소프트웨어 프로젝트 1

(Project3)

수업명:소프트웨어프로젝트1

소속: 컴퓨터정보공학부

학번: 2018202074

이름: 김상우

프로젝트 설명

해당 프로젝트는 안드로이드 스튜디오를 이용해 로그인 및 회원가입을 하는 앱을 제작하는 것을 목표로 하고 있습니다. Data를 저장하기 위한 수단으로 SQLITE를 사용할 예정입니다. 프로젝트에 대하여 최소 4개의 Activity가 구현되어야 하며 필수적인 구현은 다음과 같습니다.

-로그인 화면

아이디와 비밀번호를 User가 적을 수 있어야 하며 해당 화면에 이미지가 필요함.

또한 로그인 버튼과 회원가입 버튼이 있으며 로그인 버튼은 아이디와 비밀번호를 올바르게 입력 시 해당 정보에 맞는 유저화면으로 이동, Admin을 아이디, 1을 비밀번호로 입력할 경우 관리자 화면으로 넘어가게 된다. 그 외의 경우 로그인을 허락하면 안 된다. 또한 회원가입 버튼을 통해 회원가입 화면으로 이동을 할 수 있어야한다.

추가적인 구현 조건으로 로그인 화면에서 이름, 생년월일, 이메일을 이용해 비밀번호를 구할 수 있어야 한다. 이를 구현한다면 비밀번호를 구하기 위한 화면으로의 이동이 조건에 따라서 로그인 화면에서 가능해야 하므로 이를 위한 트리거가 필요해질 것이다.

-회원가입 화면

User로부터 이름, 아이디, 생년월일(년/월/일), 비밀번호, 이메일에 대해 입력받고 하단 회원가입 버튼을 통해 해당 정보를 데이터베이스의 등록해 로그인 및 비밀번호 검색 등이 가능하게 만들어 주어야 합니다. 이때, 아이디 입력란 옆에는 중복확인 버튼이 존재하며 이 중복버튼을 통해 중복 체크를 하지 않으면 회원가입이 되지 않습니다. 또한 ID가 이미 사용중인 경우나 관리자 접근을 방지하기 위해 Admin을 사용할 경우에 해당 ID를 사용할 수 없다고 선언해주어야합니다.

-관리자화면

로그인 화면에서 Id로 admin, 비밀번호로는 1을 받을 경우 열리게 되며 현재 존재하는 데이터들이 이름/Id/Email의 format으로 List의 형태로 출력되어야 한다. 또한 Data별로 Radio button을 통해 선택되어야 한다.

우측 상단에는 버튼 2개가 존재하며 버튼 중 왼쪽은 Login으로 돌아가기 위한 버튼, 우측은 선택한 것을 기반으로 해당 Data를 삭제 시키는 버튼이다. 특정 Data가 선택된 후 삭제 시키는 버튼이 동작될 경우 해당 Data는 Database에서 삭제된다.

-유저 화면

로그인 화면에서 적합한 Id와 비밀번호를 입력하고 로그인 버튼을 클릭하는 것으로 입장할 수 있다. 해당 화면에선 "(이름)님의 정보"로 입력된 문자열을 상단에 두며 아래엔 Login으로 돌아갈 수 있는 Imagebutton, 그 아래로는 이름과 아이디, 생년월일(format은 년/월/일), 이메일이 출력된다.

추가적인 구현 조건으로 해당 화면에서 이름, 생년월일, 이메일에 대하여 수정을 할 경우 해당 데이터가 실제로 변경되어야 한다.

-비밀번호 검색 화면(추가구현)

이름, 생년월일(년, 월, 일), 이메일을 입력 받고 이를 바탕으로 비밀번호를 출력하여야 한다.

-액티비티 별 결과 캡쳐 및 구현방법 설명

-DBHelper

SQLiteOpenHelper를 extends 하여 사용, 생성자로서 super를 사용해 SQLiteOpenHelper와 연결된 상태에서 이를 초기화하며 onCreate에서는 SQL문을 기반으로 하여 create Table users(username Text primary key, password TEXT, idinput TEXT, yearinput TEXT, monthinput TEXT...)를 통해 각각 ID를 primary key로 하며 비밀번호, 사용자, 년, 월, 일, 이메일을 표의 구분 속성으로 해주는 형태로 제작하였다. 이후 onUpgrade를 통해 다시 불리더라도 users 테이블이 이미 존재한다면 drop할 수 있도록 해주었다.

