Lab Android

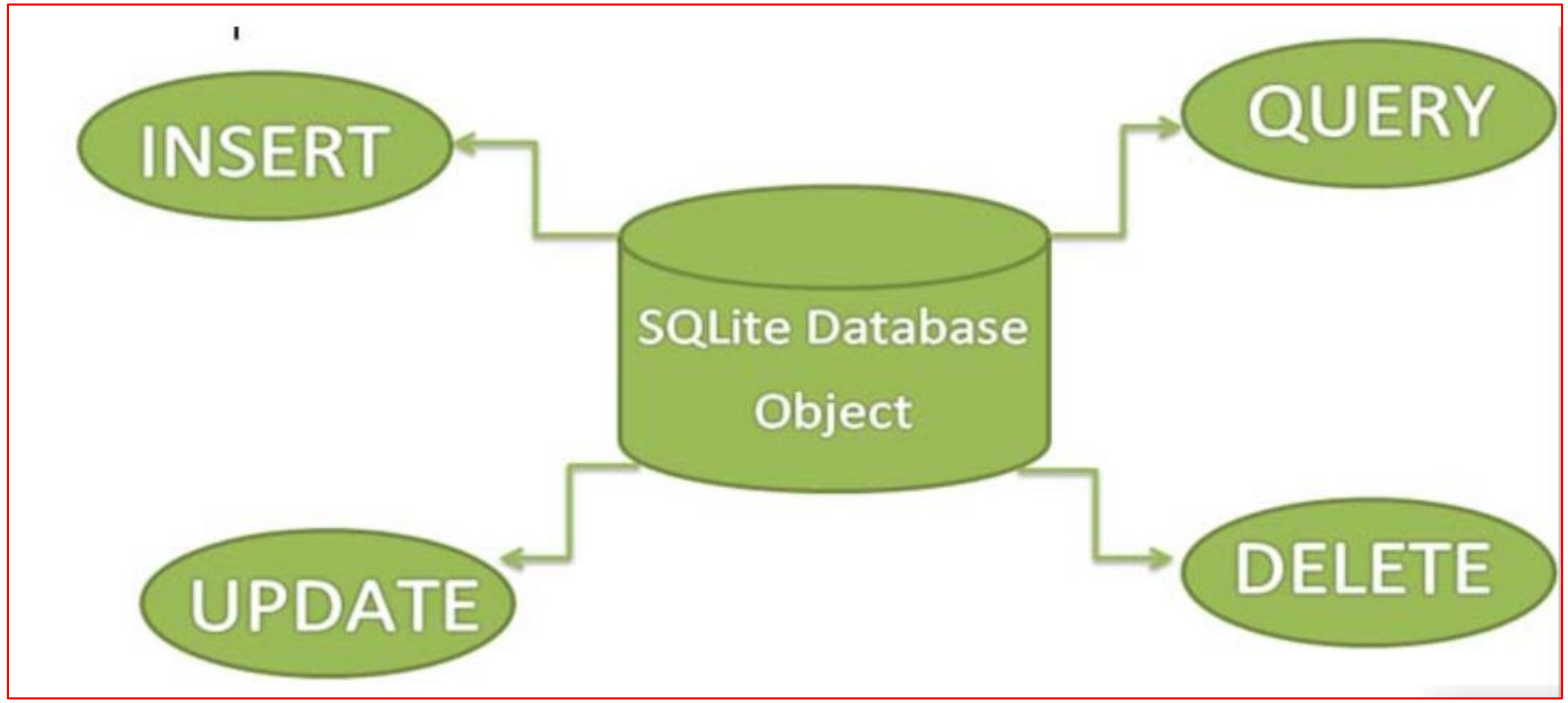
SQLite Database

SQLite in Android

- SQLite is a Structure query base database, open source, light weight, no network access and standalone database. It support embedded relational database features.
- **The database created is saved in a directory: data/data/APP_Name/databases/DATABASE_NAME.**
- /data/data/<package_name>/databases

1. Run cmd as administrator and
   cd users\username\AppData\Local\Android\sdk\platform-tools
2. C:\Users\username\AppData\Local\Android\sdk\platform-tools>adb devices
   cmd: List of device
   emulator-xxxx device ------ > This is your device and then,
3. C:\Users\username\AppData\Local\Android\sdk\platform-tools>adb -s emulator-xxxx shell
4. - shell: generic_x86:/ cd <package_name for example (com.bignerdranch.android.criminalintent)>
5. generic_x86:/data/data/com.bignerdranch.android.criminalintent cd databases
   cache databases
6. generic_x86:/data/data/com.bignerdranch.android.criminalintent/databases $ ls
   crimeBase.db crimeBase.db-journal

Database Functions



SQLiteOpenHelper class

- For creating, updating and other operations you need to create a subclass or *SQLiteOpenHelper* class. SQLiteOpenHelper is a helper class to manage database creation and version management. It provides two methods onCreate(SQLiteDatabase db), onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion).
- The SQLiteOpenHelper is responsible for opening database if exist, creating database if it does not exists and upgrading if required. The SQLiteOpenHelper only require the DATABASE_NAME to create database.
- After extending SQLiteOpenHelper you will need to implement its methods onCreate, onUpgrade and constructor.

onCreate(SQLiteDatabase sqLiteDatabase) & onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion)

- **onCreate(SQLiteDatabase sqLiteDatabase)** *method is called only once throughout the application lifecycle. It will be called whenever there is a first call to getReadableDatabase() or getWritableDatabase() function available in super SQLiteOpenHelper class.*
- So **SQLiteOpenHelper** *class call the onCreate() method after creating database and instantiate SQLiteDatabase object. Database name is passed in constructor call.*
- **onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion)** *is only called whenever there is a updation in existing version. So to update a version we have to increment the value of version variable passed in the superclass constructor.*

Create New Project

Configure your project

Empty Activity

Name

MyDatabaseApp

Package name

com.example.mydatabaseapp

Save location

C:\Users\deshinta.ad\MyDatabaseApp

Language

Java

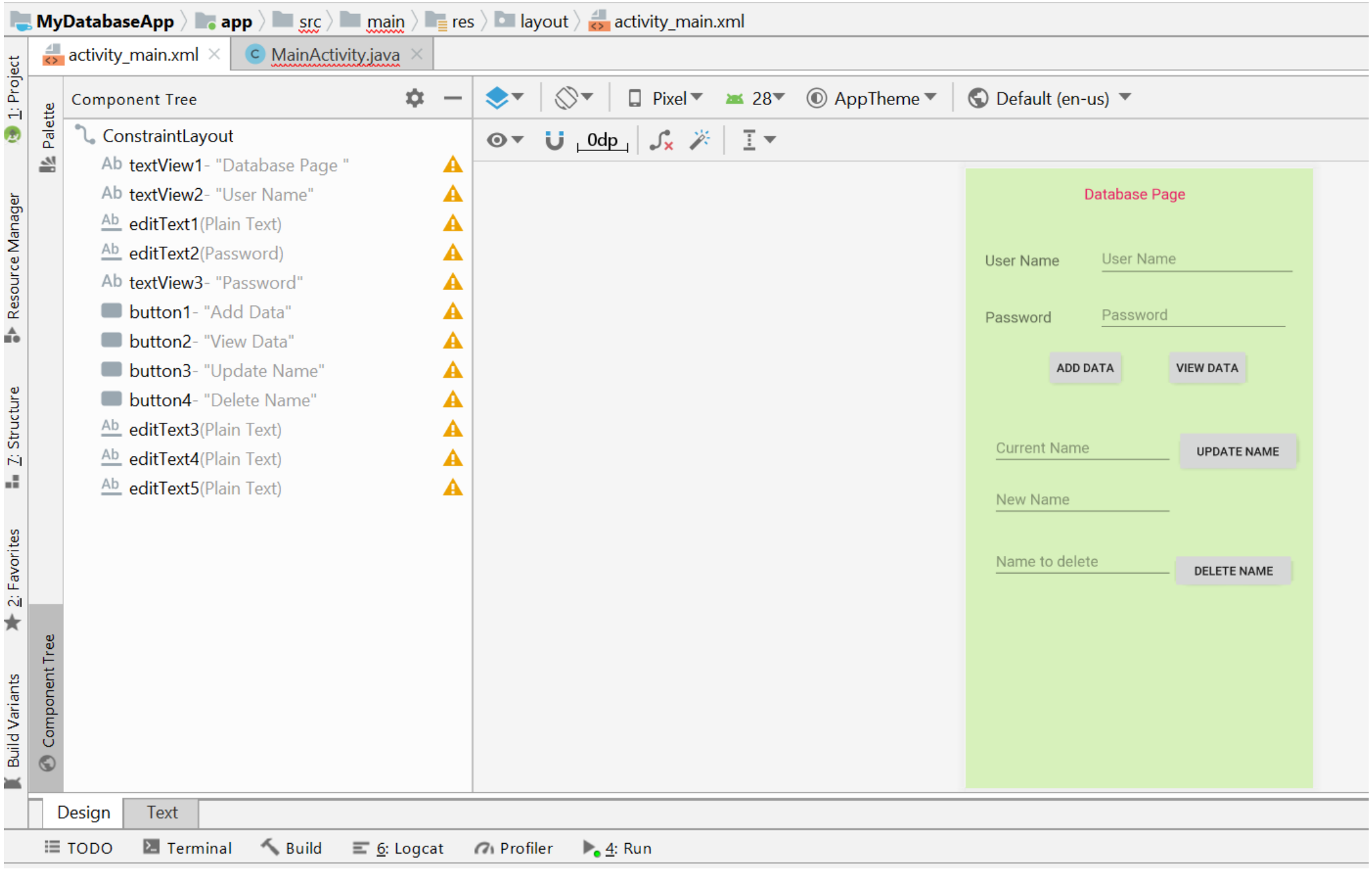
Minimum API level

API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

Previous | Next | Cancel | Finish


MyDatabaseApp
app
src
main
res
layout
activity_main.xml
activity_main.xml
MainActivity.java
Component Tree
ConstraintLayout
textView1- "Database Page "
textView2- "User Name"
editText1(Plain Text)
editText2(Password)
textView3- "Password"
button1- "Add Data"
button2- "View Data"
button3- "Update Name"
button4- "Delete Name"
editText3(Plain Text)
editText4(Plain Text)
editText5(Plain Text)
Pixel
28
AppTheme
Default (en-us)
0dp
Database Page
User Name
User Name
Password
Password
ADD DATA
VIEW DATA
Current Name
UPDATE NAME
New Name
Name to delete
DELETE NAME
Design
Text
TODO
Terminal
Build
6: Logcat
Profiler
4: Run
1: Project
Resource Manager
7: Structure
2: Favorites
Build Variants
Palette
Component Tree

MyDatabaseApp > app > src > main > java > com > example > mydatabaseapp > MainActivity

activity_main.xml | MainActivity.java

```java
package com.example.mydatabaseapp;

import ...

public class MainActivity extends AppCompatActivity {

    EditText Name, Pass , updateold, updatenew, delete;
    myDbAdapter helper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Name= (EditText) findViewById(R.id.editText1);
        Pass= (EditText) findViewById(R.id.editText2);
        updateold= (EditText) findViewById(R.id.editText3);
        updatenew= (EditText) findViewById(R.id.editText4);
        delete = (EditText) findViewById(R.id.editText5);
        helper = new myDbAdapter(this);
    }



}
```

Need to create a class myDBAdapter()

Function addUser ()

MyDatabaseApp > app > src > main > java > com > example > mydatabaseapp > MainActivity

activity_main.xml | MainActivity.java

```
23          helper = new myDbAdapter(this);
24      } //on create
25
26      public void addUser(View view)
27      {
28          String t1 = Name.getText().toString();
29          String t2 = Pass.getText().toString();
30          if(t1.isEmpty() || t2.isEmpty())
31          {
32              Message.message(getApplicationContext(),"Enter Both Name and Password");
33          }
34          else
35          {
36              long id = helper.insertData(t1,t2);
37              if(id<=0)
38              {
39                  Message.message(getApplicationContext(),"Insertion Unsuccessful");
40                  Name.setText("");
41                  Pass.setText("");
42              } else
43              {
44                  Message.message(getApplicationContext(),"Insertion Successful");
45                  Name.setText("");
46                  Pass.setText("");
47              }
48          }
49      }
```

MainActivity > addUser()

Need to create a class Message()

Function viewdata ()

```
51      public void viewdata(View view)
52      {
53          String data = helper.getData();
54          Message.message(this,data);
55      }
56
```

Function update ()

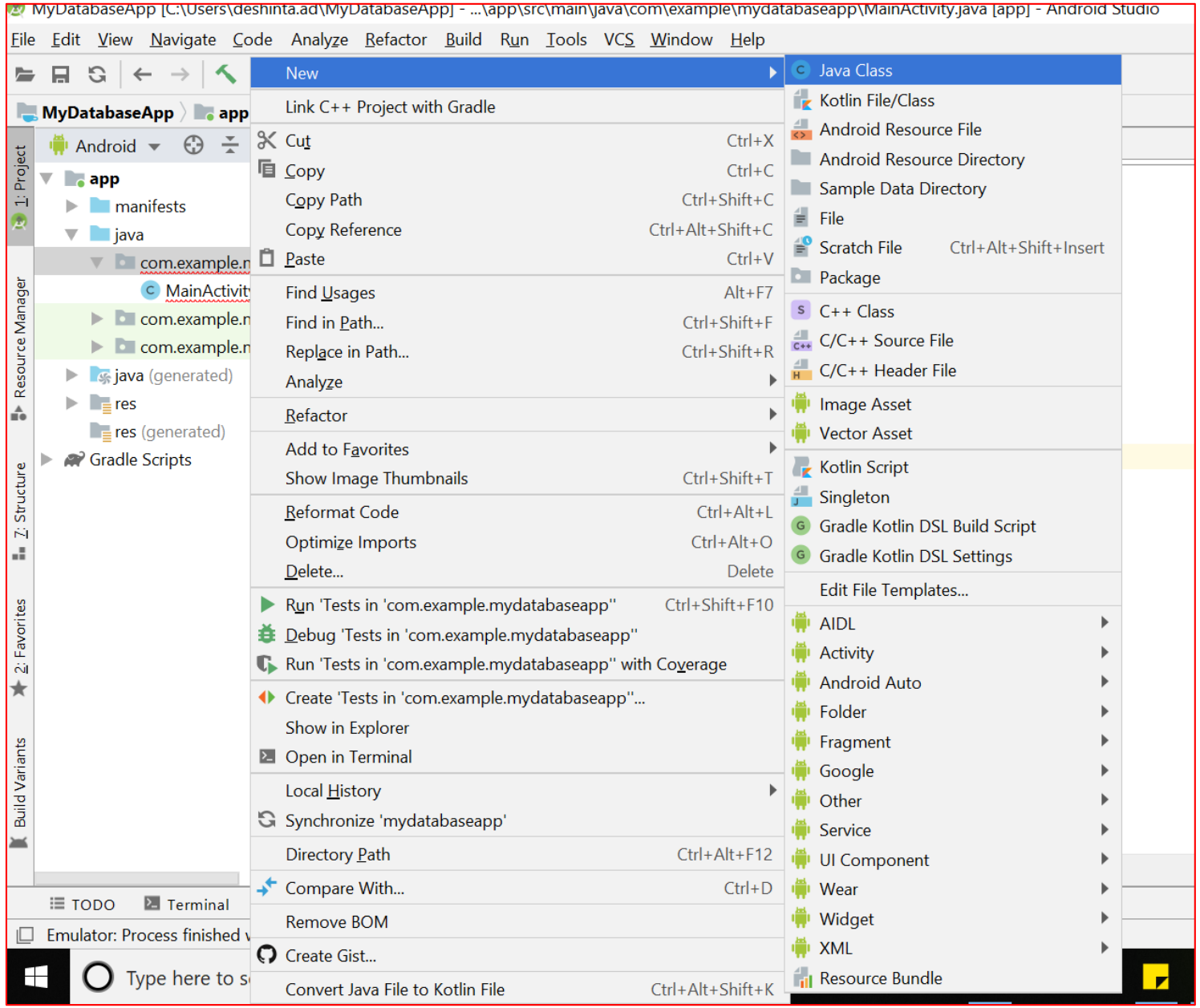
```
57      public void update( View view)
58      {
59          String u1 = updateold.getText().toString();
60          String u2 = updatenew.getText().toString();
61          if(u1.isEmpty() || u2.isEmpty())
62          {
63              Message.message(getApplicationContext(),"Enter Data");
64          }
65          else
66          {
67              int a= helper.updateName( u1, u2);
68              if(a<=0)
69              {
70                  Message.message(getApplicationContext(),"Unsuccessful");
71                  updateold.setText("");
72                  updatenew.setText("");
73              } else {
74                  Message.message(getApplicationContext(),"Updated");
75                  updateold.setText("");
76                  updatenew.setText("");
77              }
78          }
79
80      }
```

