SD-6501 Assignment 2

Jing Li ID:21902204

Table of Contents

Introuction :

In Assignment 2 , I will add two main functions. One main function is the Database implementation and management, I attach video to show steps.

I upload a demo video to display the whole progress like below list:

Create: After Clicking "Save" button in the ListActicity to create Database.

Read: In the mainActivity ,after clicking Options Menu, and click AdminAccount, It will jump to one AdminActivity to display a SQL data .

Update: In AdminActivity, after input then click "Update" button.

Delete: In AdminActivity, after input" IDNum" ,click "Delete" button.
Function 2 is one improvement about the content display.

Conceptual Framework:

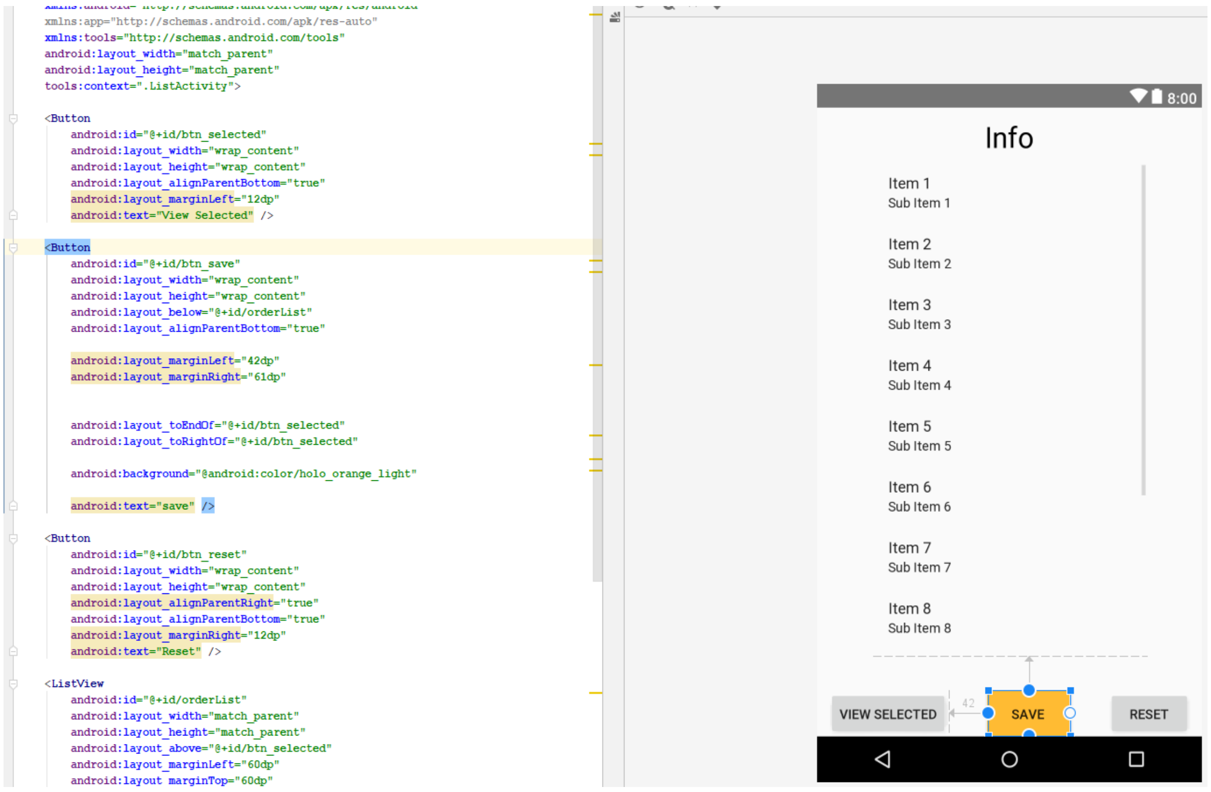
I will achieve the input information into a SQL with a DbHandler Class. Following the logic , I will create ListActicity,AdminActivity.

Advanced Feature :Database implementation and management:

Create:

1.Layout:

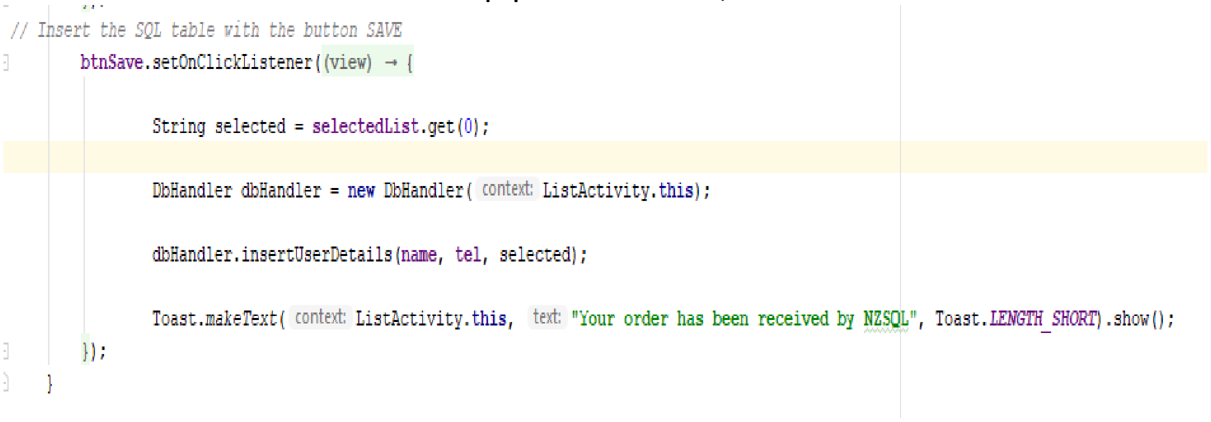
In the ListActicity ,at the base of assignment 1 including ViewSelected and Reset functions, I will add a Button "btn_save" to create the SQL with the selectedList in the _list.xml file.



```
    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"
    tools:context=".ListActivity">

    <Button
        android:id="@+id/btn_selected"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginLeft="12dp"
        android:text="View Selected" />

    <Button
        android:id="@+id/btn_save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/orderList"
        android:layout_alignParentBottom="true"

        android:layout_marginLeft="42dp"
        android:layout_marginRight="61dp"


        android:layout_toEndOf="@+id/btn_selected"
        android:layout_toRightOf="@+id/btn_selected"

        android:background="@android:color/holo_orange_light"

        android:text="save" />

    <Button
        android:id="@+id/btn_reset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="12dp"
        android:text="Reset" />

    <ListView
        android:id="@+id/orderList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/btn_selected"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="60dp"
```