updateData함수를 선언, 비밀번호를 제외한 모든 정보를 input으로 받아 contentValues.put을 통해 table에 넣을 수 있는 형태로 만든 후 update함수를 이용 users테이블에 만든 contentvalue를 username, 즉 primary key가 입력된 id에 맞는 곳을 찾아 해당 정보로 update하게 해주었다.

deleteDate함수를 선언 delete함수를 통해 users라는 테이블에서 primary key로 입력된 id를 갖는 곳을 delete해주었다. 성공시 return true, 실패시 return false해주었다.

checkusername함수를 선언, cursor를 이용해 입력된 id를 primary key로 갖는 data를 탐색, cursor의 getCount함수를 이용해 존재하지 않는다면 return false, 존재한다면 return true를 진행한다.

checkusernamepassword함수를 선언, cursor를 이용해 입력된 id를 primary key로 갖고 입력된 password를 password로 갖는 data를 탐색, cursor의 getCount함수를 이용해 존재하지 않는다면 return false, 존재한다면 return true를 진행한다.

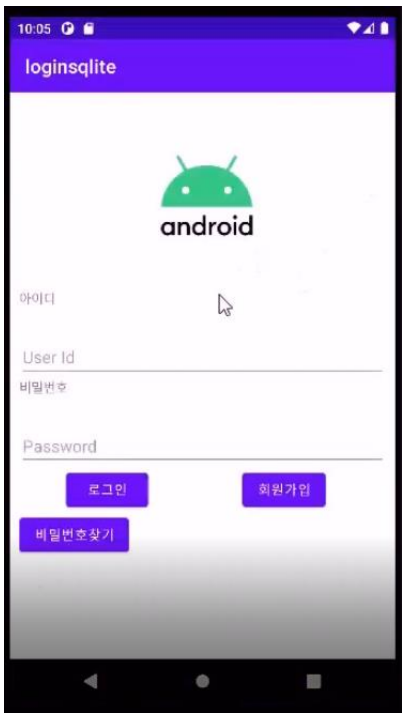
findpassword함수를 선언 password와 id를 제외한 모든 정보를 인자로 받고 rawQuery를 이용 Select를 통해 users 테이블에서 입력된 값을 이름, 년, 월, 일, 이메일로 갖는 data를 찾는다. getCount를 통해 존재하지 않을 시 return false, 존재할 시 return true를 진행한다.

Passwordprint2함수를 선언, password와 id를 제외한 모든 정보를 인자로 받고 rawQuery를 이용 Select를 통해 users 테이블에서 입력된 값을 이름, 년, 월, 일, 이메일로 갖는 data를 찾는다. Cursor를 moveToNext를 통해 옮기며 해당 data를 찾고 결과적으로 맞는 것에 대하여 return 비밀번호를 찾아 이에 대한 return를 진행한다.

Passwordprint함수를 선언, password와 id를 인자로 받고 rawQuery를 이용 Select를 통해 users 테이블에서 입력된 값을 primary key로 갖는 data를 찾는다. Cursor를 moveToNext를 통해 옮기며 해당 data를 찾고 결과적으로 맞는 것에 대하여 return 비밀번호를 찾아 이에 대한 return를 진행한다.

Idprint, yearprint, monthprint, dayprint, emailprint 함수를 선언 이들은 모두 Id를 인자로서 받고 rawQuery를 통해 입력된 값을 Id로 받는 data를 찾아 cursor을 통해 한번 더 확인후 getString을 통해 각자 알맞은 값을 찾아 이를 return하는 방식으로 함수를 진행시킨다.

