UPES
UNIVERSITY WITH A PURPOSE

School of Computer Science
Department of Cybernetics

Assignment No. 3

Name: Prashant Mishra
Subject: Open Source Mobile Platform
Roll No: R100217103
Program/Branch: B.Tech./CSE-OSOS
SAP id: 500060122
Faculty: Pushpendra K Rajput
Batch: B4

Q.1 List out different methods to make data persistence in android with their sui tability. Demonstrate use of shared preferences to store.

Ans. There are basically four different ways to store data in an Android app:

1. Shared Preferences: You should use this to save primitive data in key-value pairs. You have a key, which must be a String, and the corresponding value for that key, which can be one of: boolean, float, int, long or string. Internally, the Androi d platform stores an app's Shared Preferences in an xml file in a private directory. An app can have multiple Shared Preferences files. Ideally, you will want to use Shared preferences to store application preferences.

2. Internal Storage: There are lots of situations where you might want to persist data but Shared Preferences is too limiting. You may want to persist Java objects, or image s. Or your data logically needs to be persisted using the familiar filesystem hierarchy. The Internal Storage data storage method is specifically for those situations where you need to store data to the device filesystem, but you do not want any other app (even t he user) to read this data. Data stored using the Internal Storage method is completely private to your application, and are deleted from the device when your app is uninstall ed.

3. External Storage: Conversely, there are other instances where you might want the u ser to view the files and data saved by your app, if they wish. To save (and/or read) files to the device's external storage, your app must request for the WRITE_EXTERNA L_STORAGE permission. If you only want to read from the External Storage without writing, request for the READ_EXTERNAL_STORAGE permission.

4. SQLite database: Finally, Android provides support for apps to use SQLite database s for data storage. Databases created are app specific, and are available to any class wi thin the app, but not to outside applications. It goes without saying, that before you de cide to use an SQLite database for data storage in your app, you should have some S QL knowledge.

Using Shared Preferences Shared Preferences in Android:

It is a place where you can store data; this is an XML file with values stored in it. A lmost all applications need to store some data. Data can be a lot of different things; it can be just an email for a registration form, the last opened screen, nickname for a g

ame or a proper database. If you are not ready to use a full-blown database yet (or just don't need it), you can use SharedPreferences to store simple data. But you can also store a bit more complex data if you want.

SharedPreferences are not intended to store a lot of data, there is no limit per se (since it is an xml file), but for larger sets of data, I would suggest using Room (or SQLite for the older projects). There is also another reason why storing in a database makes more sense. For example the structure, tables and relations, primary/foreign keys… but this is a whole other topic, and today we will store data in SharedPreferences anyway .

Quick steps to store data with SharedPreferences:

- Get preferences for your context (sharedPref = getSharedPreferences)
- get editor for sharedPreferences (editor = sharedPref.edit())
- Store data (editor.putInt(10))
- Apply changes (editor.apply();)

Q.2 What is SQLite database? How to create database connection using SQLite database?

Ans. SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database. It is embedded in android by default. So, there is no need to perform any database setup or administration task.

Simple steps to create a database are as follows.

- Create "SQLiteDatabase" object.
- Open or Create a database and create a connection.
- Perform insert, update or delete operation.
- Create a Cursor to display data from the table of the database.
- Close the database connectivity.

```
package com.DataBaseDemo;

import android.app.Activity;

import android.content.ContentValues;

import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;
```

```
public class DataBaseDemoActivity extends Activity {
 SQLiteDatabase db;
 Button btnInsert;
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 btnInsert=(Button)findViewById(R.id.button1);
 try{
                  db=openOrCreateDatabase("StudentDB",SQLiteDatabase.CREATE_IF_NEC
                ESSARY,null);
 db.execSQL("Create Table Temp(id integer,name text)");
 }
 catch(SQLException e)
 {
 }
 btnInsert.setOnClickListener(new View.OnClickListener()
 {
 @Override
 public void onClick(View v)
 {

 EditText eid=(EditText) findViewById(R.id.editText1);
 EditText ename=(EditText)findViewById(R.id.editText2);
 ContentValues values=new ContentValues();
 values.put("id", eid.getText().toString());
 values.put("name", ename.getText().toString());
 if((db.insert("temp", null, values))!=-1)
 {
 Toast.makeText(DataBaseDemoActivity.this, "Record
 Successfully Inserted", 2000).show();
 }
 else
 {
 Toast.makeText(DataBaseDemoActivity.this, "Insert Error", 2000).show();
```

```
 }
 eid.setText("");
 ename.setText("");
 Cursor c=db.rawQuery("SELECT * FROM temp",null);
 c.moveToFirst();
 while(!c.isAfterLast())
 {
 Toast.makeText(DataBaseDemoActivity.this,c.getString(0)+ " "+c.getString(1),1000).show(
);
 c.moveToNext();
 }
 c.close();
 }
 });
 }
 @Override
 protected void onStop() {
 db.close();
 super.onStop();
 }
}
```

Q.3 Explain Content Provider in Android application development.

Ans. Content Providers are an important component of Android. They handle the acces s to the central repository and supply data from one application to another on request. This task of handling is done by methods of ContentResolver class. So, content provide rs can store data in various ways such as files, database or over the internet. Content providers act the same as database and also we can query it to add, delete, insert or u pdate the data.

It can be understood that a content provider hides the database details and also, it lets an application share data among other applications. Content providers are not limited to texts, but also contains images and videos as well.

The above are the four operations of content providers CRUD:

- Create: It is used for the creation of data in content providers.
- Read: It reads the data stored in the content provider.
- Update: It lets the editing in existing data in content providers.
- Delete: It deletes the existing data stored in its Storage.

Q.4 Discuss List View in detail. Write a java class to add any 10 items within Lis t View.

Ans. Android ListView is a view which groups several items and display them in verti cal scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database. An adapter actually bri dges between UI components and the data source that fill data into UI Component. Ad apter holds the data and send the data to adapter view, the view can takes the data fro m adapter view and shows the data on different views like as spinner, list view, grid view etc.

```
public class MainActivity extends Activity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 final ListView lv = (ListView) findViewById(R.id.lv);
 final Button btn = (Button) findViewById(R.id.btn);

 String[] fruits = new String[] {
 "Cape Gooseberry",
 "Capuli cherry"
 };
 final List<String> fruits_list = new ArrayList<String>(Arrays.asList(fruits));

 final ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>
 (this, android.R.layout.simple_list_item_1, fruits_list);

 lv.setAdapter(arrayAdapter);

 btn.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
 fruits_list.add("Loquat");
 fruits_list.add("Pear");
          fruits_list.add("Mango");
                fruits_list.add("Banana");
```

```
                fruits_list.add("Pear");
                fruits_list.add("Grapes");
                fruits_list.add("Plum");
                fruits_list.add("Orange");
                fruits_list.add("Honey  Dew");
                fruits_list.add("Guava");

                arrayAdapter.notifyDataSetChanged();
  }
  });
  }
}
```