2. Behaviour:

Then I will add a setOnClicklistener method to the Button “btn_save" in ListActivity:I also add a “Toast” information to help the users to ensure their order . Inside the OnClicklistener method, I will insert a message to create the SQL Table. Message include 3 variables : name, tel, selected ;
The whole method like this snap picture below,



```
// Insert the SQL table with the button SAVE
        btnSave.setOnClickListener((view) → {

                String selected = selectedList.get(0);

                DbHandler dbHandler = new DbHandler( context: ListActivity.this);

                dbHandler.insertUserDetails(name, tel, selected);

                Toast.makeText( context: ListActivity.this,  text: "Your order has been received by NZSQL", Toast.LENGTH_SHORT).show();
        });
    }
```

There is a Logic whch is need to think: Where does selected value come from ?
 I will get it from selectedList.get(0) like below:

```
 String selected = selectedList.get(0);
```

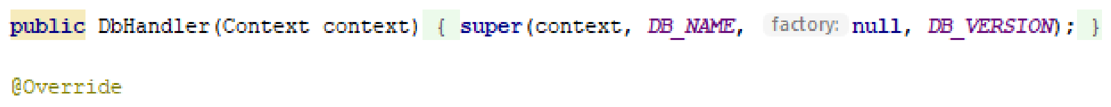
The selectedList will be set to hold only one value.
SQL is inserted by a selectedList.

I need create a method “insertUserDetails()” in a DaHandler.java Class(new built)To create the database.
Create Database-SQL:

 Create a new java class called DbHandler that I have used here
contains DbHelper class that extends SQLiteOpenHelperclass and perform all database related operations.

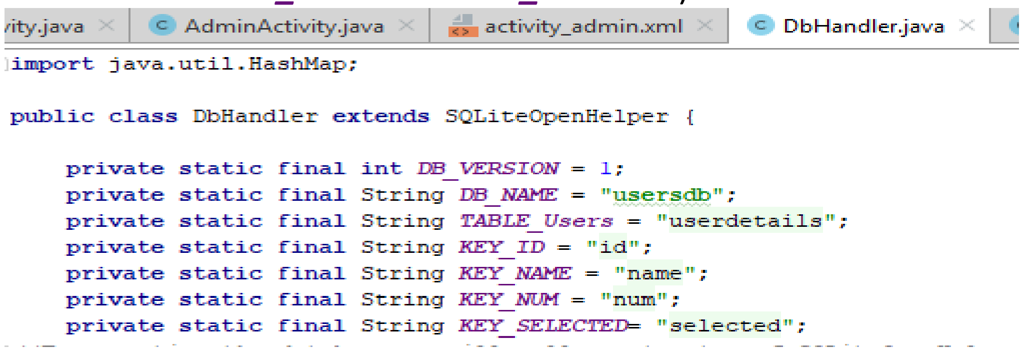
For creating the database I will call constructor of **SQLiteOpenHelper** class using **super()**.

Pass the DB_NAME and DB_VERSION in the superclass within my constructor.



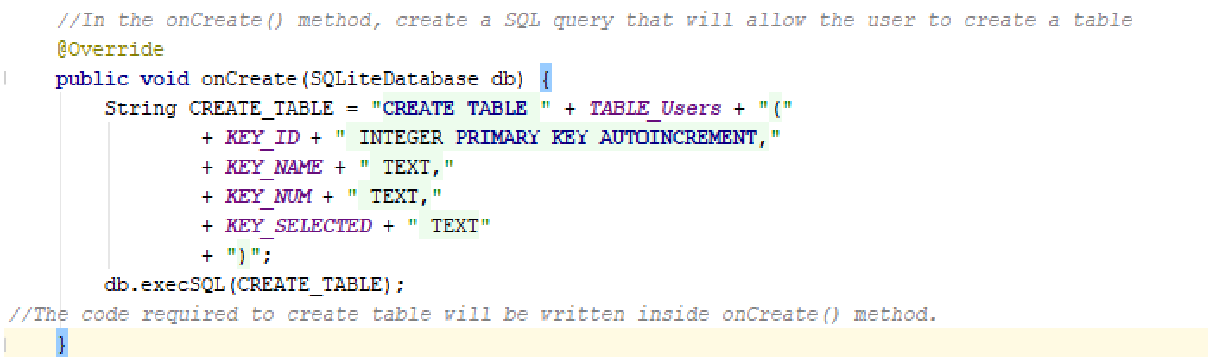
```
public DbHandler(Context context) { super(context, DB_NAME,  factory: null, DB_VERSION); }

@Override
```

Where are the ***DB_NAME*** and ***DB_VERSION***) from? I declare them at first.



```
ity.java ×   AdminActivity.java ×   activity_admin.xml ×   DbHandler.java ×

import java.util.HashMap;

public class DbHandler extends SQLiteOpenHelper {

    private static final int DB_VERSION = 1;
    private static final String DB_NAME = "usersdb";
    private static final String TABLE_Users = "userdetails";
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_NUM = "num";
    private static final String KEY_SELECTED= "selected";
```