-로그인 화면



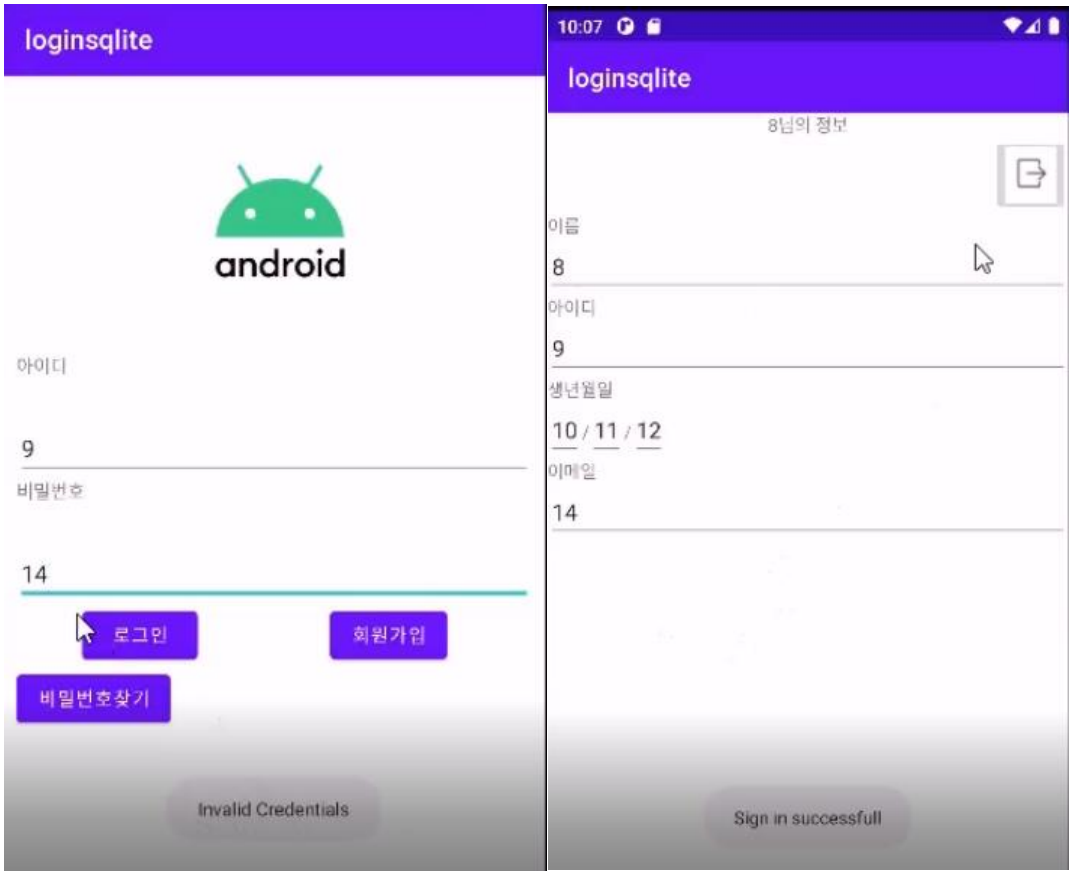
Activity_login.xml

로그인 화면의 모습이다. 기존에 제시된 배치를 위해 XML파일에 <ImageView>를 이용, drawable 파일 내에 해당 파일을 넣은 후 android:src='@drawable.button'을 지정하여 해당 파일 내 이미지를 불러올 수 있게 해주었다. 이후 TextView와 EditText를 바탕으로 아래와 같이 입력할 수 있는 칸과 입력해야 하는 것에 대한 텍스트를 배치했다 이후 Button을 통해 로그인, 회원가입, 비밀번호검색에 해당하는 화면으로의 전환을 위한 버튼을 클릭을 통해 받을 수 있도록 하였다.

LoginActivity

위에 배치된 것을 기반으로 findViewById(R.id.(id))를 바탕으로 해당 파일내 변수와 연결하여 값을 받아오게 하였다. EditText의 경우 getText().toString을 이용 입력된 값을 문자열로서 받아왔다. 이때 setOnClickListener함수와 onClick함수를 이용, 로그인 버튼을 누를 때 문자열을 받아올 수 있도록 하였다. 이후 equals함수를 이용해 Id가 admin 비밀번호가 1인지 확인하고 만약 맞다면 Intent new Intent(getApplicationContext(), AdminActivity.class)를 통해 관리자 화면을 의미하는 AdminActivity로 이동할 수 있게 해주었으며 이때 putExtra함수를 이용, deletename이라는 이름으로 admin이라는 문자열을 넣어주었다. 이는 추후 관리자 화면에서 삭제될 데이터를 선택하지 않은 상태임을 나타내기 위해 사용된다. 만약 관리자 로그인이 아닌 경우 Id가 admin이거나 Id, password가 입력되지 않는 경우 Wrong Input이라는 Toast Message를 띄워준다. 만약 admin에 해당하지 않은 로그인 정보가 들어올 경우 checkusernamepassword를 이용, 입력된 ID와 비밀번호를 갖는 Data가 존재하는 지 확인하고 존재할 경우 intent를 통해 findname이라는 이름으로 id를 putExtra로 추가하고 HomeActivity로 이동할 수 있게 해준다.(startActivity) 만약 로그인 시도임에

도 로그인정보가 맞지 않으면 ToastMessage를 출력해준다.