Function delete ()

activity_main.xml | MainActivity.java | myDbAdapter.java | Message.java

```
80      }
81      public void delete( View view)
82      {
83          String uname = delete.getText().toString();
84          if(uname.isEmpty())
85          {
86              Message.message(getApplicationContext(), message: "Enter Data");
87          }
88          else{
89              int a= helper.delete(uname);
90              if(a<=0)
91              {
92                  Message.message(getApplicationContext(), message: "Unsuccessful");
93                  delete.setText("");
94              }
95              else
96              {
97                  Message.message( context: this,  message: "DELETED");
98                  delete.setText("");
99              }
100         }
101     }
102
103 } // main Activity
104
```

Create a new class namely myDBAdapter



Create New Class

Name: myDbAdapter

Kind: Class

Superclass:

Interface(s):

Package: com.example.mydatabaseapp

Visibility: Public Package Private

Modifiers: None Abstract Final

Show Select Overrides Dialog

OK Cancel Help

```java
public class myDbAdapter {

    myDbHelper myhelper;
    public myDbAdapter(Context context)
    {
        myhelper = new myDbHelper(context);
    }

    public long insertData(String name, String pass)
    {
        SQLiteDatabase dbb = myhelper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(myDbHelper.NAME, name);
        contentValues.put(myDbHelper.MyPASSWORD, pass);
        long id = dbb.insert(myDbHelper.TABLE_NAME, nullColumnHack: null , contentValues);
        return id;
    }

```

```java
    public String getData()
    {
        SQLiteDatabase db = myhelper.getWritableDatabase();
        String[] columns = {myDbHelper.UID,myDbHelper.NAME,myDbHelper.MyPASSWORD};
        Cursor cursor =db.query(myDbHelper.TABLE_NAME,columns, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
        StringBuffer buffer= new StringBuffer();
        while (cursor.moveToNext())
        {
            int cid =cursor.getInt(cursor.getColumnIndex(myDbHelper.UID));
            String name =cursor.getString(cursor.getColumnIndex(myDbHelper.NAME));
            String  password =cursor.getString(cursor.getColumnIndex(myDbHelper.MyPASSWORD));
            buffer.append(cid+ "    " + name + "    " + password +" \n");
        }
        return buffer.toString();
    }

    public  int delete(String uname)
    {
        SQLiteDatabase db = myhelper.getWritableDatabase();
        String[] whereArgs ={uname};

        int count =db.delete(myDbHelper.TABLE_NAME , whereClause: myDbHelper.NAME+" = ?",whereArgs);
        return  count;
    }
```

```
52      public int updateName(String oldName , String newName)
53      {
54          SQLiteDatabase db = myhelper.getWritableDatabase();
55          ContentValues contentValues = new ContentValues();
56          contentValues.put(myDbHelper.NAME,newName);
57          String[] whereArgs= {oldName};
58          int count =db.update(myDbHelper.TABLE_NAME,contentValues,  whereClause: myDbHelper.NAME+" = ?",whereArgs );
59          return count;
60      }
61
```

```java
62      static class myDbHelper extends SQLiteOpenHelper
63      {
64          private static final String DATABASE_NAME = "myDatabase";    // Database Name
65          private static final String TABLE_NAME = "myTable";   // Table Name
66          private static final int DATABASE_Version = 1;    // Database Version
67          private static final String UID="_id";     // Column I (Primary Key)
68          private static final String NAME = "Name";    //Column II
69          private static final String MyPASSWORD= "Password";    // Column III
70          private static final String CREATE_TABLE = "CREATE TABLE "+TABLE_NAME+
71                  " ("+UID+" INTEGER PRIMARY KEY AUTOINCREMENT, "+NAME+" VARCHAR(255) ,"+ MyPASSWORD+" VARCHAR(225));";
72          private static final String DROP_TABLE ="DROP TABLE IF EXISTS "+TABLE_NAME;
73          private Context context;
74
75          public myDbHelper(Context context) {
76              super(context, DATABASE_NAME, factory: null, DATABASE_Version);
77              this.context=context;
78          }
```

```
80          public void onCreate(SQLiteDatabase db) {
81
82              try {
83                  db.execSQL(CREATE_TABLE);
84              } catch (Exception e) {
85                  Message.message(context,""+e);
86              }
87          }
88
89          @Override
90          public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
91              try {
92                  Message.message(context,"OnUpgrade");
93                  db.execSQL(DROP_TABLE);
94                  onCreate(db);
95              }catch (Exception e) {
96                  Message.message(context,""+e);
97              }
98          }
99      }
100 }
101
```

activity_main.xml | MainActivity.java | myDbAdapter.java | Message.java

```java
package com.example.mydatabaseapp;

import android.content.Context;
import android.widget.Toast;

public class Message {
    public static void message(Context context, String message) {
        Toast.makeText(context, message, Toast.LENGTH_LONG).show();
    }
}

```

Update XML files for all buttons

Add the following code "onClick" for each button

```
android:text="Add Data"

Put this code:
android:onClick="addUser"
```

```
android:text="Update Name"

Put this code:
android:onClick="update"
```

```
android:text="View Data"

Put this code:
android:onClick="viewdata"
```

```
android:text="Delete Name"

Put this code:
android:onClick="delete"
```

Some got this, some ok, no worries

MyDatabaseApp [C:\Users\deshinta.ad\MyDatabaseApp] - C:\Users\deshinta.ad\AppData\Local\Android\Sdk\platforms\android-28\android.jar!\android\database\sqlite\SQLiteOpenHelper.class [Android API 2...

File Edit View Navigate Code Analyze Refactor Build

app | Nexus S API 2

android.jar > android > database > sqlite > SQLi

activity_main.xml | MainActivity.java | myDb

Decompiled .class file, bytecode version: 52.0 (Java 8)

Sources for 'Android API 28 Platform' not found.

```
//
// Source code recreated from a .c
// (powered by Fernflower decompil
//

package android.database.sqlite;

import android.content.Context;
import android.database.DatabaseEr
import android.database.sqlite.SQL
import android.database.sqlite.SQL
import androidx.annotation.Recentl
import androidx.annotation.Recentl

public abstract class SQLiteOpenHe
    public SQLiteOpenHelper(@Recen
        throw new RuntimeException
    }

    public SQLiteOpenHelper(@Recen
        throw new RuntimeException
    }

    public SQLiteOpenHelper(@Recen
```

SQLiteOpenHelper > onCreate()

int version

int version

openParams)

SDK Quickfix Installation

Component Installer

Android Studio

Installing Requested Components

SDK Path: C:\Users\deshinta.ad\AppData\Local\Android\Sdk

```
Packages to install:
- Sources for Android 28 (sources;android-28)

Preparing "Install Sources for Android 28 (revision: 1)".
Downloading https://dl.google.com/android/repository/sources-28_r01.zip
```

Downloading (3%): 1.2 / 40.6 MB ...

https://dl.google.com/android/repository/sources-28_r01.zip

Please wait until the installation finishes

Previous | Next | Cancel | Finish

TODO | Terminal | Build | 6: Logcat | Profiler | 4: Run | Event Log

Emulator: Process finished with exit code 0 (today 7:21 PM)

Downloading (3%): 1.2 / 40.6 MB ...

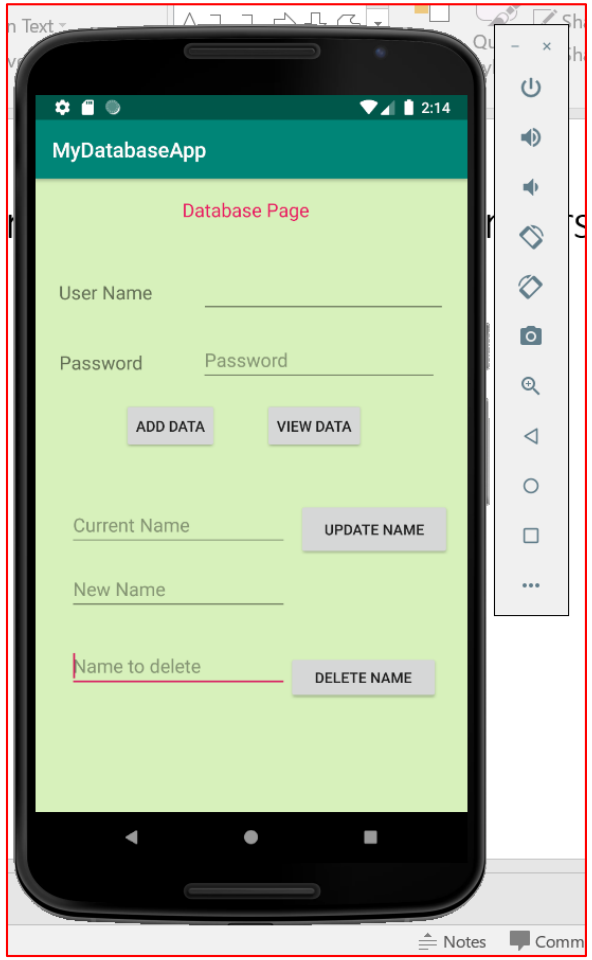
64:26 LF UTF-8

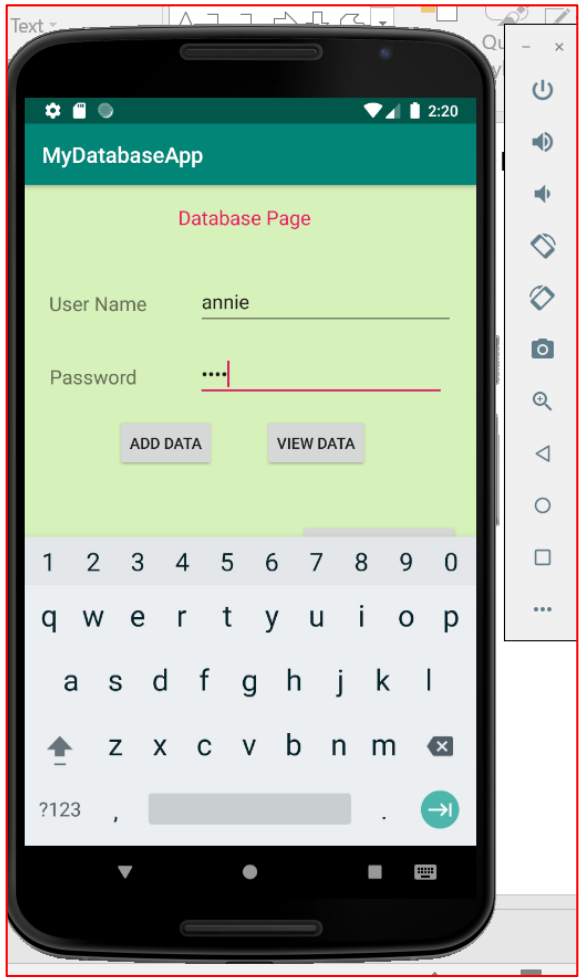
Type here to search

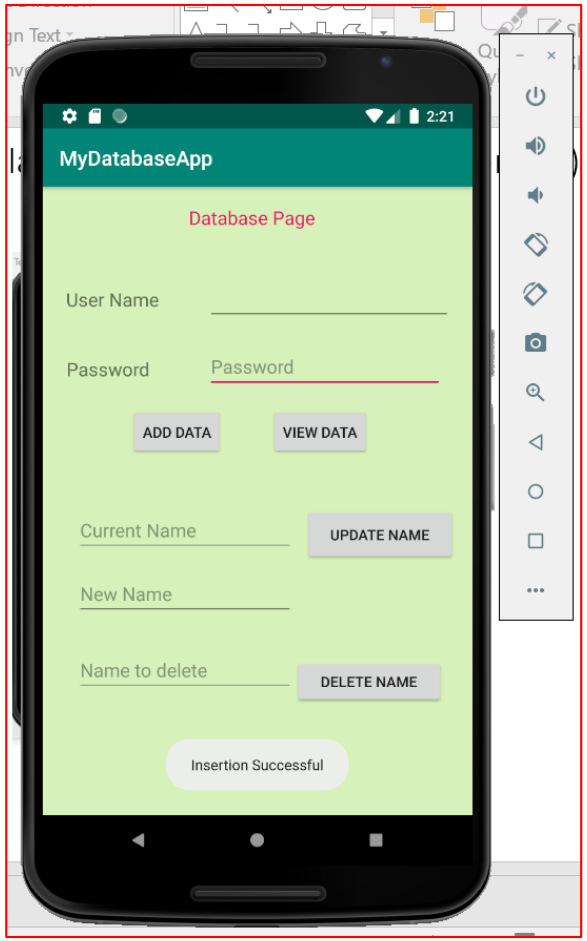
9:08 PM 10/6/2019

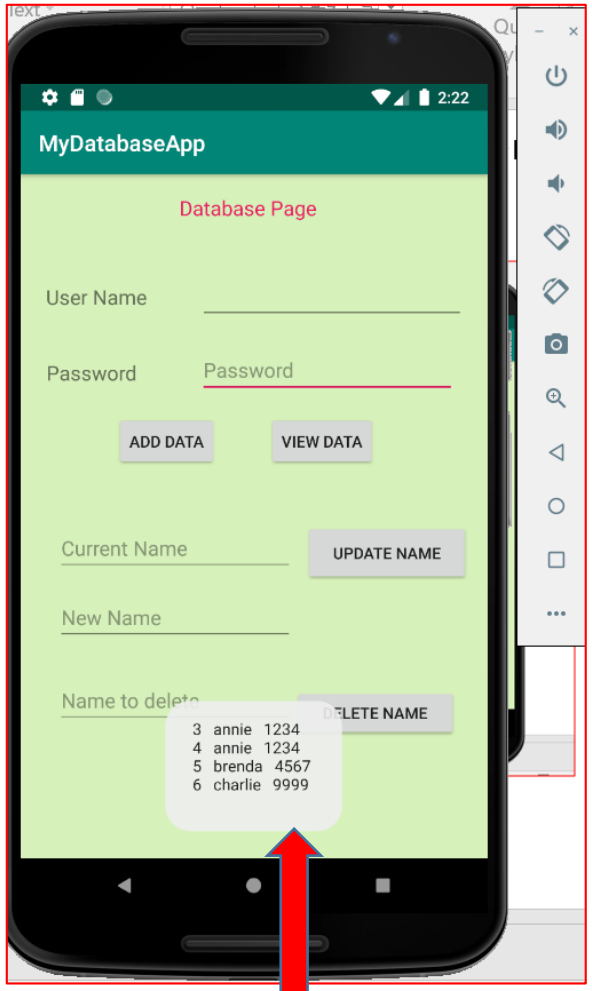
Emulator: Nexus 6, API 27 (bigger version)

