Lab 14-15. Ứng dụng đăng nhập và đăng ký

(Phần 2. Lưu trữ thông tin với *SQLite* - 4 tiết)

I. Yêu cầu

- Sinh viên sử dụng công cụ Android Studio để làm bài. Kết quả bài làm cần được chụp lại và giữ lại toàn bộ dự án để sử dụng khi cần thiết.

- Lab này chỉ được thực hiện khi thực hiện xong Lab trước

- Khi có yêu cầu, sinh viên nộp qua email giáo viên hoặc một kênh khác.

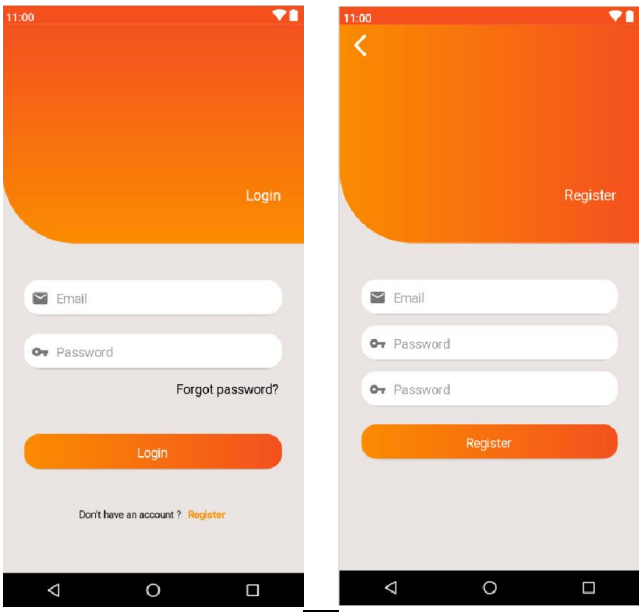
II. Luyện tập

Phạm vi kiến thức vận dụng

- Các kiến thức căn bản về Design UI
  - Drawable, Color Schemes, String constants, Style & themes
  - Layout constraint rules (padding, margin…)
- Các đối tượng View System
  - TextView, EditText, Button, ImageView
  - Các ViewGroup gồm: LinearLayout, FrameLayout, TableRow
- Các đối tượng xử lý logic: Activity, Fragment; SQLite

Nội dung bài thực hành

1. Yêu cầu: Tạo 1 ứng dụng **Account** và xây dựng giao diện layout sau sử dụng SQlite



Mô tả: Giao diện này đã được thực hiện ở bài lab trước, phần này chỉ thực hiện phần SQLite

2. Xây dựng đối tượng xử lý logic

Bước 1: Xây dựng lớp **Account**:

```
package com.example.account;

public class Account {
    String email;
    String pass;
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPass() {
        return pass;
    }
    public void setPass(String pass) {
        this.pass = pass;
    }

    public Account(String email, String pass) {
        this.email = email;
        this.pass = pass;
    }
}
```

Bước 2: Xây dựng lớp **SQLiteHelper** kế thừa lớp SQLiteOpenHelper như sau:

- Có các phương thức kết nối cơ sở dữ liệu
- Có các phương thức cơ bản như: insert, update, delete, getAll

```
package com.example.account;

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

import androidx.annotation.Nullable;

import java.util.ArrayList;
import java.util.List;

public class SQLiteHelper extends SQLiteOpenHelper {
    Context context;
    private static String DB_NAME ="acount.db";
    SQLiteDatabase myDB;
    public SQLiteHelper(@Nullable Context context){
        super(context,DB_NAME,null,1);
        this.context = context;
    }
    public void openDB() throws SQLException {
        if(myDB == null)
        myDB = getWritableDatabase();
    }
    public void closeDB(){
```

```
        if(myDB != null)
            myDB.close();
        super.close();
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
    public void createTable(){
        //myDB.execSQL("DROP TABLE IF EXISTS Account");
        String query = "create table if not exists Account (email text PRIMARY KEY, password
text)";
        myDB.execSQL(query);
    }
    public void insert(Account acc){
        ContentValues contentValues = new ContentValues();
        contentValues.put("email",acc.getEmail());
        contentValues.put("password",acc.getPass());
        myDB.insert("Account",null,contentValues);
    }
    public boolean login(String email, String pass) {
        String query = "select * from account where email ='"+email+"' and password='" +
pass+"'";
        Cursor cursor = myDB.rawQuery(query,null);
        return (cursor.getCount() > 0) ? true : false;
    }

   /* public void update(Account acc){
        ContentValues contentValues = new ContentValues();
        contentValues.put("email",acc.getEmail());
        contentValues.put("password",acc.getPass());
        myDB.update("Account",contentValues,("email = ?"),new
String[]{String.valueOf(acc.getEmail())});
    }
    public void delete(String email){
        myDB.delete("Account",("email= ? "),new String[]{String.valueOf(email)});
    }
    public List<Account> getAll(){
        List<Account> L = new ArrayList<>();
        String query = "select * from Account";
        Cursor cursor = myDB.rawQuery(query,null);
        while (cursor.moveToNext()){
            String email = cursor.getString(cursor.getColumnIndex("email"));
            String pass = cursor.getString(cursor.getColumnIndex("password"));
            Account acc = new Account(email,pass);
            L.add(acc);
        }
        return L;
    }
    */
}
```

Bước 3: Trong lớp MainActivity, viết phương thức InitialDB() và gọi trong hàm onCreate như sau:

```
private void InitialDB(){
    helper = new SQLiteHelper( context: this);
    helper.openDB();
    helper.createTable();
}
```

```
public class MainActivity extends AppCompatActivity {

    public static final String SAVE_PREF = "save_pref";
    SQLiteHelper helper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        InitialDB();
        gotoLoginScreen();
    }
```

Bước 4: Trong lớp M001RegisterFragment thêm các đoạn mã như sau:

```
SQLiteHelper helper;
public M001RegisterFragment(SQLiteHelper helper){
    this.helper = helper;
}

private void register(String mail, String pass, String repass) {
    if (mail.isEmpty() || pass.isEmpty() || repass.isEmpty()) {
        Toast.makeText(mContext, text: "Empty value", Toast.LENGTH_SHORT).show();
        return;
    }
    if (!pass.equals(repass)) {
        Toast.makeText(mContext, text: "Password is not match", Toast.LENGTH_SHORT).show();
    }
    Account account = new Account(mail,pass);
    helper.insert(account);
    SharedPreferences pref = mContext.getSharedPreferences(MainActivity.SAVE_PREF, Context.MODE_PRIV
    String savedPass = pref.getString(mail, defValue: null);
```

Lưu ý: Sinh viên có thể bỏ hoặc che đi đoạn mã liên quan tới SharedPreferences nếu muốn.

Bước 5: Trong lớp M000LoginFragment thêm đoạn mã như sau trong phương thức login. Lưu ý, sinh viên phải che đi đoạn mã về login bằng SharedPreferences:

```
SQLiteHelper helper;
private void login(String mail, String pass) {
    if (mail.isEmpty() || pass.isEmpty()) {
        Toast.makeText(mContext, text: "Empty value", Toast.LENGTH_SHORT).show();
        return;
    }
   /* SharedPreferences pref = mContext.getSharedPreferences(MainActivity.SAVE_PREF, Context.MODE_P
    String savedPass = pref.getString(mail, null);
    if (savedPass == null) {
        Toast.makeText(mContext, "Email is not existed!", Toast.LENGTH_SHORT).show();
        return;
    }
    if (!pass.equals(savedPass)) {
        Toast.makeText(mContext, "Password is not correct!", Toast.LENGTH_SHORT).show();
        return;
    }*/
    if(helper.login(mail,pass))
    Toast.makeText(mContext, text: "Login account successfully!", Toast.LENGTH_SHORT).show();
    else
        Toast.makeText(mContext, text: "Wrong email or password!", Toast.LENGTH_SHORT).show();
}
```

III. Bài tập

1. Sinh viên thực hiện code để sau khi đăng nhập thì hiển thị danh sách tài khoản đã đăng nhập. Có nút cho phép xem profile từng tài khoản.

2. Lý thuyết về SQLite, sinh viên xem thêm tại đây:

https://openplanning.net/10433/android-sqlite-database

--Hết--