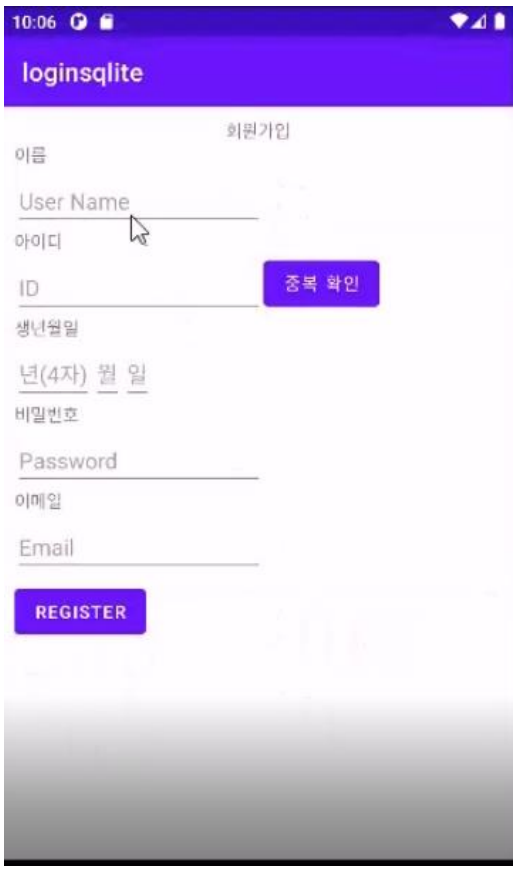


(좌)잘못된 비밀번호 입력시 Toast메시지를 통해 틀렸음을 전달

(우)로그인 정보로 잘 입력하자 Sign in이 잘됨을 ToastMessage로 알려주며 유저 화면이 뜬다.

마찬가지로 Button1에서 사용한 SetOnclickListener와 startAcitivity를 이용, 회원가입과 비밀번호 찾기 버튼을 누를 시 알맞은 화면으로 이동함을 확인했다.

-회원가입 화면



Activity_main.xml

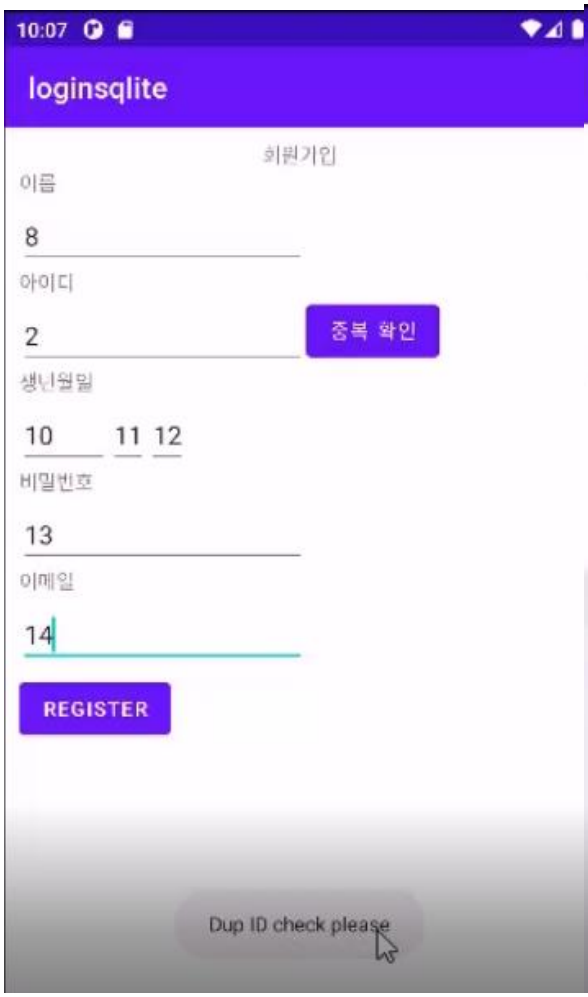
위는 회원가입 화면의 모습이다 제시된 것과 같이 이름, 아이디, 생년월일(년, 월, 일)에 대해 적을 수 있게 되어있으며,(TextView를 기반으로 Text를, EditText를 기반으로 입력할 수 있는 Text칸을 제작하였다.) 생년월일에 대한 부분이나 아이디 옆 중복확인 버튼의 경우 가로줄로 표현하기 위해 layout_toRightOf를 사용하였으며 수직적인 표현은 layout_below를 통해 구현하였다.