Add Data Operation



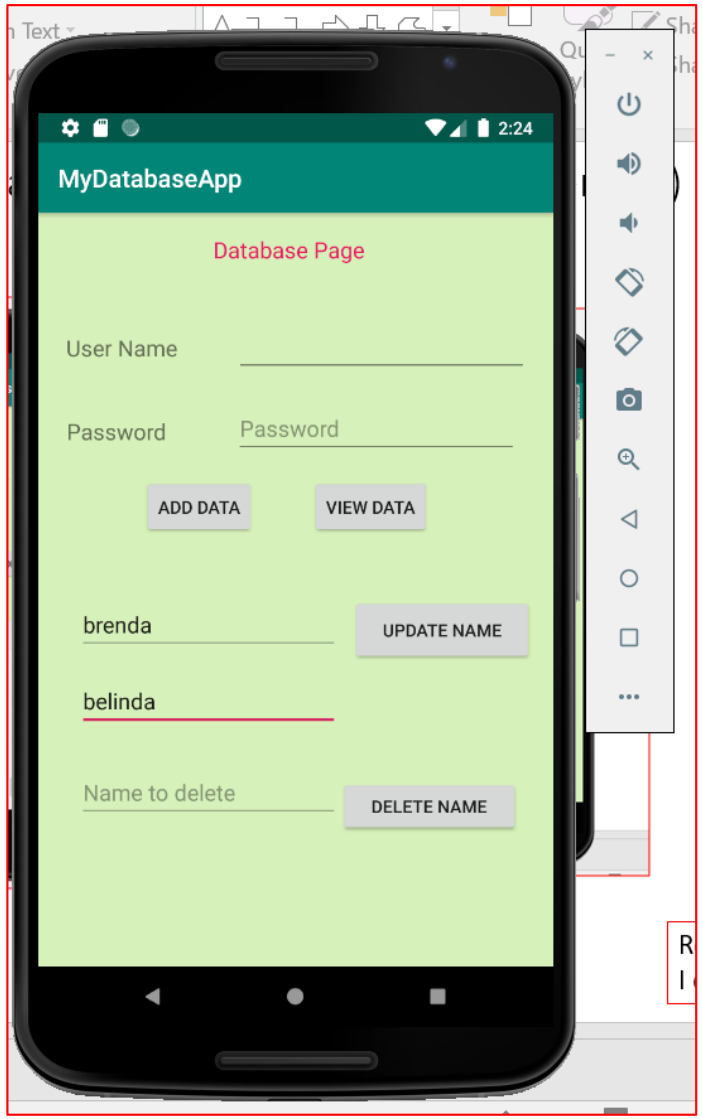


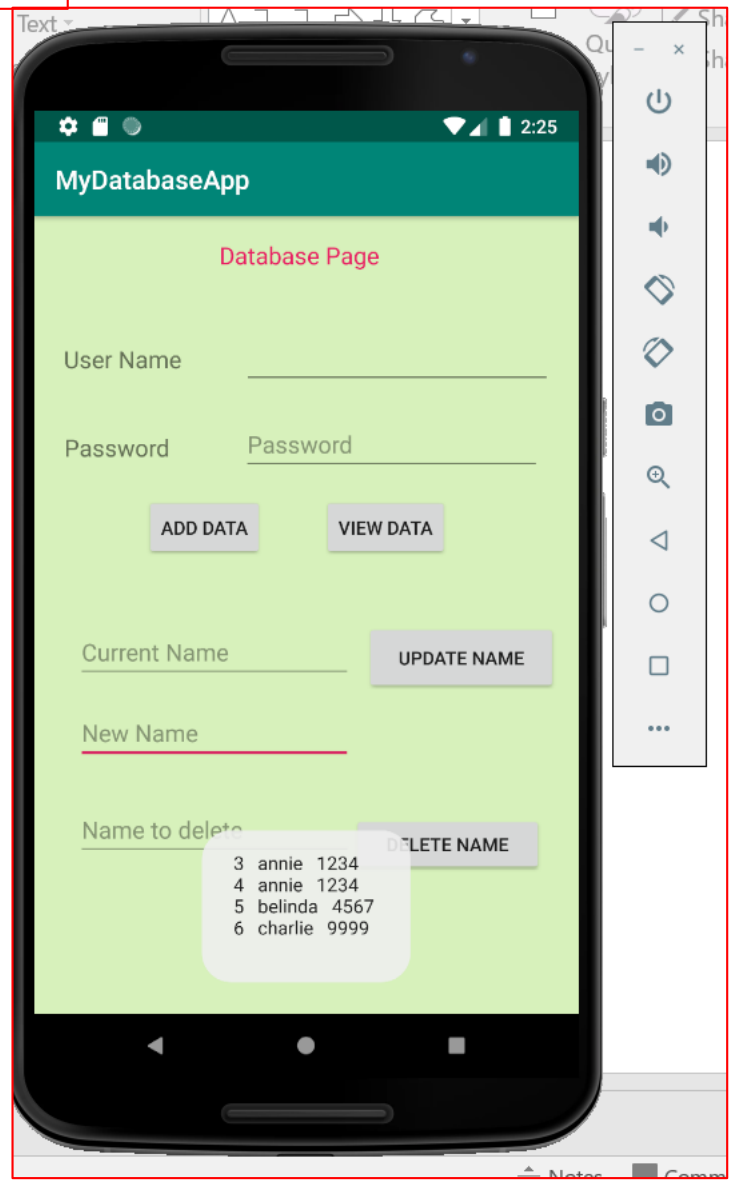




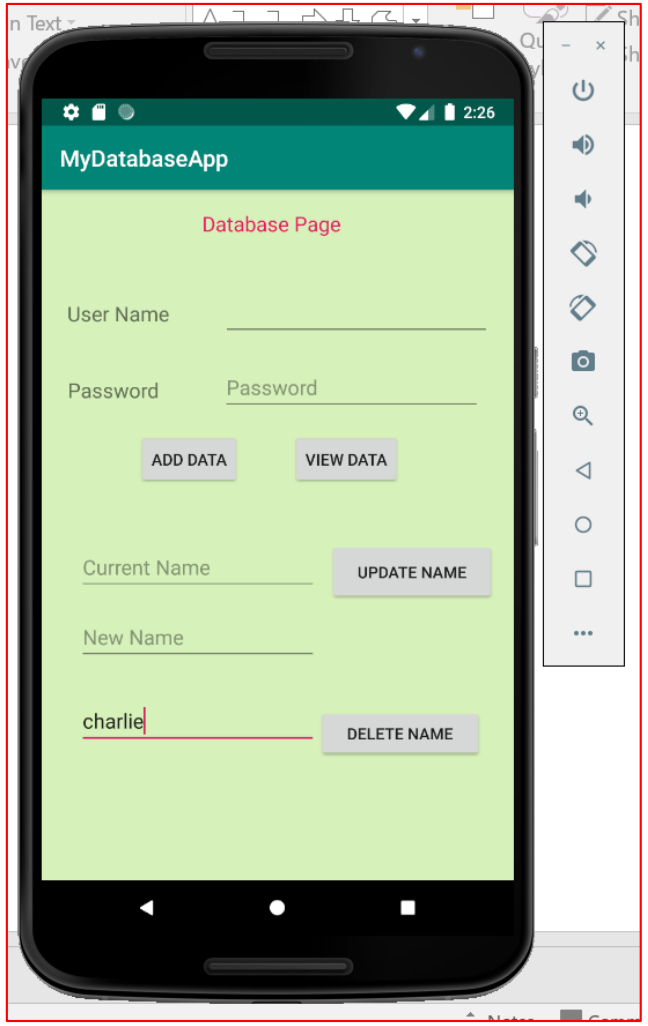
Record starts from 3, because I deleted record 1 and 2 before.