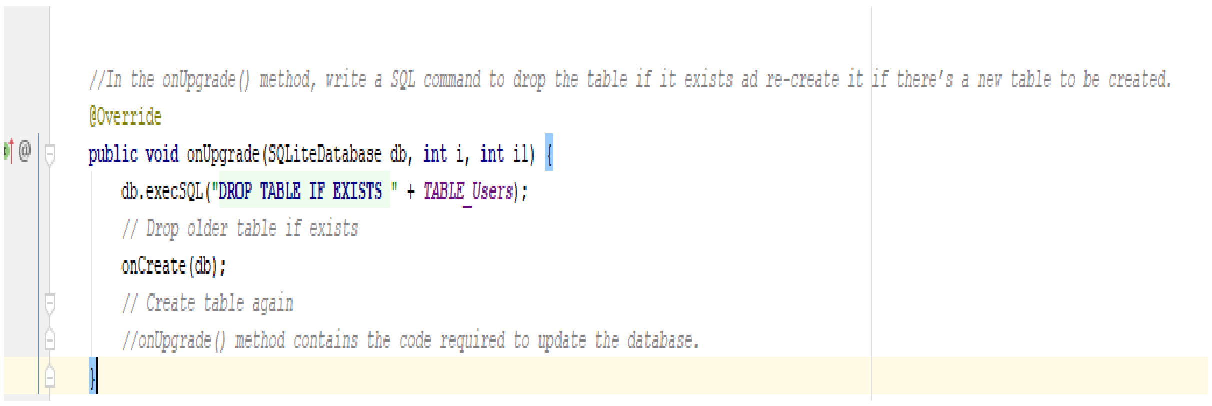
To make it simpler, modify the argument name in the onCreate() method to db. In the onCreate() method, create a SQL query that will allow the user to create a table.



```
    //In the onCreate() method, create a SQL query that will allow the user to create a table
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_TABLE = "CREATE TABLE " + TABLE_Users + "("
                + KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
                + KEY_NAME + " TEXT,"
                + KEY_NUM + " TEXT,"
                + KEY_SELECTED + " TEXT"
                + ")";
        db.execSQL(CREATE_TABLE);
//The code required to create table will be written inside onCreate() method.
    }
```

Add the unimplemented methods (onCreate() and onUpgrade()) and a constructor. Configure the constructor to only accept Context argument.

In the onUpgrade() method, write a SQL command to drop the table if it exists ad re-create it if there’s a new table to be created.

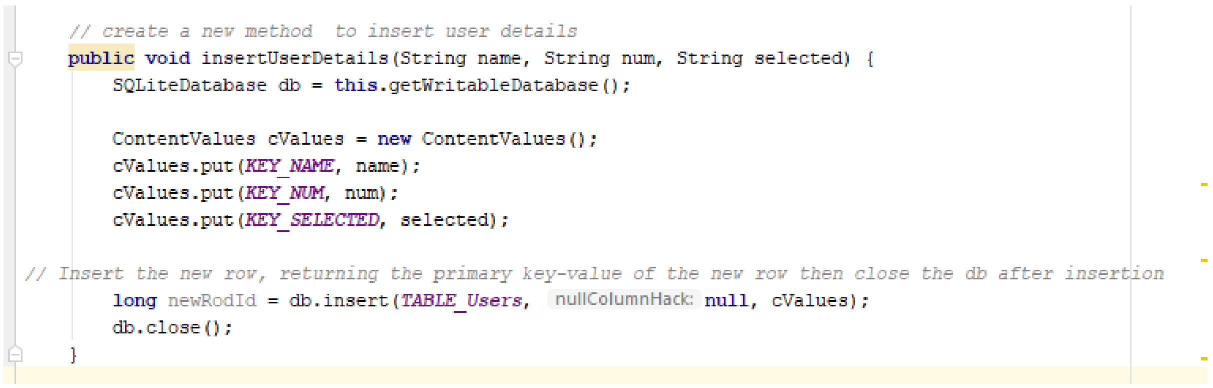


```
//In the onUpgrade() method, write a SQL command to drop the table if it exists ad re-create it if there's a new table to be created.
@Override
public void onUpgrade(SQLiteDatabase db, int i, int i1) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_Users);
    // Drop older table if exists
    onCreate(db);
    // Create table again
    //onUpgrade() method contains the code required to update the database.
}
```

In order to create new users, add a new method called insertUserDetails() within this class.

This method will require three (3) parameters namely name, location and designation all of String types, I have created them at the above step.

Still within the insertUserDetails() method, add the required code and call the necessary method to get the data repository in write mode.