MainActivity.java

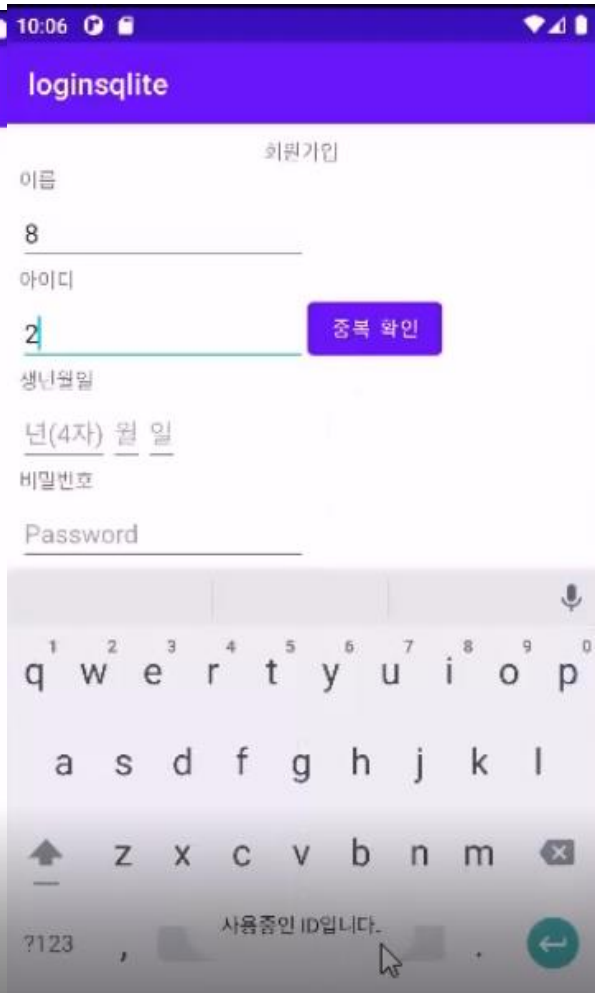
로그인 화면과 마찬가지로 findViewById를 이용, 해당 요소들을 코드에서 접근할 수 있게 해주었다. 또한 중복확인 체크를 위해 Idchk라는 변수를 추가하였으며 중복 ID확인을 위한 Database접근을 위해 DBHelper를 추가하였다. 이때 중복확인 후 아이디가 수정되는 경우를 막기위해 addTextChangedListener를 사용, onTextChanged함수 내에 Idchk=false를 넣어주어 Id에 해당하는 EditText에 값이 변경될 시 해당 변수를 false가 될 수 있게 해주었다.

이후 SetOnClickListener함수를 이용 중복확인 버튼을 눌렀을 시 checkusername을 통해 Id를 함수로서 Id가 존재하는 지 확인한다. 이후 존재하지 않는다면 Idchk를 true로 만들어주고 Toast Message로 사용 가능하다는 메시지를 출력해주었다. 이때 예외처리로 Id가 이미 사용중일 때와 공백이 입력되었을 경우를 처리해주었다.

또한 회원가입 버튼에 대한 클릭을 처리하기 위해 setOnClickListener를 이용, onClick내부에서 getText().toString을 통해 입력된 값들을 받으며 이후 아이디와 비밀번호 칸이 비어있지 않은지 확인한다.(예외처리로 다른 정보의 경우 이후 login을 통해 수정가능 하므로 위의 2개만을 처리했다.) 또한 ID 중복을 한번 더 확인하고 (checkusername이용) Idchk가 true인지 확인하여 회원가입 여부를 판단한다. 만약 모든 조건을 통과하였을 경우, insertData를 통해 해당 값들을 Database에 저장한다. 이후 startActivity를 통해 Login화면으로 돌아간다.