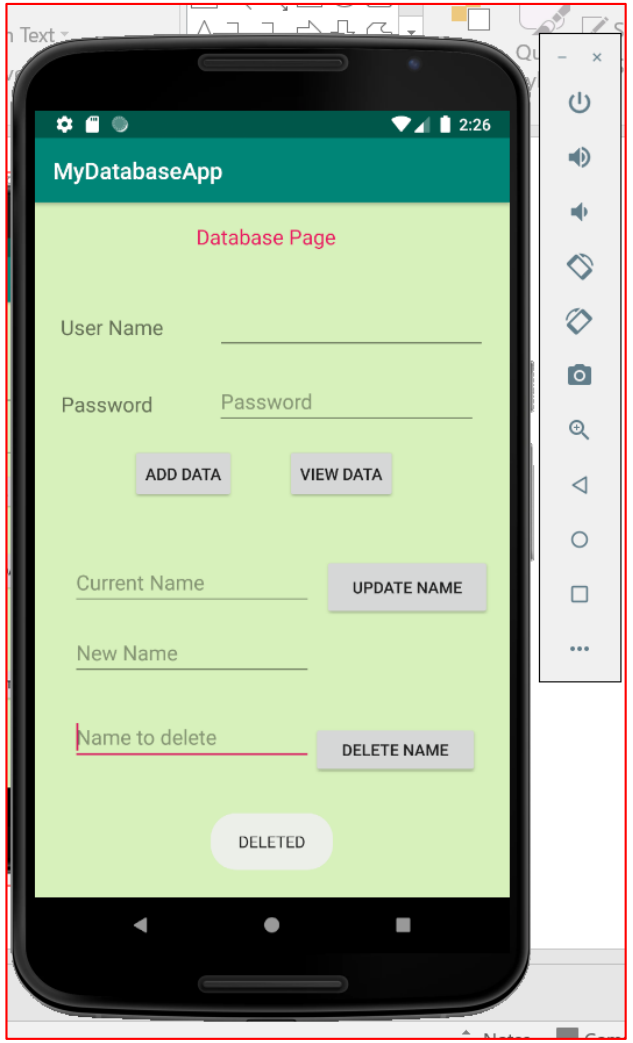
Modify / Update Name Operation

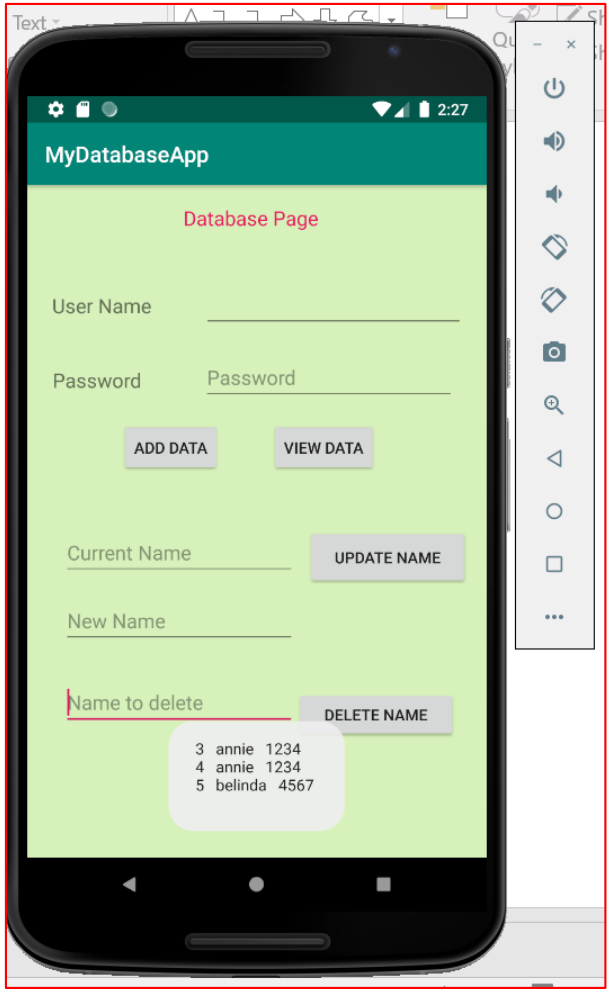




Delete Operation





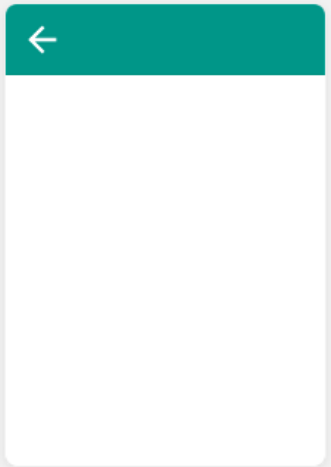


More Complex Coding SQLite Database

We will create a database with a spinner, a listview and dialogs

Create New Project

Configure your project



Empty Activity

Name

MyDatabaseManager

Package name

com.example.mydatabasemanager

Save location

C:\Users\deshinta.ad\MyDatabaseManager

Language

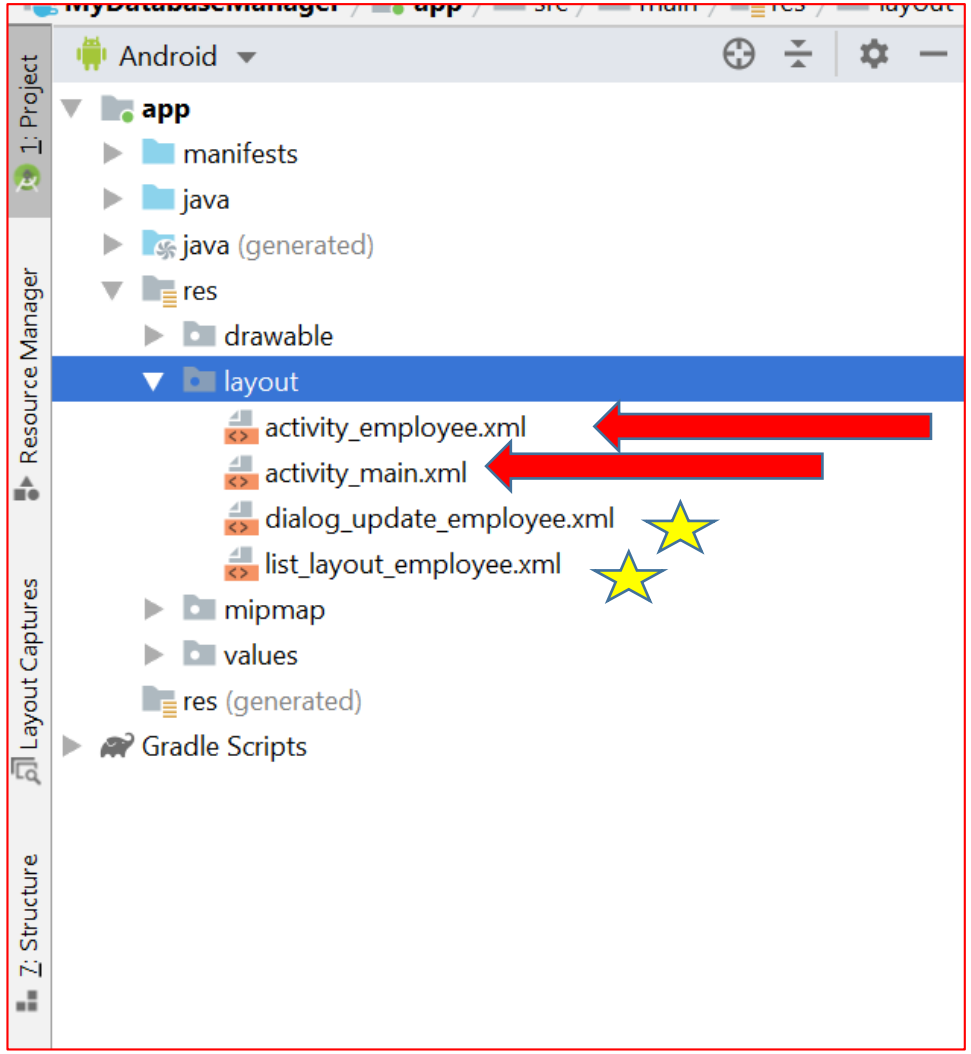
Java

Minimum API level API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

Previous Next Cancel Finish



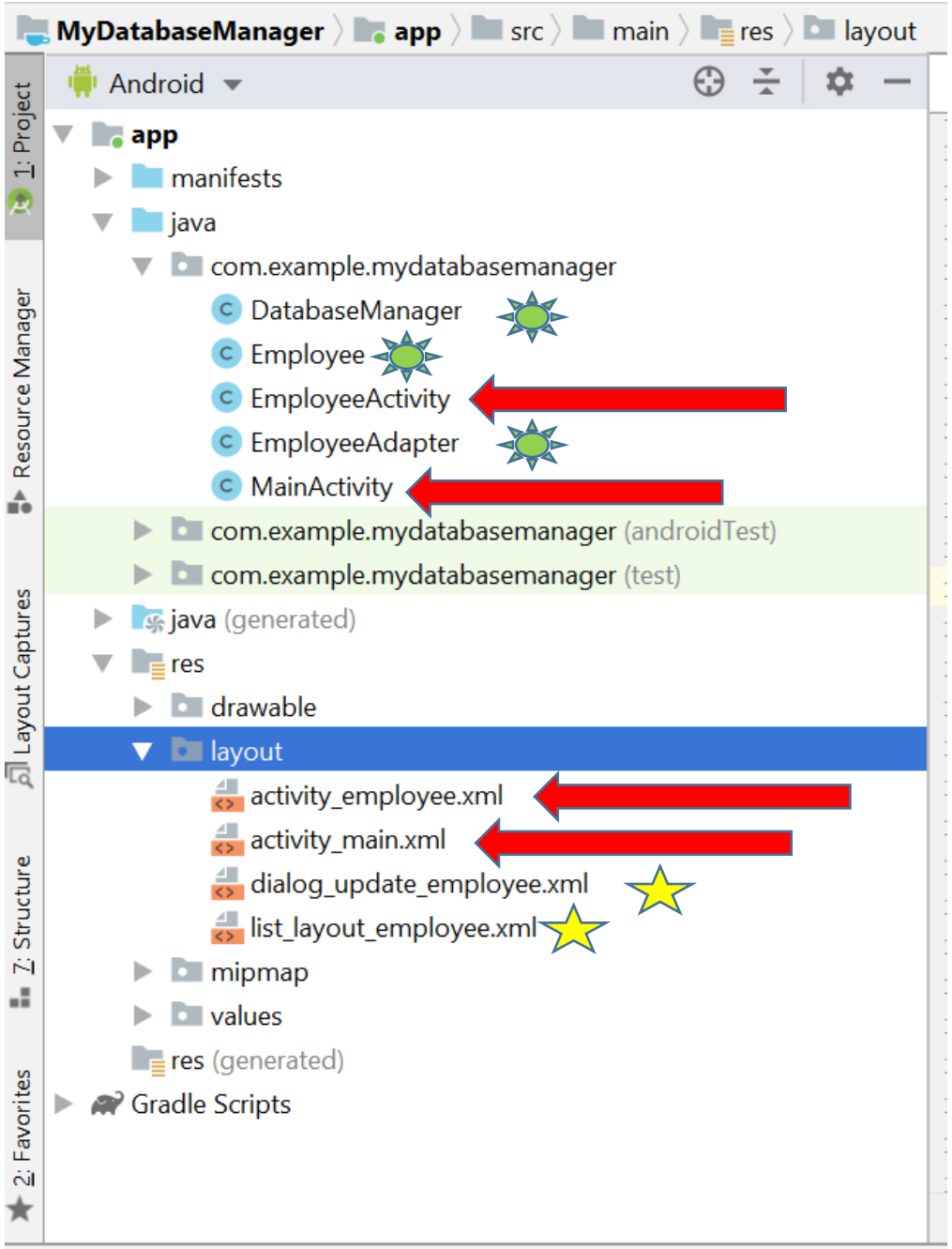
We need 4 XML files

<u>2 new layout activities, will auto-generate related Java file</u>

(activity_employee.xml)
(activity_main.xml)

<u>2 XML layout (without activity), will not generate Java file</u>
Or we call it stand_alone XML layout file

(dialog_update_employee.xml)
(list_layout_employee.xml)



We need 5 Java files
2 is generated from layout activities
(EmployeeActivity.java)
(MainActivity.Java)

3 is new Java class (stand-alone Java class)
(DatabaseManager.Java)
(Employee.Java)
(Employee Adapter.Java)

Let us create ALL XML files first there are 4 of them

MyDatabaseManager > app > src > main > res > layout > activity_main.xml

activity_main.xml

Component Tree

- ConstraintLayout
  - LinearLayout(vertical)
    - textView- "Add a new employee"
    - editText(Plain Text)
    - textView2
    - spinnerDepartment
    - editText2(Number)
    - button- "Add Employee"
    - button4- "View Employee"

Activity_main.xml

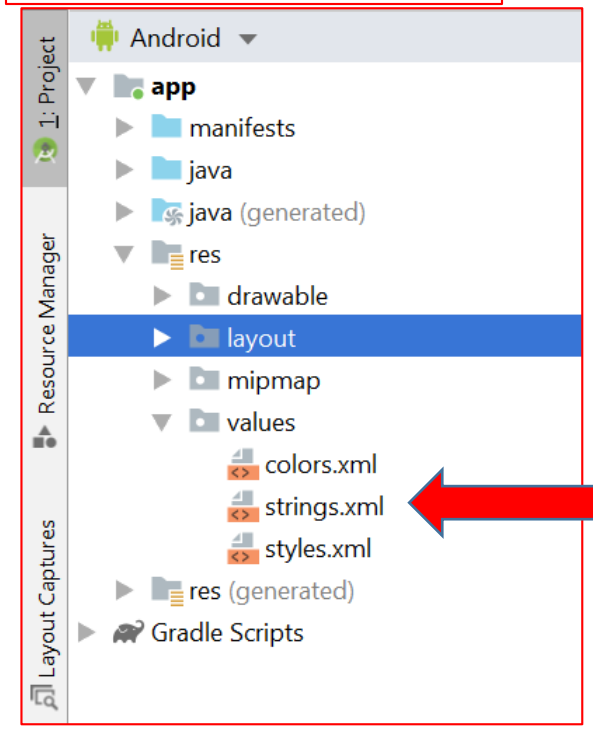
Do all, but spinner. Spinner code on next slide

Activity_main.xml

Reuse this spinner code

```
<Spinner
    android:id="@+id/spinnerDepartment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/departments" />
```

Go here after you complete the design



Reuse this code for string.xml

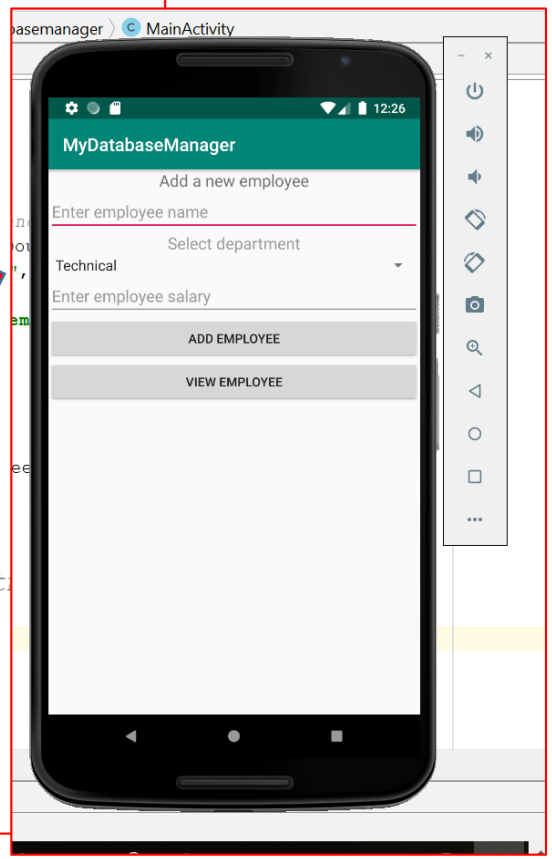
```
<resources>
    <string name="app_name">MyDatabaseManager</string>
    <array name="departments">
        <item>Technical</item>
        <item>Support</item>
        <item>Research and Development</item>
        <item>Marketing</item>
        <item>Human Resource</item>
    </array>
</resources>
```

Activity_main.xml is associated with MainActivity.Java.
Let us do coding for MainActivity.Java

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 MainActivity

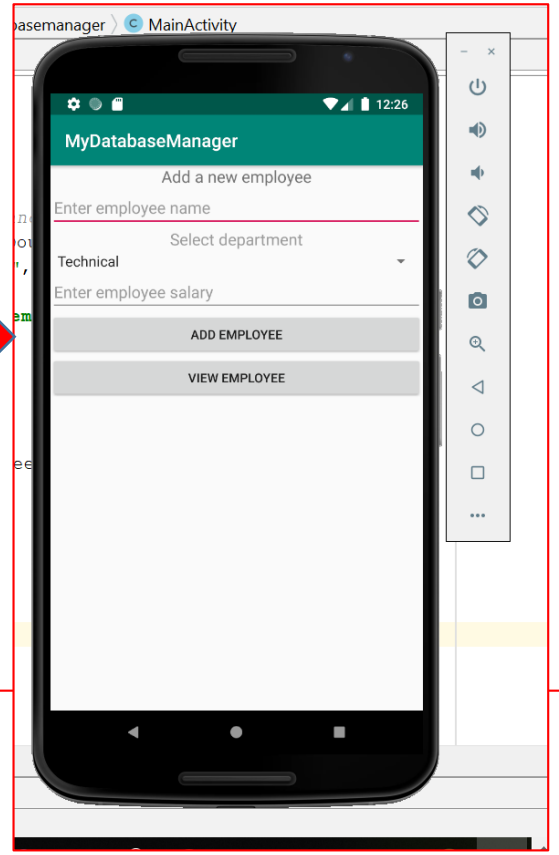
MainActivity.java

```
14
15  public class MainActivity extends AppCompatActivity  {
16  //implements View.OnClickListener
17      EditText inputName, inputSalary;
18      Spinner inputspinnerDept;
19      Button buttonadd, buttonview;
20      DatabaseManager mDatabase;
21
22      @Override
23      protected void onCreate(Bundle savedInstanceState) {
24          super.onCreate(savedInstanceState);
25          setContentView(R.layout.activity_main);
26          mDatabase = new DatabaseManager( context: this);
27          //textViewEmployees = (TextView) findViewById(R.id.textView3);
28          inputName = (EditText) findViewById(R.id.editText);  // input name
29          inputSalary = (EditText) findViewById(R.id.editText2); //input salary
30          inputspinnerDept = (Spinner) findViewById(R.id.spinnerDepartment); //spinner departi
31          buttonadd = (Button) findViewById(R.id.button);
32          buttonadd.setOnClickListener((v) → { addEmployee(); });
38          buttonview = (Button)findViewById(R.id.button4);
39          buttonview.setOnClickListener((v) → { viewdata(); });
45      }// oncreate
46
```



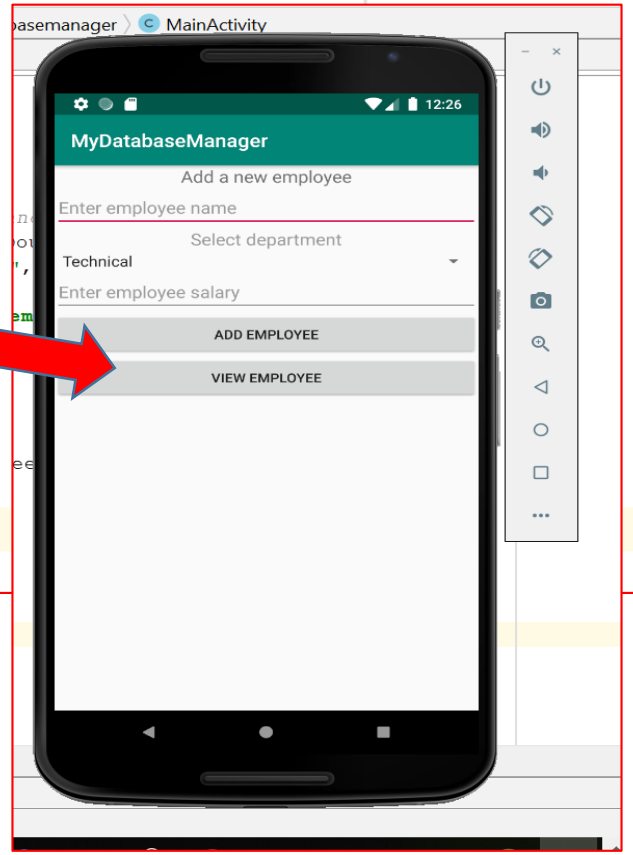
MainActivity.Java

```java
47  private void addEmployee() {
48      String name = inputName.getText().toString();
49      String dept = inputspinnerDept.getSelectedItem().toString();
50      Calendar cal = Calendar.getInstance();
51      SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy-mm-dd hh:mm:ss");
52      String joiningDate = sdf.format(cal.getTime());
53      String salary = inputSalary.getText().toString();
54      //validation
55      if (name.isEmpty()) {
56          inputName.setError("Name can't be empty");
57          inputName.requestFocus();
58          return;
59      }
60      if (salary.isEmpty()) {
61          inputSalary.setError("Salary can't be empty");
62          inputSalary.requestFocus();
63          return;
64      }
```