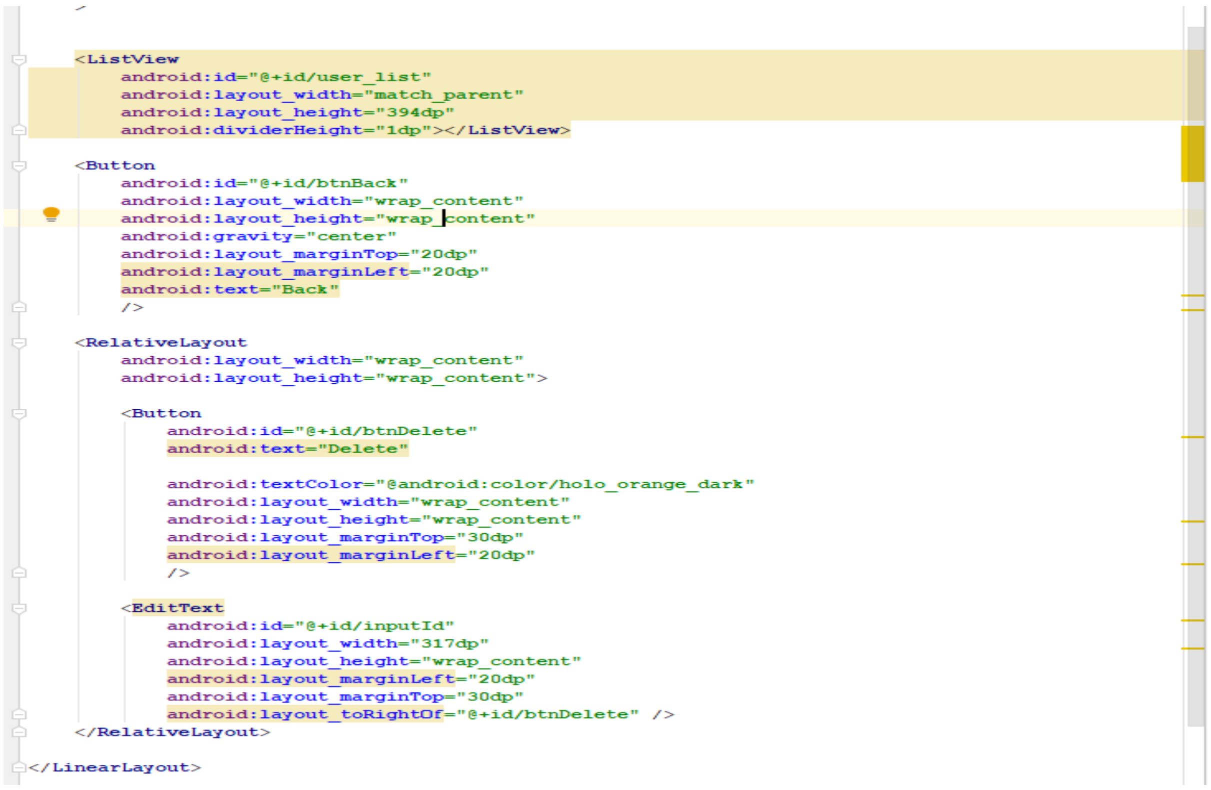
```
    // create a new method  to insert user details
    public void insertUserDetails(String name, String num, String selected) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues cValues = new ContentValues();
        cValues.put(KEY_NAME, name);
        cValues.put(KEY_NUM, num);
        cValues.put(KEY_SELECTED, selected);

// Insert the new row, returning the primary key-value of the new row then close the db after insertion
        long newRodId = db.insert(TABLE_Users,  nullColumnHack: null, cValues);
        db.close();
    }
```

In the above example the insert operation is handled by insertUserDetails() Method.
It takes name , num, selected as 3 arguments and insert them into table.
I have to first add all the values in **ContentValues** object ---“cValues”and then finally insert into table using **insert()** method of **SQLiteDatabase** class.

Read:
The adminsraters can see all users ordered information in AdminActivity.

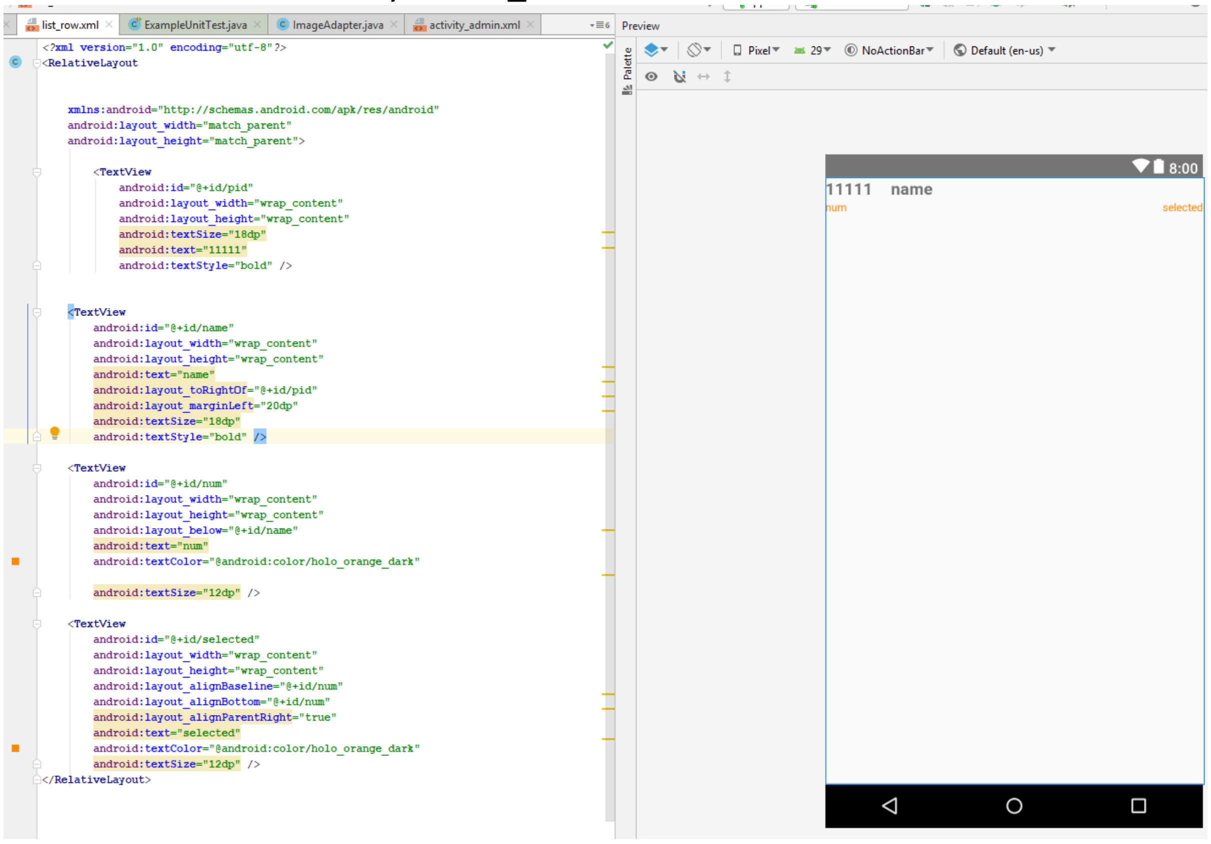
1.Layout:

Create a new empty activity called AdminActivity.
 While configuring this, set the name of the layout to activity_admin.xml



```
    <ListView
        android:id="@+id/user_list"
        android:layout_width="match_parent"
        android:layout_height="394dp"
        android:dividerHeight="1dp"></ListView>

    <Button
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        android:text="Back"
        />

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:id="@+id/btnDelete"
            android:text="Delete"

            android:textColor="@android:color/holo_orange_dark"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="30dp"
            android:layout_marginLeft="20dp"
            />

        <EditText
            android:id="@+id/inputId"
            android:layout_width="317dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="30dp"
            android:layout_toRightOf="@+id/btnDelete" />
    </RelativeLayout>

</LinearLayout>
```

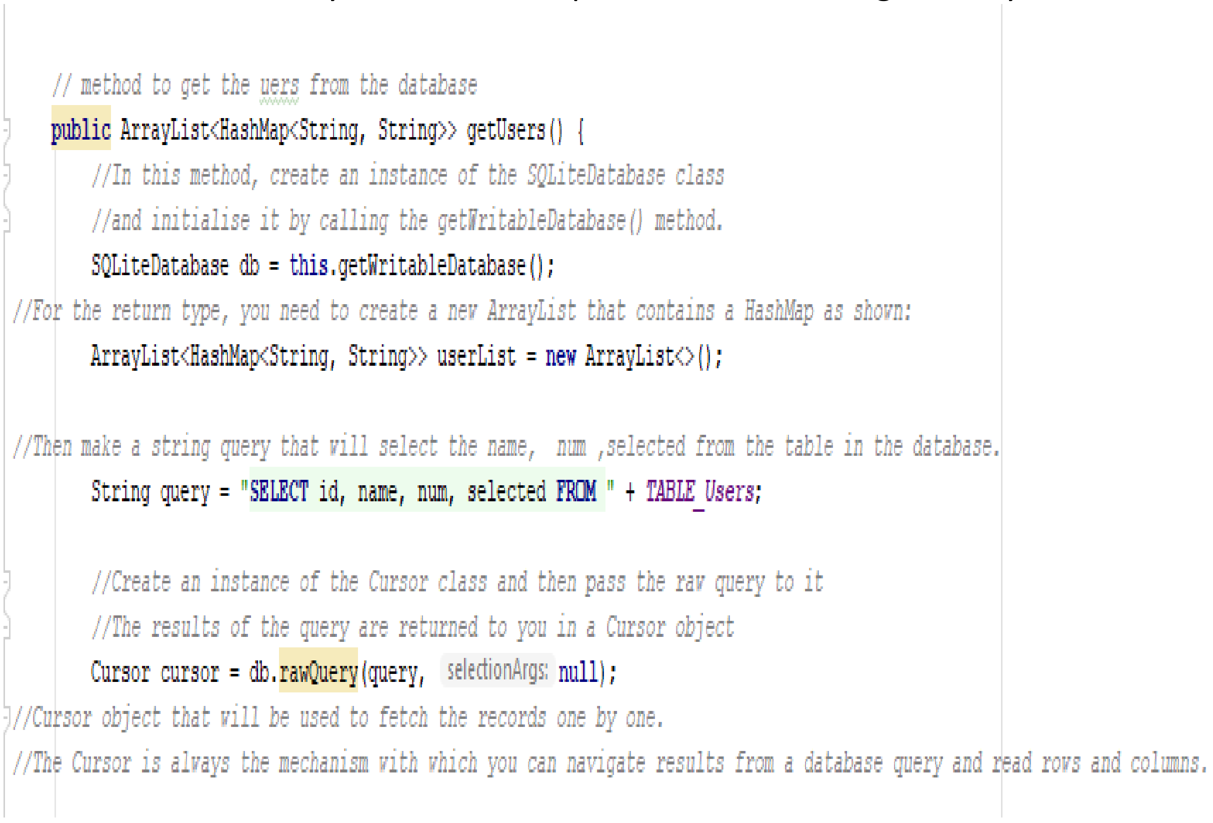
For the _admin layout, I am adding the following widgets:
Because the ListView will display rows in it as shown, I need to make another layout that will hold the row values.
So I create and call this layout list_row.xml.



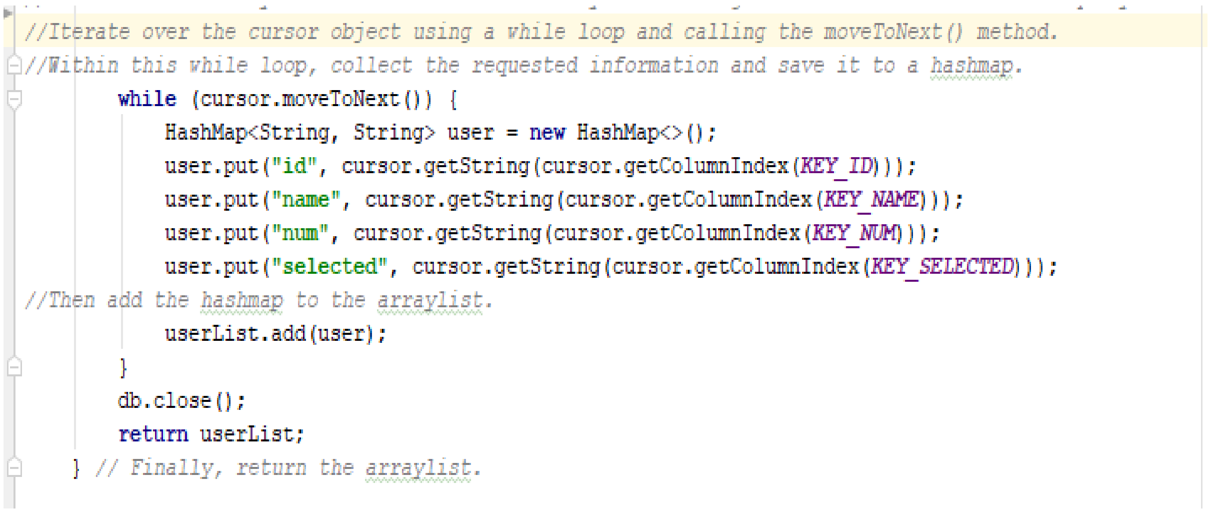
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout