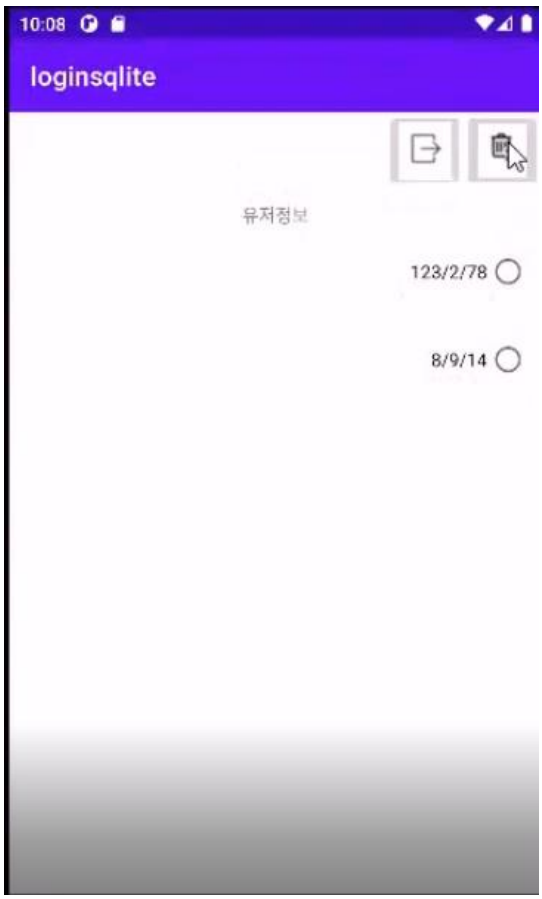


(좌)중복확인을 하지 않았을 경우



(우)중복확인 결과로 중복이 판별된 모습이다.

-관리자 화면



위 화면을 구현하기 위해서는 두개의 xml파일이 사용되었다.

Activity_admin.xml

AdminActivity 파일에 직접적으로 연결된 xml파일로 상단의 버튼 2개를 ImageButton을 통해 받으며 (우측에 붙은 상태로 가로 출력을 위해 LinearLayout, gravity="right"을 사용) 유저정보라는 text를 textview로서 받고 ListView를 통해 DataBase의 정보들을 출력한다. 이때 사용되는 List의 정보는 item_listview.xml파일에 저장되어있다.

Item_listview.xml

Activity_admin.xml의 ListView에 대한 내용을 답고 있다. 실제로 작동하는 것은 Radiobutton으로 선택을 위한 버튼을 우측으로 하기 위해 layoutDirection="rtl"을 이용해 선택버튼을 우측으로 하였으며 alignParentRight를 이용, 우측에 생성될 수 있도록 하였다.

ListViewAdapterData.java

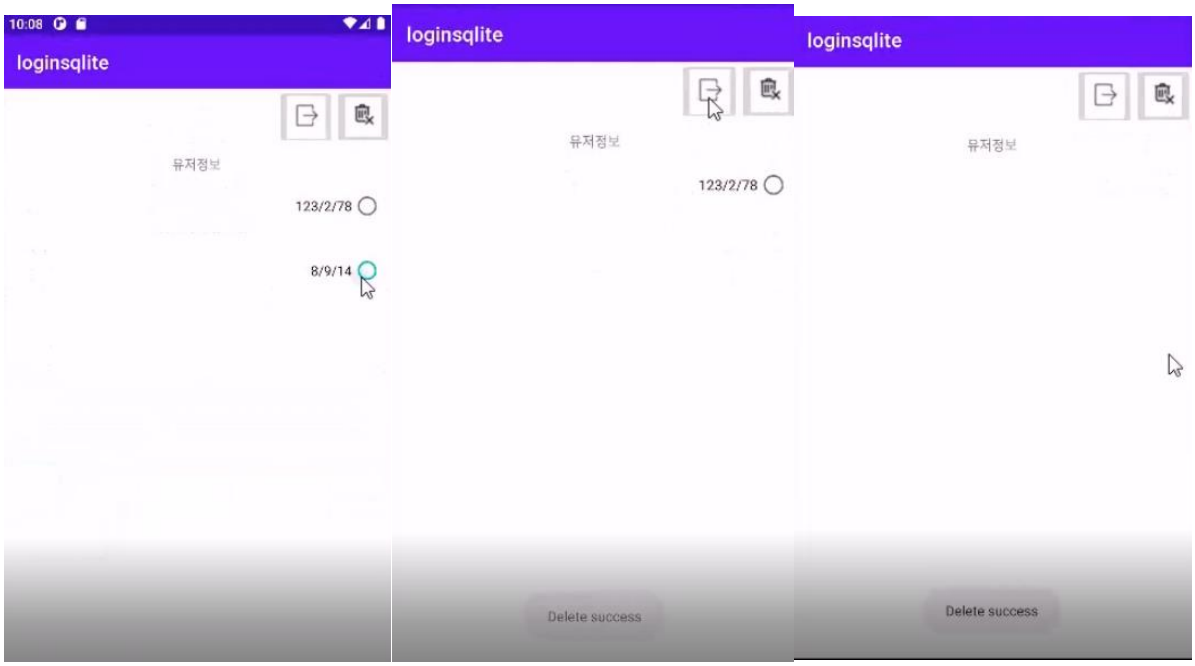
리스트에 출력될 값들을 변수로 가지며 Get, Set을 진행하는 class이다. 이름과 아이디, 이메일에 대해 this.변수를 바탕으로 진행한다.