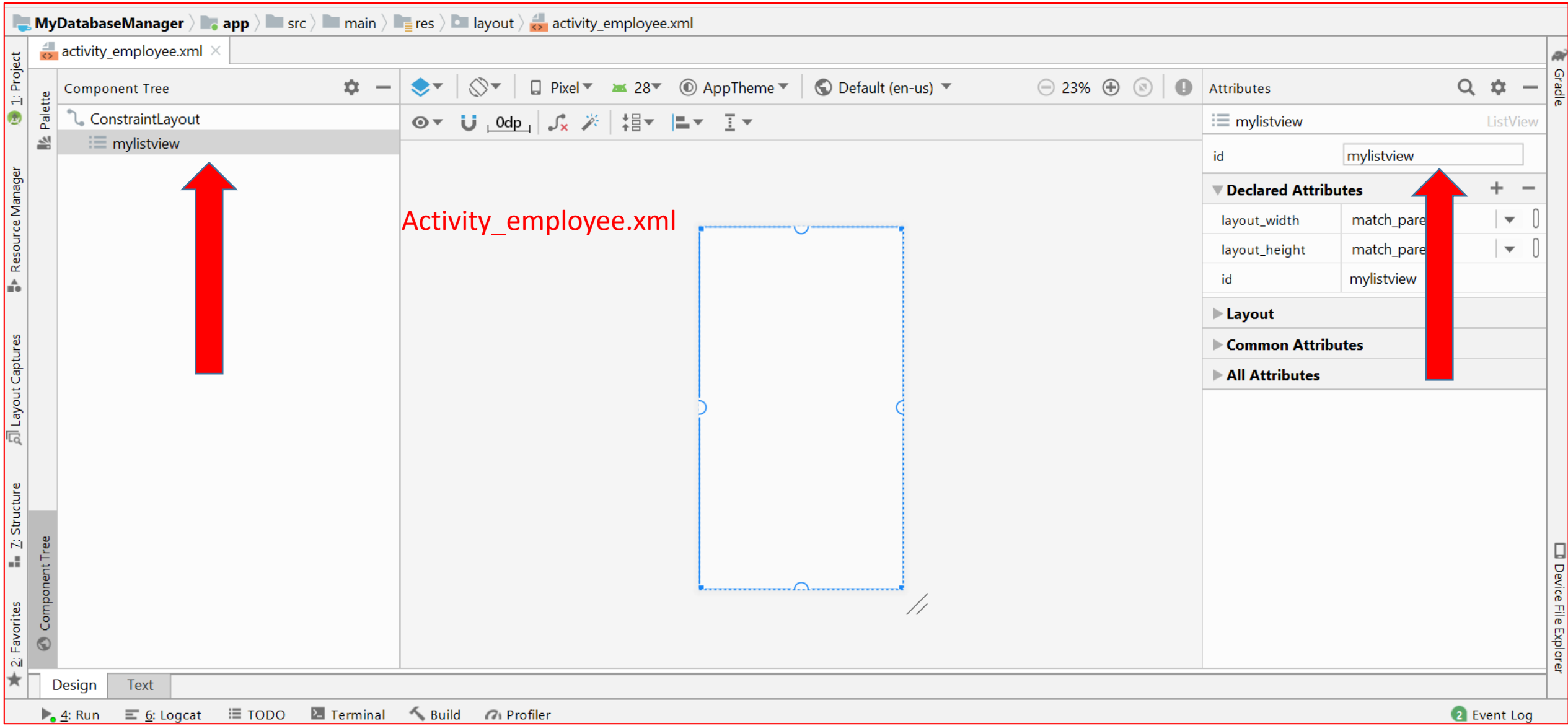


MainActivity.Java

```
65
66          //adding the employee with the DatabaseManager instance
67          if (mDatabase.addEmployee(name, dept, joiningDate, Double.parseDouble(salary)))
68              Toast.makeText( context: this,  text: "Employee Added", Toast.LENGTH_SHORT).show();
69          else
70              Toast.makeText( context: this,  text: "Could not add employee", Toast.LENGTH_SHORT).show();
71
72      }
73
74      public void viewdata()
75      {
76          Intent intent = new Intent( packageContext: this, EmployeeActivity.class);
77          startActivity(intent);
78      }
79
80
81  }
82
83
```



Go to res, layout, right click, new, activity, empty activity, name it Activity_employee



This activity_employee will auto-generate a Java file, namely EmployeeActivity.java

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

app | Nexus 6 API 27

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 EmployeeActivity

activity_employee.xml | EmployeeActivity.java

1: Project | Resource Manager | Layout Captures

```java
package com.example.mydatabasemanager;

import ...

public class EmployeeActivity extends AppCompatActivity {

    List<Employee> employeeList;
    ListView listView;
    DatabaseManager mDatabase;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_employee);
        mDatabase = new DatabaseManager( context: this);
        employeeList = new ArrayList<>();
        listView = (ListView) findViewById(R.id.mylistview);
        loadEmployeesFromDatabase();
    }
```

EmployeeActivity.java

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

app | Nexus 6 API 27

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 EmployeeActivity

activity_employee.xml | EmployeeActivity.java

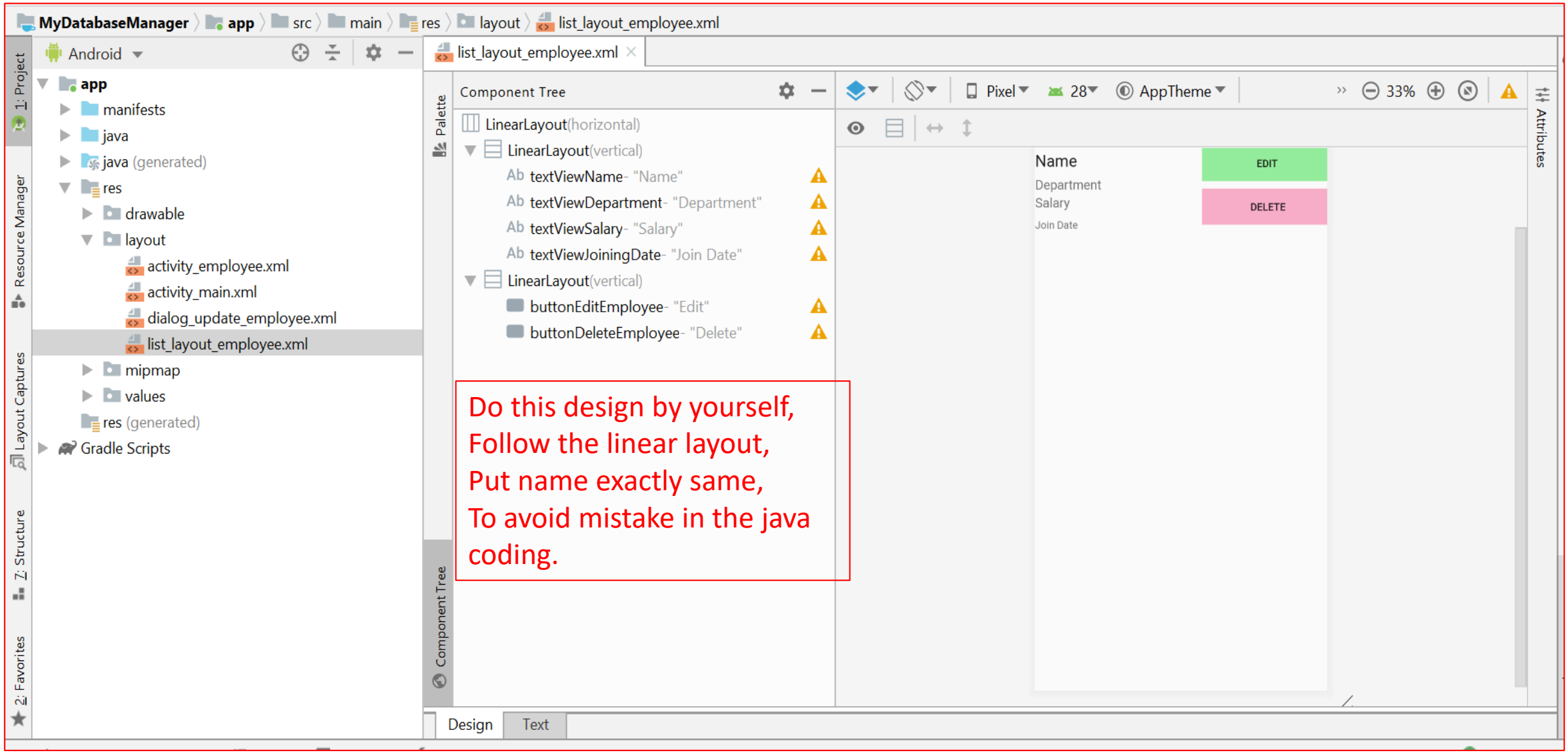
```
26      }
27
28      private void loadEmployeesFromDatabase() {
29          //we are here using the DatabaseManager instance to get all employees
30          Cursor cursor = mDatabase.getAllEmployees();
31
32          if (cursor.moveToFirst()) {
33              do {
34                  employeeList.add(new Employee(
35                          cursor.getInt( columnIndex: 0),
36                          cursor.getString( columnIndex: 1),
37                          cursor.getString( columnIndex: 2),
38                          cursor.getString( columnIndex: 3),
39                          cursor.getDouble( columnIndex: 4)
40                  ));
41              } while (cursor.moveToNext());
42
43              //passing the database manager instance this time to the adapter
44              EmployeeAdapter adapter = new EmployeeAdapter( mCtx: this, R.layout.list_layout_employee, employeeList, mDatabase);
45              listView.setAdapter(adapter);
46          }
47      }
48  }
49
```

EmployeeActivity.java

OK, by this time, we have done 2 XML activities with the associated 2 Java files

Now, we need to create : 3 standalone Java files and 2 standalone XML layout files

Go to res, layout, right click, new, XML, Layout XML file, put name: list_layout_employee



Do this design by yourself,
Follow the linear layout,
Put name exactly same,
To avoid mistake in the java coding.

Go to res, layout, right click, new, XML, Layout XML file, put name: dialog_update_employee

MyDatabaseManager 〉 app 〉 src 〉 main 〉 res 〉 layout 〉 dialog_update_employee.xml

Android

- app
  - manifests
  - java
  - java (generated)
  - res
    - drawable
    - layout
      - activity_employee.xml
      - activity_main.xml
      - dialog_update_employee.xml
      - list_layout_employee.xml
    - mipmap
    - values
  - res (generated)
- Gradle Scripts

dialog_update_employee.xml

Component Tree

- LinearLayout(vertical)
  - TextView- "Edit Employee"
  - editTextName
  - TextView- "Select Department"
  - spinnerDepartment
  - editTextSalary(Number)
  - buttonUpdateEmployee- "Update"

Pixel 28 AppTheme

Edit Employee

Enter Employee Name

Select Department

Technical

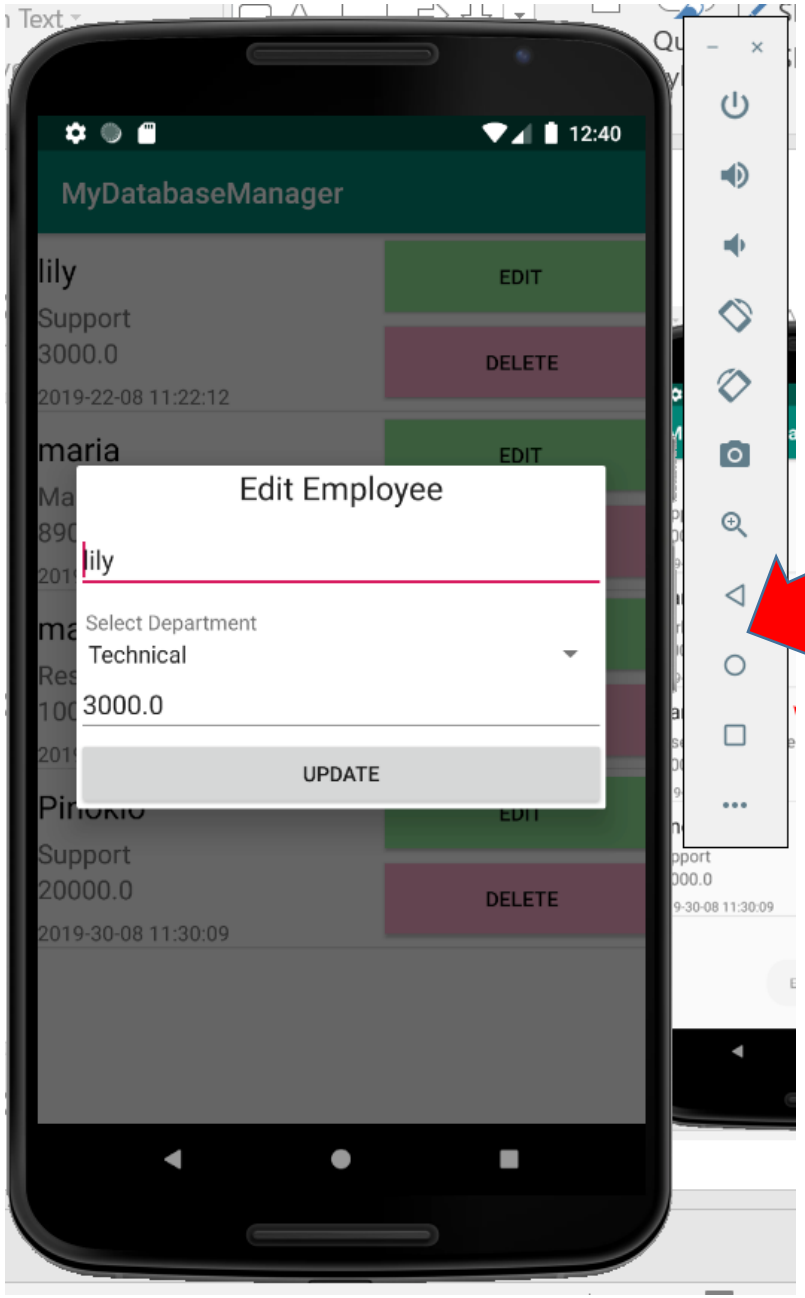
Enter Employee Salary

UPDATE

This design is almost same with activity_main.xml
The difference is the top TextView, it is written “Edit Employee” in here.
And Button is written Update in here.

So, you can reuse the previous code, just change the textview into “Edit Employee” and button is “Update”

Design Text



Previous codes inside dialog_update_employee Will give you this interface for Edit Employee

This is dialog.

OK, by now, we have done ALL 4 XML files with 2 Java files.

Next, we need to do 3 stand-alone Java files.

Go to java folder, com.example folder, right click, new Java class, put name Employee

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 Employee

Android

- app
  - manifests
  - java
    - com.example.mydatabasemanager
      - DatabaseManager
      - Employee
      - EmployeeActivity
      - EmployeeAdapter
      - MainActivity
    - com.example.mydatabasemanager (androidTe
    - com.example.mydatabasemanager (test)
  - java (generated)
  - res
  - res (generated)
- Gradle Scripts

Employee.java

```java
package com.example.mydatabasemanager;

public class Employee {
    int id;
    String name, dept, joiningDate;
    double salary;

    public Employee(int id, String name, String dept, String joiningDate, double salary) {
        this.id = id;
        this.name = name;
        this.dept = dept;
        this.joiningDate = joiningDate;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }
```

Employee.Java

Type this code by yourself.

