cursor.moveToFirst()) {

                @SuppressLint("Range") String name = cursor.getString(cursor.getColumnIndex(columns[0]));
                Log.i("name", "" + name);

                @SuppressLint("Range") String contact = cursor.getString(cursor.getColumnIndex(columns[1]));
                Log.i("contact", "" + contact);

                TextView tvname = findViewById(R.id.textView);
                tvname.setText(name);
                TextView tvcontact = findViewById(R.id.textView2);
                tvcontact.setText(contact);
            } else {
                Toast.makeText(this, "Unable to load contact", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Contact not selected", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Conclusion: - Hence successfully performed Developing Android applications using Receivers and Content Providers.