ListViewAdapter

BaseAdapter를 extends하여 사용한다. ListViewAdapterData를 ArrayList로서 다을 수 있는 배열을 생성하고 GetCount, getItem, getItemId에 대해 Override를 해주었다 .또한 추가적으로 View getView를 선언해주었다. 내부에서는 LayoutInflater와 Context를 사용, context.getSystemService를 이용해 LayoutInflater를 사용할 수 있도록 만들고 이를 View형태의 변수에 또다시 저장, 이후 findViewById함수를 바탕으로 RadioButton을 코드와 연결하였다. 이후 setOnClickListener를 이용해 radio button이 클릭되었을 때, Intent를 생성하여 다시금 해당 화면으로 들어올 수 있게 해주었다. 이때, putExtra를 이용, deletename이라는 이름으로 선택된 데이터의 Id를 다시금 틀어지는 AdminActivity에 전달할 수 있게 해주었다.(이때 id의 경우 ListViewAdapterData를 이용하였으며 이전에 만들어둔 list를 기반으로 동작되었다.) 추가적으로 외부에서 값들을 print하기 위해 view에 이름, 아이디, 이메일에 대한 정보를 list에 추가하기 위한 함수 addItemTolist를 만들어두었다.(set과 add로 이뤄짐.)

AdminActivity

AppCompatActivity를 extend해서 사용, ListView와 ImageButton을 변수로 받아오고, DBHelper또한 선언한다. 이후 getStringExtra를 바탕으로 해당 Activity를 호출 시 Extra로서 들어온 deletename이란 이름의 문자열을 받아 저장한다. 이후 이후 setOnClickListener를 통해 나가기 버튼이 눌러질 경우 startActivity를 통해 LoginActivity로 이동할 수 있게 하며 만약 삭제 버튼을 누를 경우 extra로 받아온 값이 admin인지 확인, admin일 경우 삭제 대상이 없는 경우이므로 Toast Message를 띄워주며 admin이 아닐 경우 삭제 대상이 지정된 상황이므로 deleteDate함수를 이용, extra로 받아온 String값을 Id로서 갖는 데이터를 찾아 삭제한다. 이후 startActivity를 통해 다시금 해당 화면을 호출하게 해준다. 버튼 클릭과 별개로 displayList하는 함수를 호출할 수 있게 해준다. 해당 함수는 ViewList출력을 위한 함수이다. SQLiteDatabase의 형태를 갖는 helper를 호출하고 cursor를 rawQuery함수를 통해 제작한다. 이때 SELECT * FROM users라는 sql을 통해 Database를 지정해주었다. 히우 ListViewAdapter를 클래스로 갖는 변수를 선언한다. 이후 cursor를 moveToNext함수를 이용해 처음부터 끝까지 옮기며 adapter의 addItemToList함수를 이용, cursor.getString을 이용해 표에서 이름, Id, 이메일에 관한 정보를 adapter의 list로 넣는다. 이후 Viewlist의 setAdapter를 통해 방금 만든 adapter를 listview의 adapter로 설정하여 listview를 출력할 수 있게 한다.



(좌)데이터에 해당하는 Radio button 클릭 후 위의 삭제 버튼 클릭을 하자 (우)다시 생성된 관리자 화면에서는 해당 Data가 더 이상 출력되지 않음을 알 수 있다. Delete success라는 Toast Message 또한 출력된다. 모든 데이터가 삭제되어도 에러는 발생하지 않는다.