- app
  - manifests
  - java
    - com.example.mydatabasemanager
      - DatabaseManager
      - Employee
      - EmployeeActivity
      - EmployeeAdapter
      - MainActivity
    - com.example.mydatabasemanager (androidTe
    - com.example.mydatabasemanager (test)
  - java (generated)
  - res
  - res (generated)
- Gradle Scripts

```
15
16      public int getId() {
17          return id;
18      }
19
20      public String getName() {
21          return name;
22      }
23
24      public String getDept() {
25          return dept;
26      }
27
28      public String getJoiningDate() {
29          return joiningDate;
30      }
31
32      public double getSalary() {
33          return salary;
34      }
35  }
36
```

Employee.Java

Complete your code until finish.

Go to java folder, com.example folder, right click, new Java class, put name DatabaseManager

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 DatabaseManager

Android

- app
  - manifests
  - java
    - com.example.mydatabasemanager
      - DatabaseManager
      - Employee
      - EmployeeActivity
      - EmployeeAdapter
      - MainActivity
    - com.example.mydatabasemanager (androidTe
    - com.example.mydatabasemanager (test)
  - java (generated)
  - res
  - res (generated)
- Gradle Scripts

DatabaseManager.java

```java
package com.example.mydatabasemanager;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseManager extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "EmployeesDatabase";
    private static final int DATABASE_VERSION = 3;
    private static final String TABLE_NAME = "employees";
    private static final String COLUMN_ID = "id";
    private static final String COLUMN_NAME = "name";
    private static final String COLUMN_DEPT = "department";
    private static final String COLUMN_JOIN_DATE = "joiningDate";
    private static final String COLUMN_SALARY = "salary";

    DatabaseManager(Context context) {
        super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);

    }
```

DatabaseManager.Java

Type this code by yourself.

Complete your code until finish.

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 DatabaseManager

DatabaseManager.java

DatabaseManager.Java

```
25      @Override
26      public void onCreate(SQLiteDatabase sqLiteDatabase) {
27
28          String sql = "CREATE TABLE " + TABLE_NAME + " (\n" +
29                  "     " + COLUMN_ID + " INTEGER NOT NULL CONSTRAINT employees_pk PRIMARY KEY AUTOINCREMENT ,\n" +
30                  "     " + COLUMN_NAME + " varchar(200) NOT NULL,\n" +
31                  "     " + COLUMN_DEPT + " varchar(200) NOT NULL,\n" +
32                  "     " + COLUMN_JOIN_DATE + " datetime NOT NULL,\n" +
33                  "     " + COLUMN_SALARY + " double NOT NULL\n" +
34                  ");";
35
36          // Executing the string to create the table
37          sqLiteDatabase.execSQL(sql);
38      }
39
40      @Override
41          public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
42          String sql = "DROP TABLE IF EXISTS " + TABLE_NAME + ";";
43          sqLiteDatabase.execSQL(sql);
44          onCreate(sqLiteDatabase);
45      }
46
47      // operations
48
```

Complete your code until finish.

DatabaseManager.Java

```
46
47      // operations
48
49      boolean addEmployee(String name, String dept, String joiningdate, double salary) {
50          ContentValues contentValues = new ContentValues();
51          contentValues.put(COLUMN_NAME, name);
52          contentValues.put(COLUMN_DEPT, dept);
53          contentValues.put(COLUMN_JOIN_DATE, joiningdate);
54          contentValues.put(COLUMN_SALARY, salary);
55          SQLiteDatabase db = getWritableDatabase();
56          return db.insert(TABLE_NAME, nullColumnHack: null, contentValues) != -1;
57      }
58
59      Cursor getAllEmployees() {
60          SQLiteDatabase db = getReadableDatabase();
61          return db.rawQuery( sql: "SELECT * FROM " + TABLE_NAME, selectionArgs: null);
62      }
63
```

Complete your code until finish.

DatabaseManager.Java

```java
    Cursor getAllEmployees() {
        SQLiteDatabase db = getReadableDatabase();
        return db.rawQuery( sql: "SELECT * FROM " + TABLE_NAME,  selectionArgs: null);
    }

    boolean updateEmployee(int id, String name, String dept, double salary) {
        SQLiteDatabase db = getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COLUMN_NAME, name);
        contentValues.put(COLUMN_DEPT, dept);
        contentValues.put(COLUMN_SALARY, salary);
        return db.update(TABLE_NAME, contentValues,  whereClause: COLUMN_ID + "=?", new String[]{String.valueOf(id)}) == 1;
    }

    boolean deleteEmployee(int id) {
        SQLiteDatabase db = getWritableDatabase();
        return db.delete(TABLE_NAME,  whereClause: COLUMN_ID + "=?", new String[]{String.valueOf(id)}) == 1;
    }
}
```

Go to java folder, com.example folder, right click, new Java class, put name EmployeeAdapter

app | Nexus 6 API 27

MyDatabaseManager 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 mydatabasemanager 〉 EmployeeAdapter

Android

- app
  - manifests
  - java
    - com.example.mydatabasemanager
      - DatabaseManager
      - Employee
      - EmployeeActivity
      - EmployeeAdapter
      - MainActivity
    - com.example.mydatabasemanager (androidTe
    - com.example.mydatabasemanager (test)
  - java (generated)
  - res
  - res (generated)
- Gradle Scripts

EmployeeAdapter.java

```java
package com.example.mydatabasemanager;

import ...

public class EmployeeAdapter extends ArrayAdapter<Employee> {

    Context mCtx;
    int layoutRes;
    List<Employee> employeeList;
    DatabaseManager mDatabase;

    //modified the constructor and we are taking the DatabaseManager instance here
    public EmployeeAdapter(Context mCtx, int layoutRes, List<Employee> employeeList,
                           DatabaseManager mDatabase) {
        super(mCtx, layoutRes, employeeList);
        this.mCtx = mCtx;
        this.layoutRes = layoutRes;
        this.employeeList = employeeList;
        this.mDatabase = mDatabase;
    }
```

EmployeeAdapter.Java

Type this code by yourself.

Continue your code until finish

EmployeeAdapter.Java

Android

- app
  - manifests
  - java
    - com.example.mydatabasemanager
      - DatabaseManager
      - Employee
      - EmployeeActivity
      - EmployeeAdapter
      - MainActivity
    - com.example.mydatabasemanager (androidTe
    - com.example.mydatabasemanager (test)
  - java (generated)
  - res
  - res (generated)
- Gradle Scripts

EmployeeAdapter.java

```java
39      @Override
40      public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
41          LayoutInflater inflater = LayoutInflater.from(mCtx);
42
43          View view = inflater.inflate(layoutRes,  root: null);
44          // from list_layout_employee
45          TextView textViewName = view.findViewById(R.id.textViewName);
46          TextView textViewDept = view.findViewById(R.id.textViewDepartment);
47          TextView textViewSalary = view.findViewById(R.id.textViewSalary);
48          TextView textViewJoinDate = view.findViewById(R.id.textViewJoiningDate);
49          final Employee employee = employeeList.get(position);
50
51          textViewName.setText(employee.getName());
52          textViewDept.setText(employee.getDept());
53          textViewSalary.setText(String.valueOf(employee.getSalary()));
54          textViewJoinDate.setText(employee.getJoiningDate());
55
56          view.findViewById(R.id.buttonDeleteEmployee).setOnClickListener((view) → {
59                  deleteEmployee(employee);
60          });
62
63          view.findViewById(R.id.buttonEditEmployee).setOnClickListener((view) → {
66                  updateAnEmployee(employee);
67          });
69          return view;
70      }//getView
71
```

Continue your code until finish

EmployeeAdapter.Java

EmployeeAdapter.java

```
72  @   private void updateAnEmployee(final Employee employee) {
73          AlertDialog.Builder builder = new AlertDialog.Builder(mCtx);
74          LayoutInflater inflater = LayoutInflater.from(mCtx);
75          View view = inflater.inflate(R.layout.dialog_update_employee,  root: null);
76          builder.setView(view);
77
78          final AlertDialog alertDialog = builder.create();
79          alertDialog.show();
80          //from dialog_update_employee.xml
81          final EditText editTextName = view.findViewById(R.id.editTextName);
82          final EditText editTextSalary = view.findViewById(R.id.editTextSalary);
83          final Spinner spinner = view.findViewById(R.id.spinnerDepartment);
84
85          editTextName.setText(employee.getName());
86          editTextSalary.setText(String.valueOf(employee.getSalary()));
87
88
```

Continue your code until finish

EmployeeAdapter.Java

```java
87
88          view.findViewById(R.id.buttonUpdateEmployee).setOnClickListener((view) → {
91                  String name = editTextName.getText().toString();
92                  String salary = editTextSalary.getText().toString();
93                  String dept = spinner.getSelectedItem().toString();
94
95                  if (name.isEmpty()) {
96                      editTextName.setError("Name can't be empty");
97                      editTextName.requestFocus();
98                      return;
99                  }
100
101                 if (salary.isEmpty()) {
102                     editTextSalary.setError("Salary can't be empty");
103                     editTextSalary.requestFocus();
104                     return;
105                 }
106                 //calling the update method from database manager instance
107                 if (mDatabase.updateEmployee(employee.getId(), name, dept, Double.valueOf(salary))) {
108                     Toast.makeText(mCtx, text: "Employee Updated", Toast.LENGTH_SHORT).show();
109                     loadEmployeesFromDatabaseAgain();
110                 }
111                 alertDialog.dismiss();
112         });
114     }// update employee
115
```

Continue your code until finish

EmployeeAdapter.Java

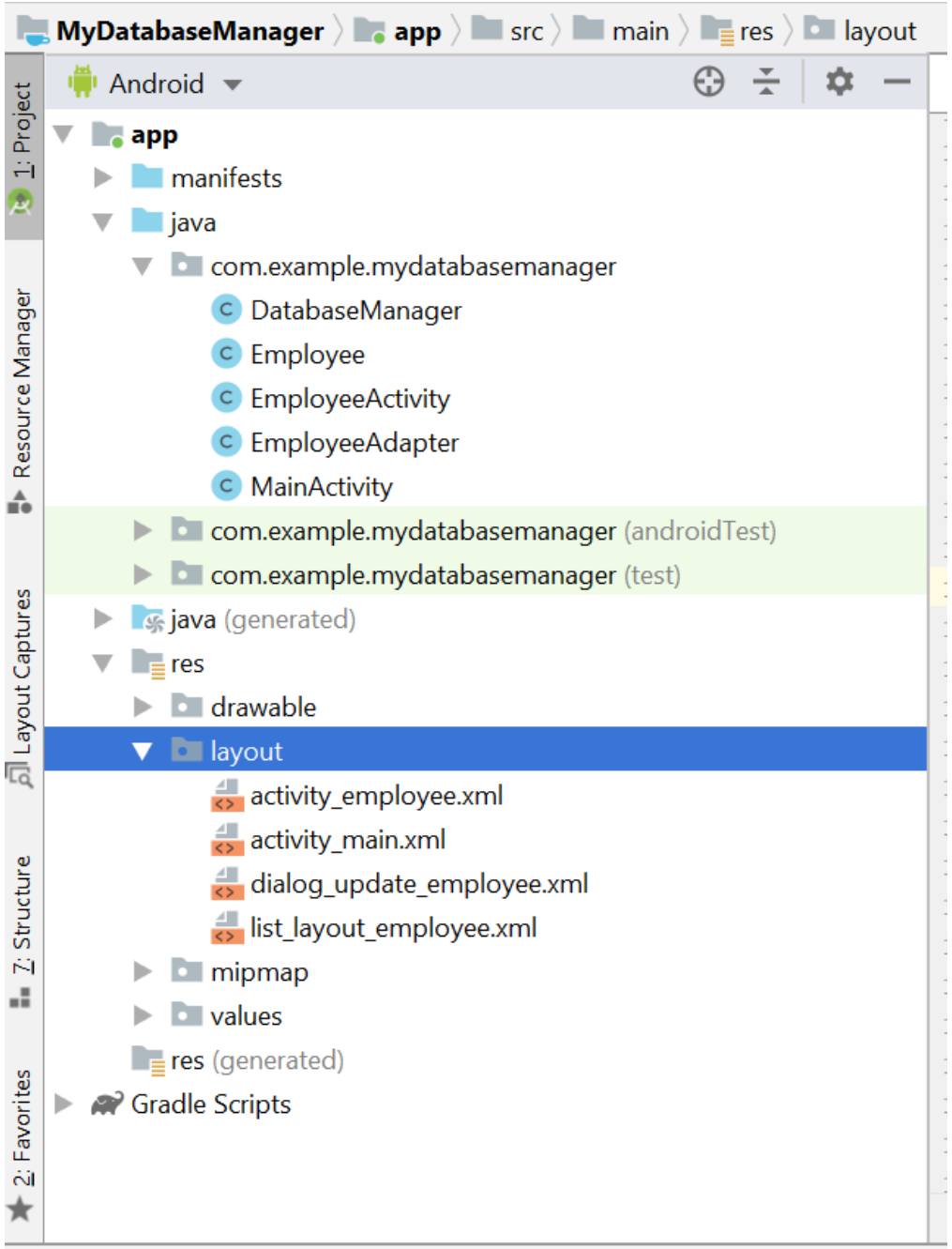
EmployeeAdapter.java

```java
    private void deleteEmployee(final Employee employee) {
        AlertDialog.Builder builder = new AlertDialog.Builder(mCtx);
        builder.setTitle("Are you sure?");

        builder.setPositiveButton( text: "Yes", (dialogInterface, i) → {

                //calling the delete method from the database manager instance
                if (mDatabase.deleteEmployee(employee.getId()))
                    loadEmployeesFromDatabaseAgain();
        });

        builder.setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {

            }
        });

        AlertDialog alertDialog = builder.create();
        alertDialog.show();
    }
```

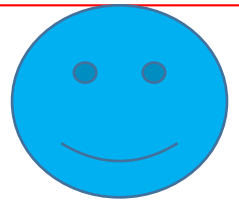
Continue your code until finish

EmployeeAdapter.Java

```
141
142     private void loadEmployeesFromDatabaseAgain() {
143         //calling the read method from database instance
144         Cursor cursor = mDatabase.getAllEmployees();
145
146         employeeList.clear();
147         if (cursor.moveToFirst()) {
148             do {
149                 employeeList.add(new Employee(
150                         cursor.getInt( columnIndex: 0),
151                         cursor.getString( columnIndex: 1),
152                         cursor.getString( columnIndex: 2),
153                         cursor.getString( columnIndex: 3),
154                         cursor.getDouble( columnIndex: 4)
155                 ));
156             } while (cursor.moveToNext());
157         }
158         notifyDataSetChanged();
159     } //load employees
160 } //employee adapter
161
```