    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <TextView
            android:id="@+id/pid"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18dp"
            android:text="11111"
            android:textStyle="bold" />


    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="name"
        android:layout_toRightOf="@+id/pid"
        android:layout_marginLeft="20dp"
        android:textSize="18dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/num"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/name"
        android:text="num"
        android:textColor="@android:color/holo_orange_dark"

        android:textSize="12dp" />

    <TextView
        android:id="@+id/selected"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/num"
        android:layout_alignBottom="@+id/num"
        android:layout_alignParentRight="true"
        android:text="selected"
        android:textColor="@android:color/holo_orange_dark"
        android:textSize="12dp" />
</RelativeLayout>
```

2. Behaviour:

Create a new method called getUsers() to view user details. This method should return an array list of hash map that contains strings for key and values.



```
    // method to get the uers from the database
    public ArrayList<HashMap<String, String>> getUsers() {
        //In this method, create an instance of the SQLiteDatabase class
        //and initialise it by calling the getWritableDatabase() method.
        SQLiteDatabase db = this.getWritableDatabase();
//For the return type, you need to create a new ArrayList that contains a HashMap as shown:
        ArrayList<HashMap<String, String>> userList = new ArrayList<>();

//Then make a string query that will select the name,  num ,selected from the table in the database.
        String query = "SELECT id, name, num, selected FROM " + TABLE_Users;

        //Create an instance of the Cursor class and then pass the raw query to it
        //The results of the query are returned to you in a Cursor object
        Cursor cursor = db.rawQuery(query, selectionArgs: null);
//Cursor object that will be used to fetch the records one by one.
//The Cursor is always the mechanism with which you can navigate results from a database query and read rows and columns.
```

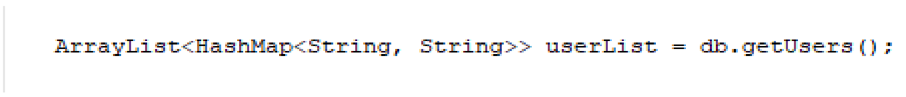


```
 //Iterate over the cursor object using a while loop and calling the moveToNext() method.
 //Within this while loop, collect the requested information and save it to a hashmap.
        while (cursor.moveToNext()) {
            HashMap<String, String> user = new HashMap<>();
            user.put("id", cursor.getString(cursor.getColumnIndex(KEY_ID)));
            user.put("name", cursor.getString(cursor.getColumnIndex(KEY_NAME)));
            user.put("num", cursor.getString(cursor.getColumnIndex(KEY_NUM)));
            user.put("selected", cursor.getString(cursor.getColumnIndex(KEY_SELECTED)));
 //Then add the hashmap to the arraylist.
            userList.add(user);
        }
        db.close();
        return userList;
    } // Finally, return the arraylist.
```

In the onCreate() method, create a DbHandler instance and initialise it with the 'this' keyword.

```
final DbHandler db = new DbHandler( context: this);
```

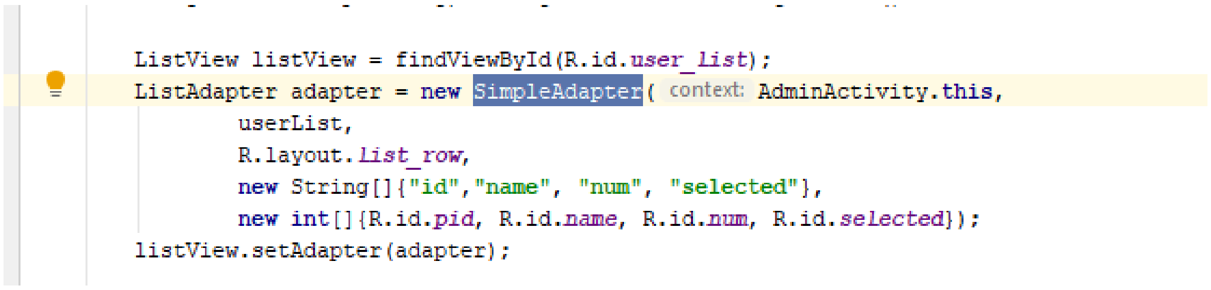
Similar to what I did in the DbHandler class, create an ArrayList that contains a HashMap. This will receive the returned value when the db instance calls the getUser() method.



```
ArrayList<HashMap<String, String>> userList = db.getUsers();
```

Initialise the ListView and Button instances by referencing their equivalent widget ids in the activity_admin.xml .

To display the list, add the following code that uses the “SimpleAdapter” to hold the information from the database.

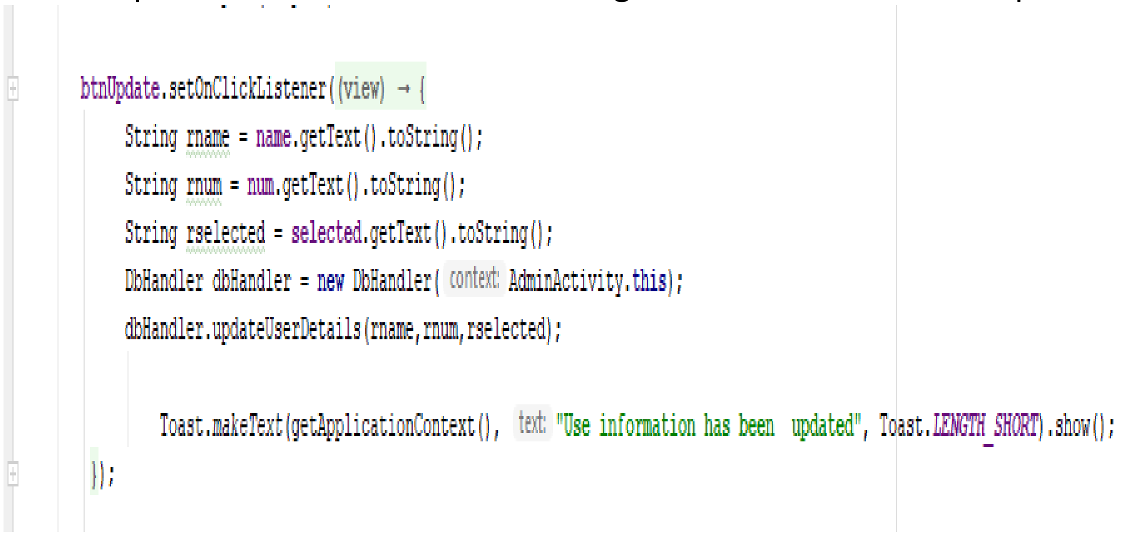