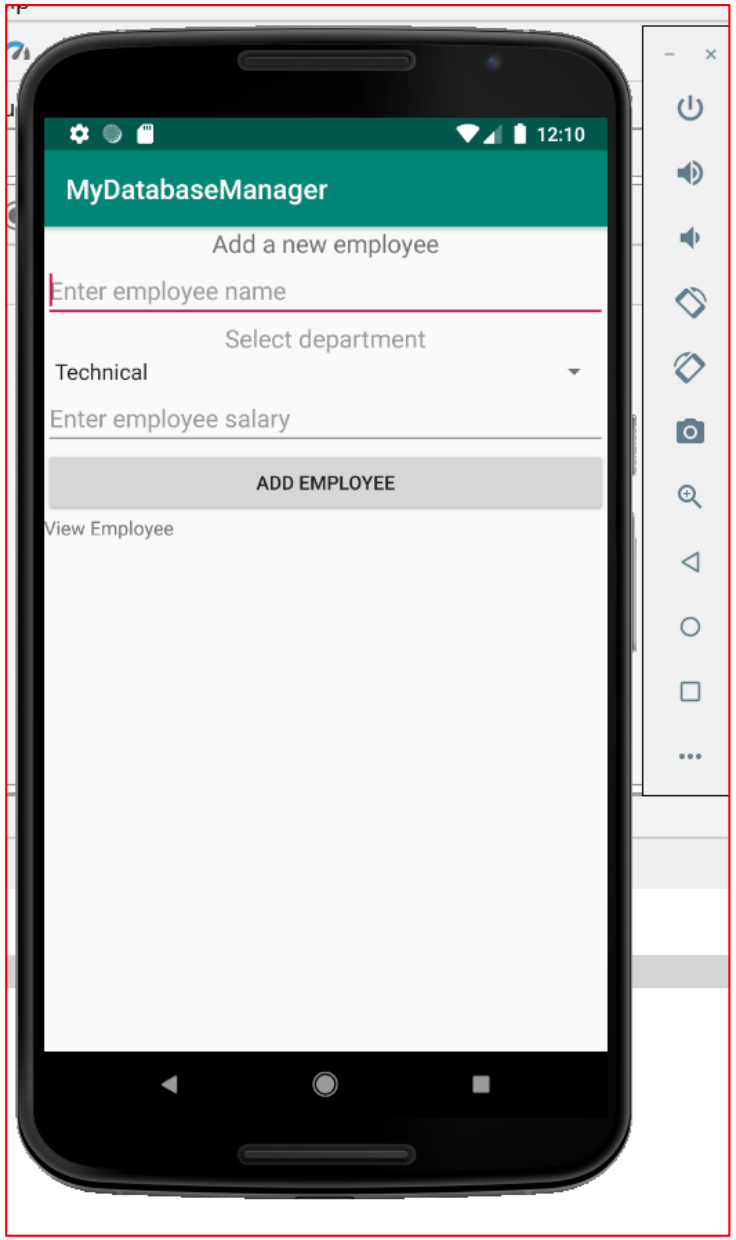
Allright,

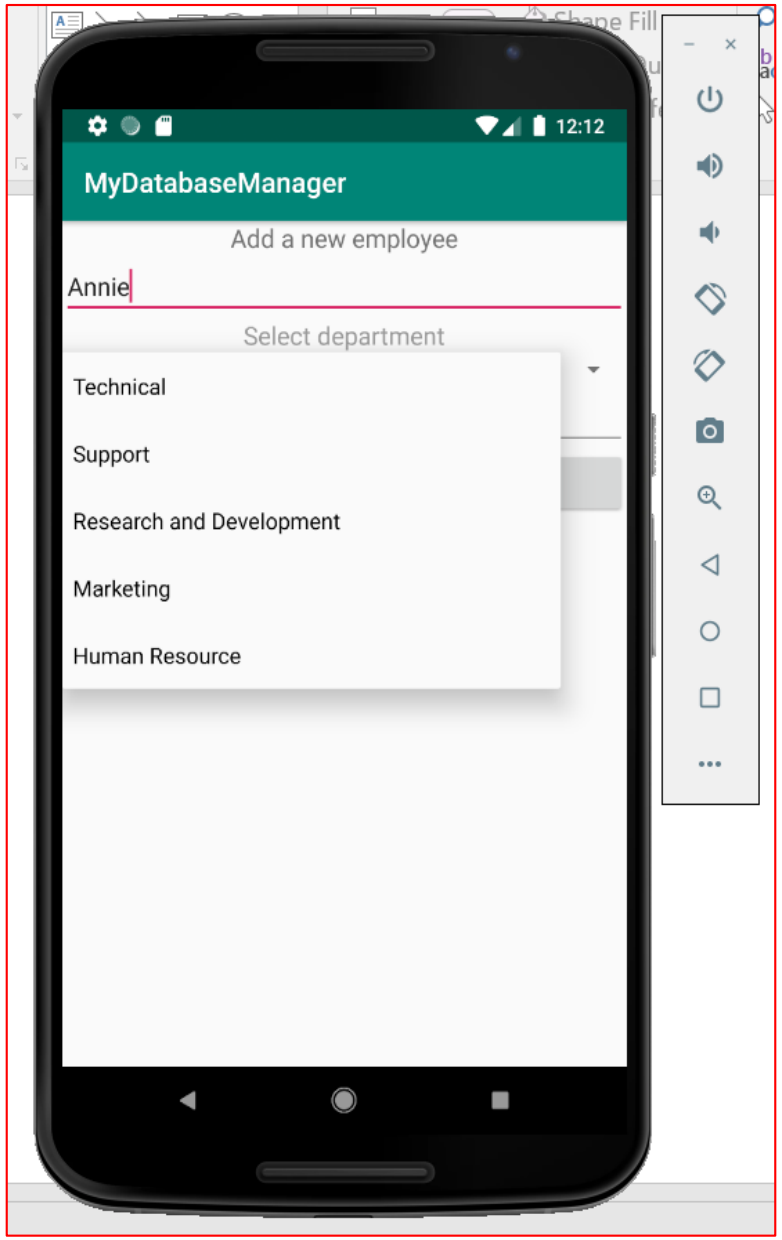


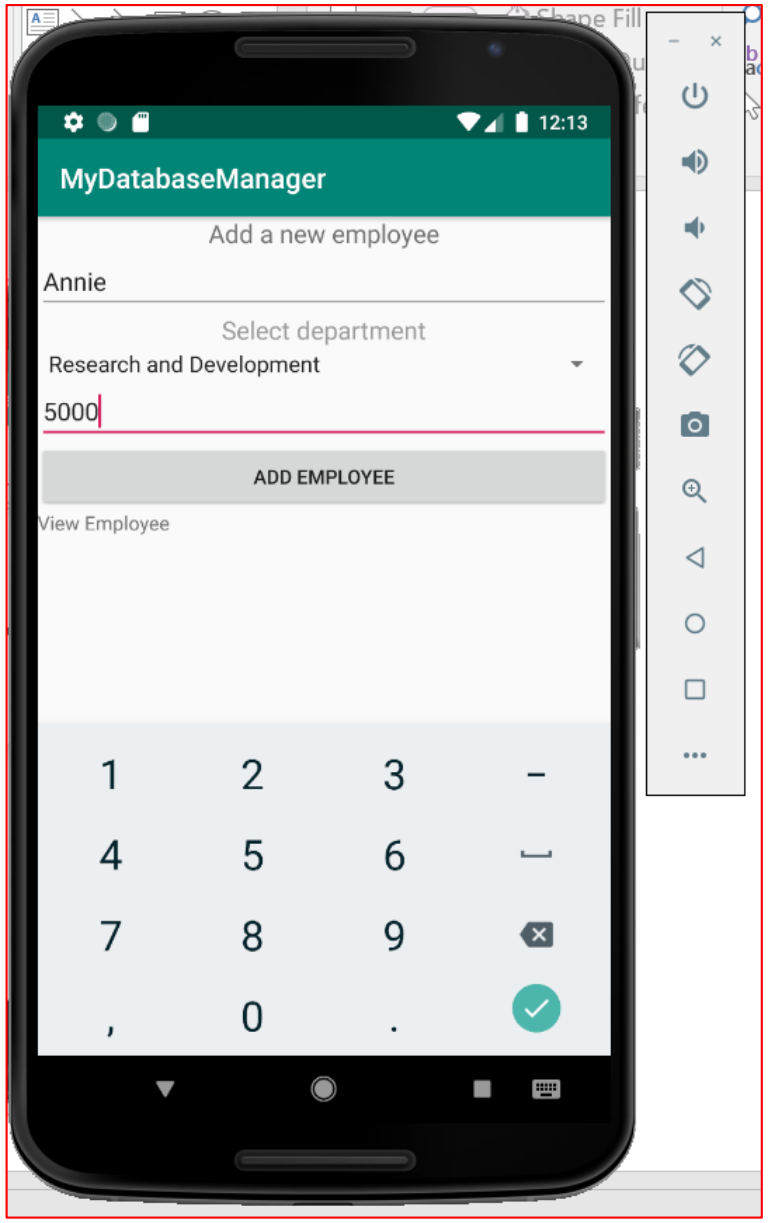
we have done all 9 files
Please check again, have you done all?

Emulator: Nexus 6, API 27 (bigger version)

Add Data

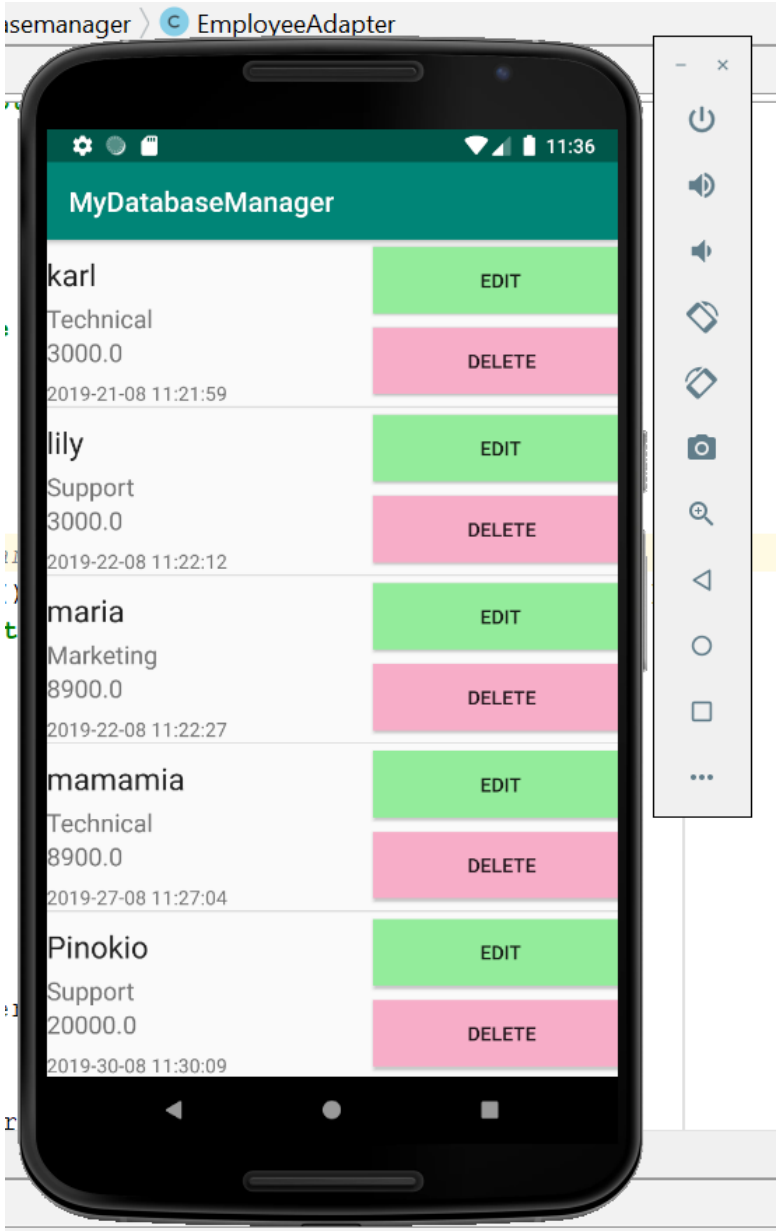




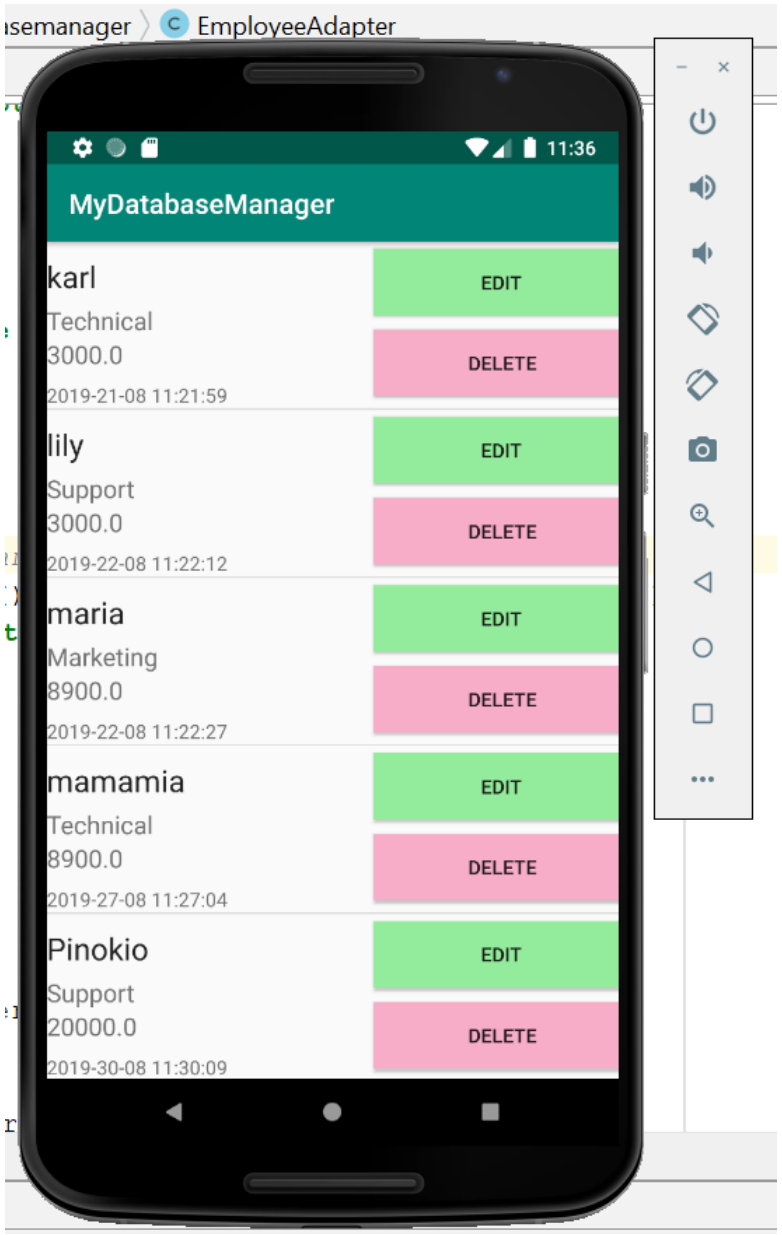


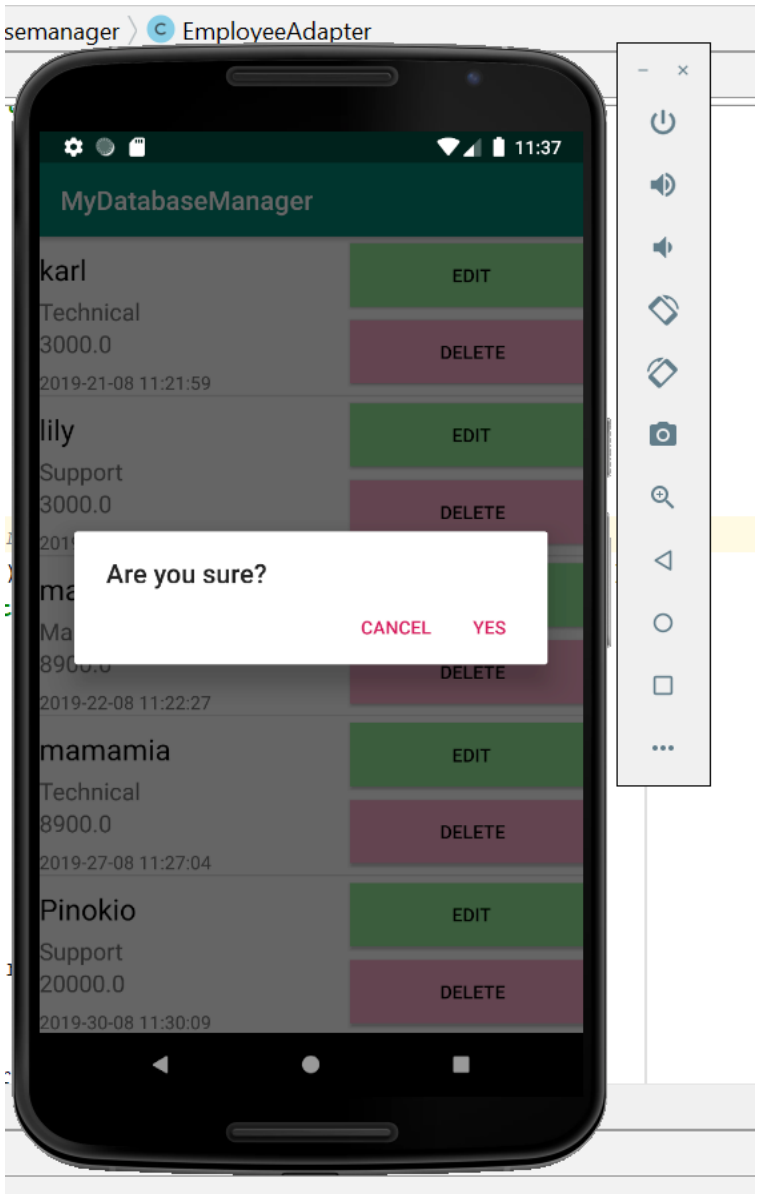
View Data

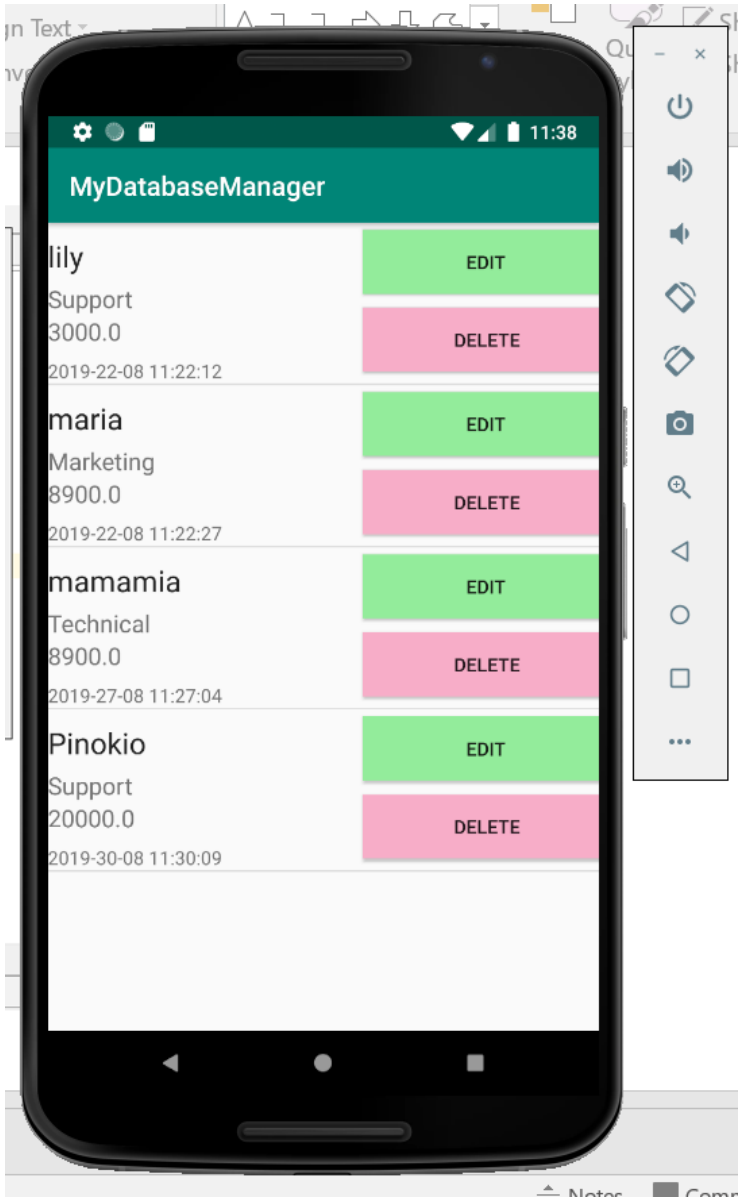
We can see ListView is shown



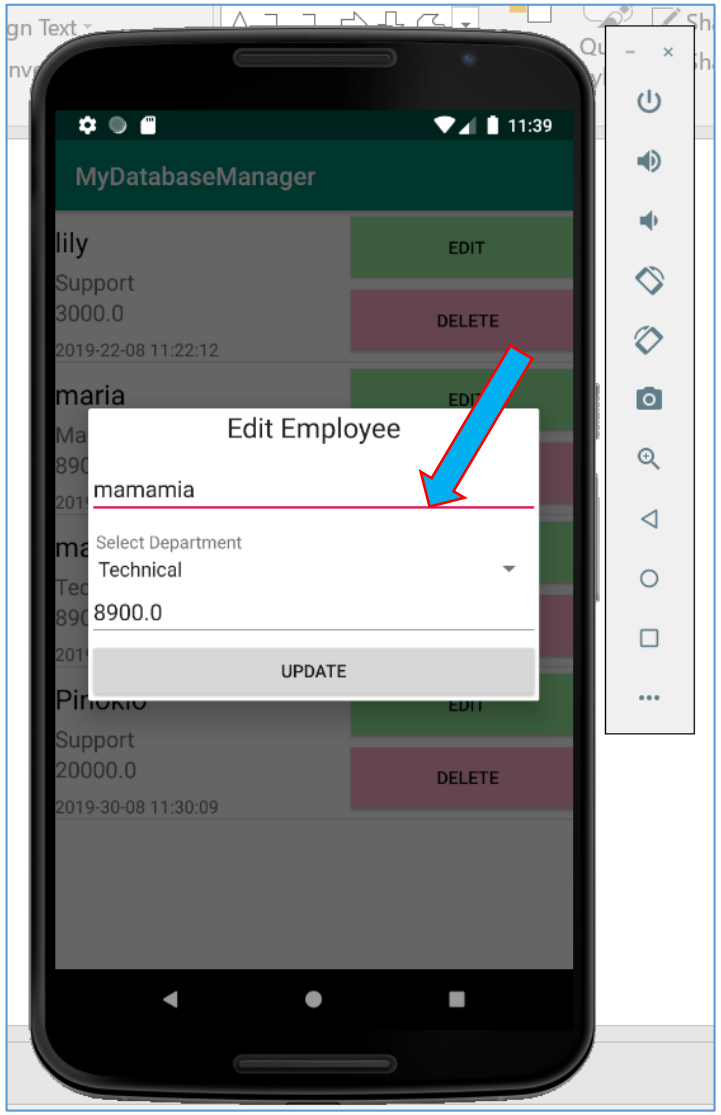
Delete Karl

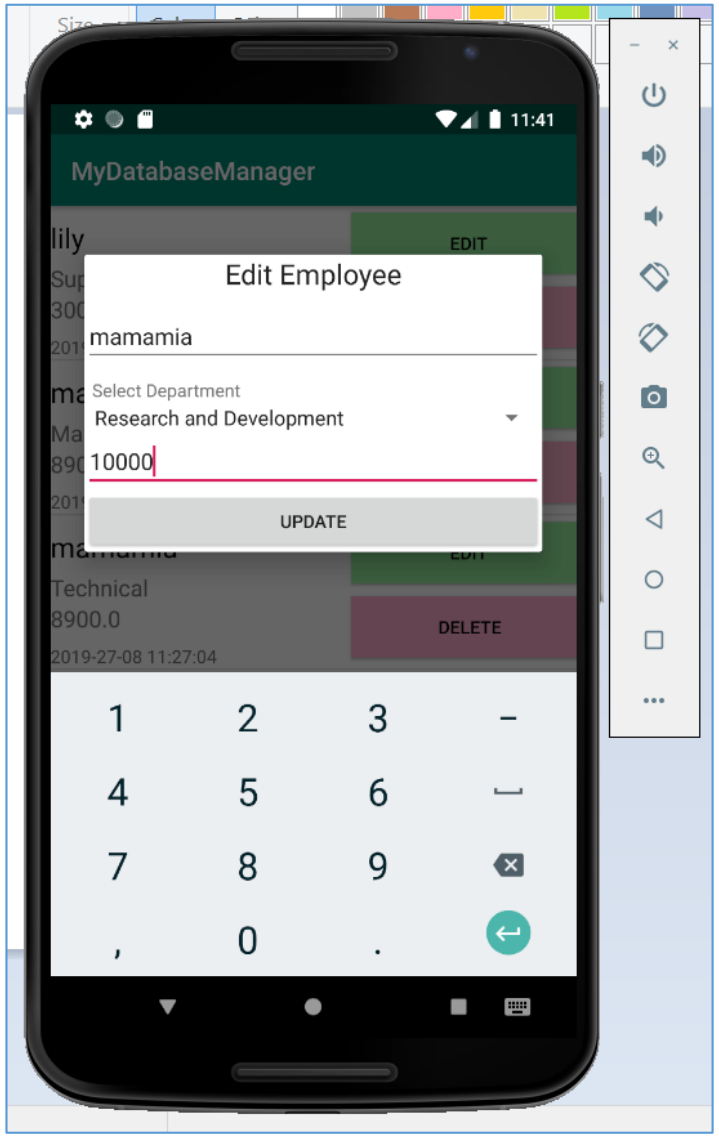

semanager
EmployeeAdapter
11:36
MyDatabaseManager
karl
Technical
3000.0
2019-21-08 11:21:59
EDIT
DELETE
lily
Support
3000.0
2019-22-08 11:22:12
EDIT
DELETE
maria
Marketing
8900.0
2019-22-08 11:22:27
EDIT
DELETE
mamamia
Technical
8900.0
2019-27-08 11:27:04
EDIT
DELETE
Pinokio
Support
20000.0
2019-30-08 11:30:09
EDIT
DELETE

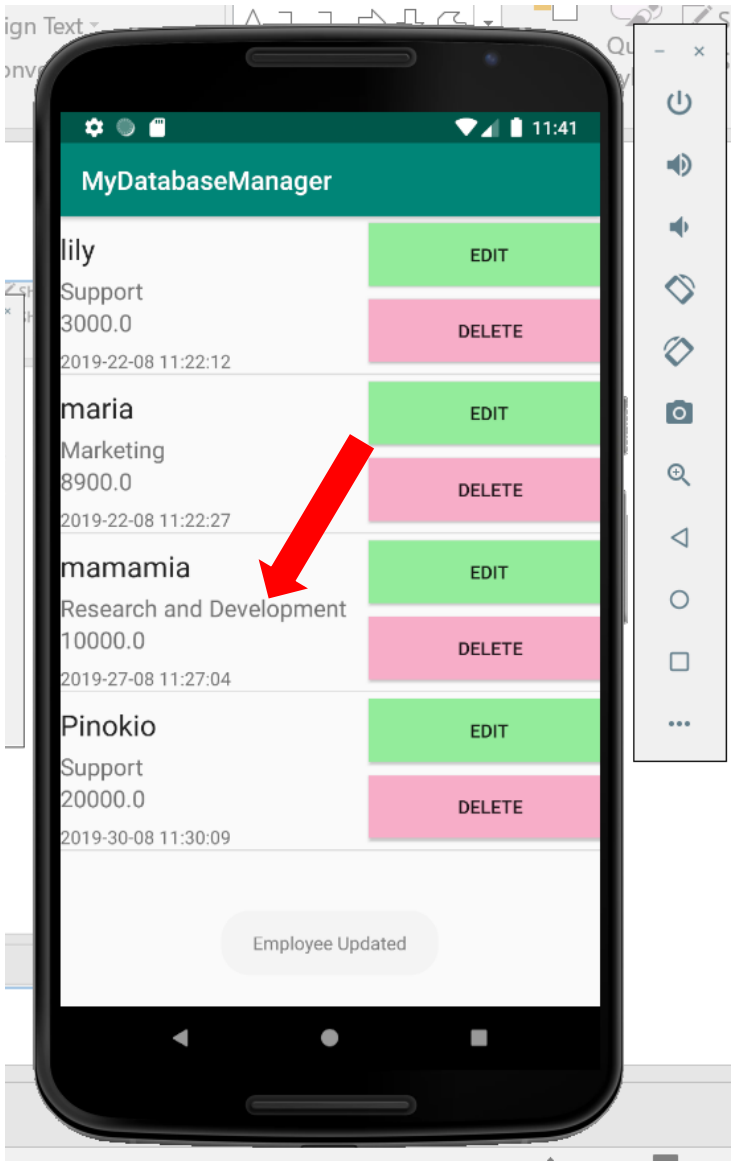

semanager
EmployeeAdapter
11:37
MyDatabaseManager
karl
Technical
3000.0
2019-21-08 11:21:59
EDIT
DELETE
lily
Support
3000.0
EDIT
DELETE
Are you sure?
CANCEL
YES
2019-22-08 11:22:27
DELETE
mamamia
Technical
8900.0
2019-27-08 11:27:04
EDIT
DELETE
Pinokio
Support
20000.0
2019-30-08 11:30:09
EDIT
DELETE


11:38
MyDatabaseManager
lily
Support
3000.0
2019-22-08 11:22:12
EDIT
DELETE
maria
Marketing
8900.0
2019-22-08 11:22:27
EDIT
DELETE
mamamia
Technical
8900.0
2019-27-08 11:27:04
EDIT
DELETE
Pinokio
Support
20000.0
2019-30-08 11:30:09
EDIT
DELETE

Edit Mamamia







The End