```
ListView listView = findViewById(R.id.user_list);
ListAdapter adapter = new SimpleAdapter( context: AdminActivity.this,
        userList,
        R.layout.list_row,
        new String[]{"id","name", "num", "selected"},
        new int[]{R.id.pid, R.id.name, R.id.num, R.id.selected});
listView.setAdapter(adapter);
```

Update:

1.Layout:

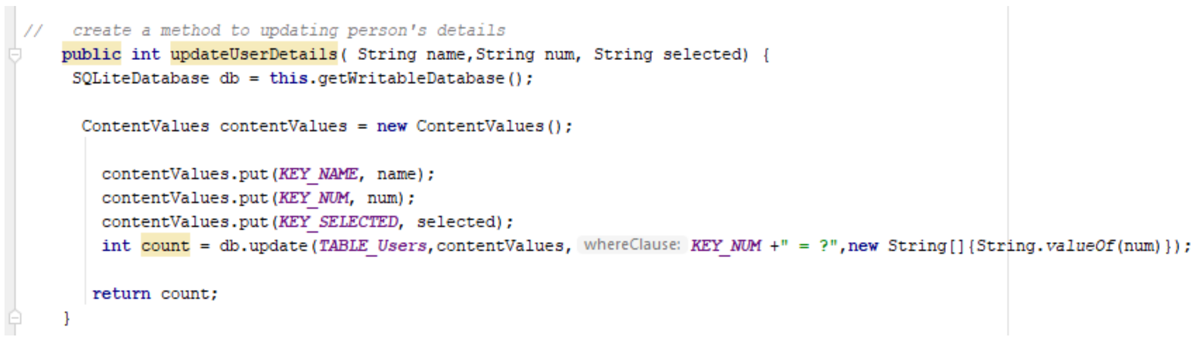
Set the onClickListener for the Update button. When this button is clicked, it should update the Ordered information got from three EditText input.



```
btnUpdate.setOnClickListener((view) → {
    String rname = name.getText().toString();
    String rnum = num.getText().toString();
    String rselected = selected.getText().toString();
    DbHandler dbHandler = new DbHandler( context: AdminActivity.this);
    dbHandler.updateUserDetails(rname,rnum,rselected);

      Toast.makeText(getApplicationContext(), text: "Use information has been  updated", Toast.LENGTH_SHORT).show();
});
```

2. Behaviour:

I create a method to update users' order details:

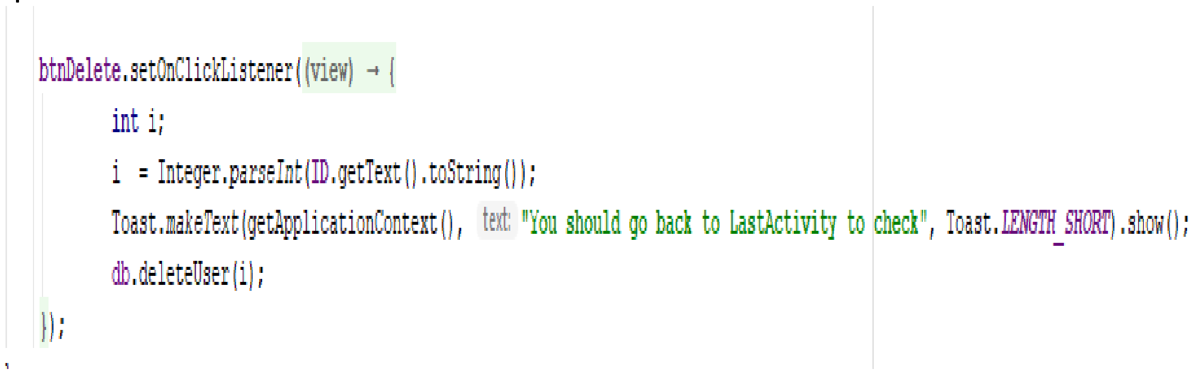


```
//   create a method to updating person's details
    public int updateUserDetails( String name,String num, String selected) {
     SQLiteDatabase db = this.getWritableDatabase();

      ContentValues contentValues = new ContentValues();

        contentValues.put(KEY_NAME, name);
        contentValues.put(KEY_NUM, num);
        contentValues.put(KEY_SELECTED, selected);
        int count = db.update(TABLE_Users,contentValues, whereClause: KEY_NUM +" = ?",new String[]{String.valueOf(num)});

       return count;
    }
```

Delete:

1.Layout:

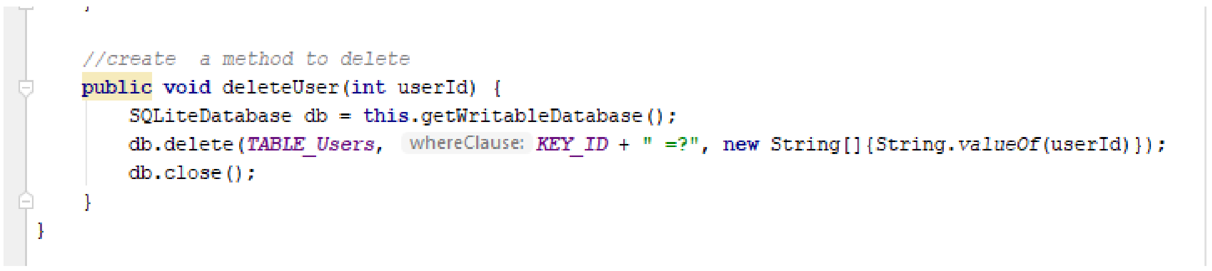
Set the onClickListener method for the Delete Button . When this button is clicked, it should delete the ordered information by the ID .got from EditText input.



```
btnDelete.setOnClickListener((view) → {
        int i;
        i  = Integer.parseInt(ID.getText().toString());
        Toast.makeText(getApplicationContext(), text: "You should go back to LastActivity to check", Toast.LENGTH_SHORT).show();
        db.deleteUser(i);
});
```

2. Behaviour:

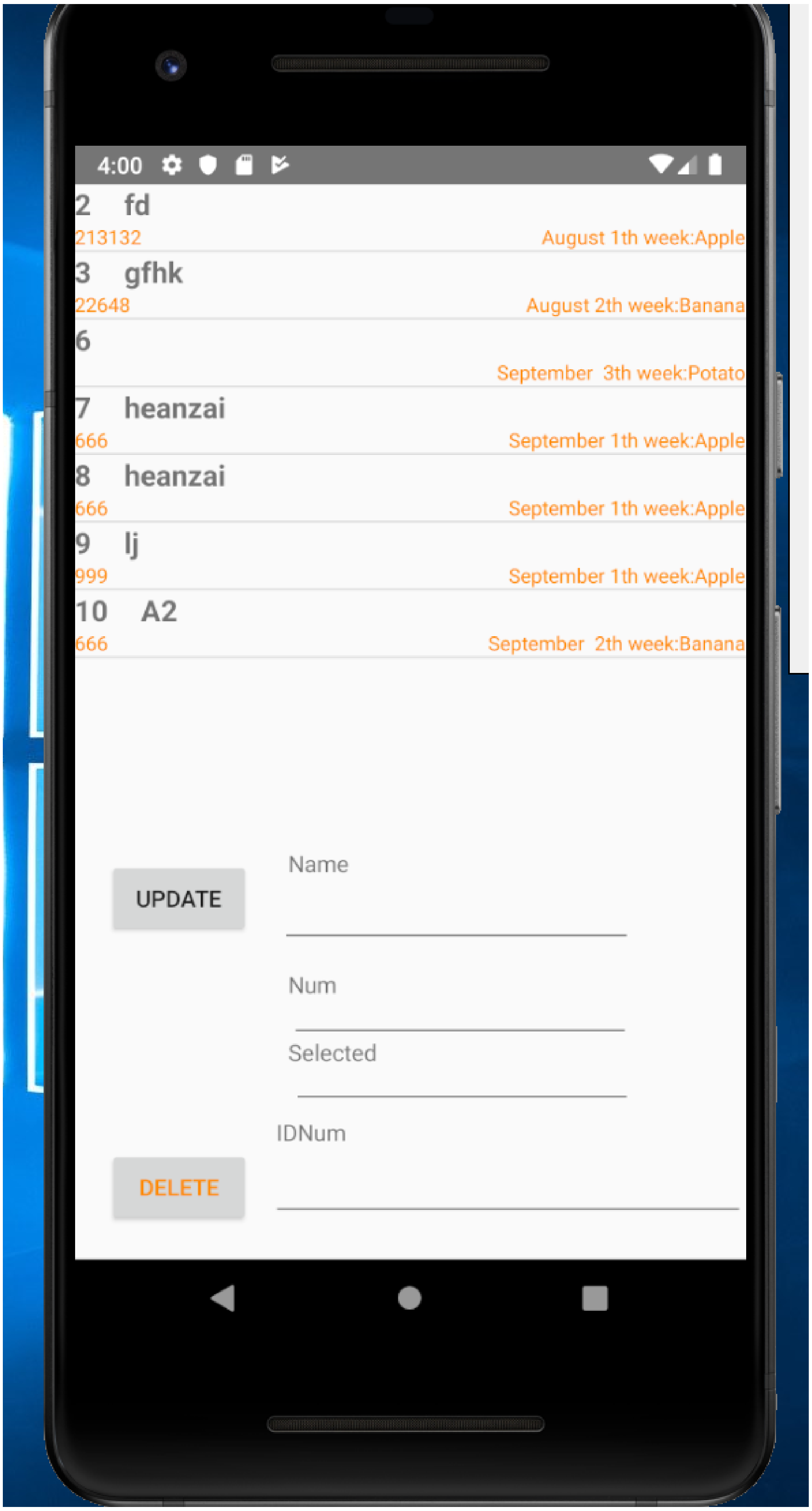
I create a method to delete the selected users' order details based on” ID”



```
        //create  a method to delete
        public void deleteUser(int userId) {
            SQLiteDatabase db = this.getWritableDatabase();
            db.delete(TABLE_Users,  whereClause: KEY_ID + " =?", new String[]{String.valueOf(userId)});
            db.close();
        }
    }
```

Discussion of constraints encountered, and strategies applied during the development

During the development , I find it is hard to delete ,because I need the “id” to delete, if “id” doesn’t show, it is hard to clarify the Num of ID, So I make “id” will be displayed also in the list so as to read and check.
In the above code, I put “id” also saved into a hashmap.



4:00

2 fd
213132 August 1th week:Apple

3 gfhk
22648 August 2th week:Banana

6
September 3th week:Potato

7 heanzai
666 September 1th week:Apple

8 heanzai
666 September 1th week:Apple

9 lj
999 September 1th week:Apple

10 A2
666 September 2th week:Banana

Conclusion

Above all , Database implementation (CRUD functions) have made the App become a better product to use , User can manage the order information and CRUD in one phone .

If this app want to be in market use , It needs the SQL in the cloud service and more testing to run well in different kinds of phones ,including IOS,Android.