-유저 화면

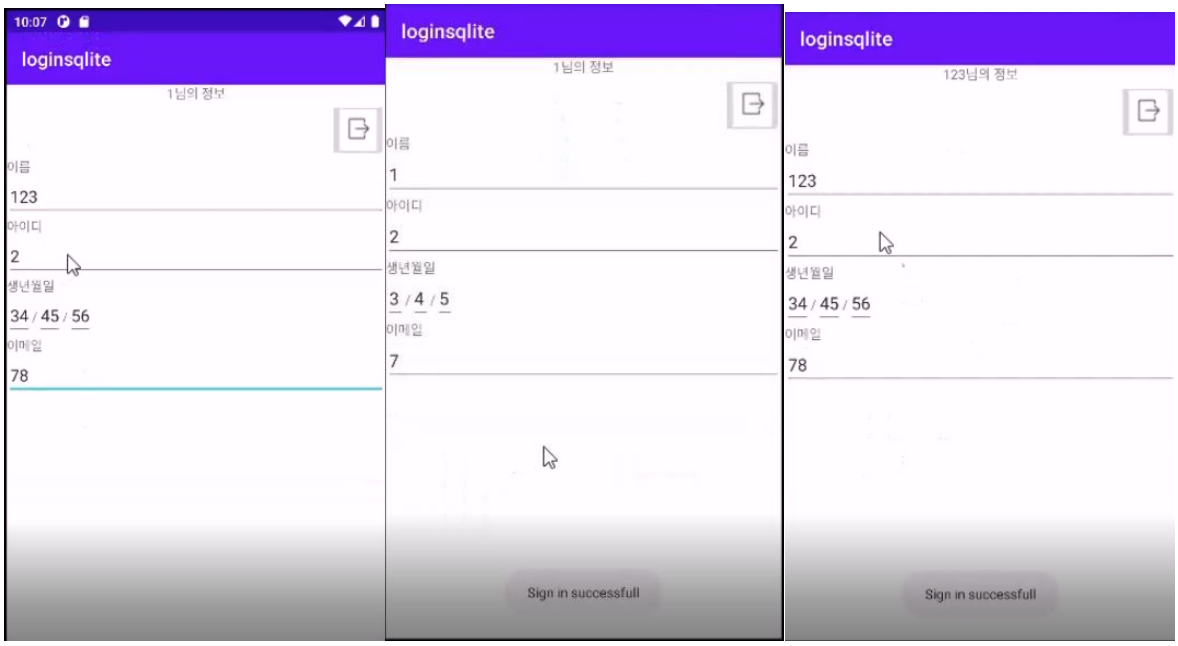
Activity_home.xml

TextView와 EditText를 이용해 기본 text와 입력받을 수 있는 text칸을 만들었으며 ImageButton을 통해 나가는 버튼을 이미지를 기반으로 만들었다.(이전에 서술한 방식과 같다.) 이때 추가적인 구현을 위해 데이터 출력을 위한 부분을 모두 EditText로 제작하였다.

HomeActivity.java

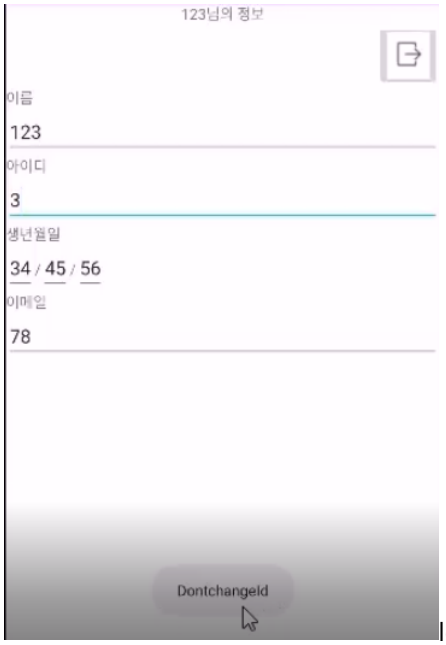
TextView, EditText, DBHelper, ImageButton을 Class로 갖는 변수들을 선언, 해당 변수들과 맞는 것들을 findViewById를 통해 연결했다. 이때 가장 상단의 ~님의 정보를 출력해주기 위해 TextView 또한 변수로 받는 것을 확인할 수 있다. 이후 DBHelper에서 만들어둔 yearprint, monthprint, dayprint, 등의 함수를 이용, putextra로 findname이라는 이름으로 넘겨진 아이디를 인자로 하여 해당 data에 해당하는 모든 정보들을 String으로 받아주었다. 이후 setText를 이용, EditText와 TextView에 해당 값들을 알맞게 넣어주었다. 또한 SetOnClickListener를 이용 나가기 버튼을 감지하게 해주었다. 이때 추가구현을 위해 변경된 값들을 받아 Database에 업데이트할 필요가 있는데, Update는 이 버튼이 눌려 해당 화면에서 나가고자 할 때 작동하게 해주었다. 이러한 처리를 위해 getText().toString을 이용, 버튼을 누른 당시의 EditText의 문자열들을 받는다. 이후 id가 변했는지 확인, 변했을 경우 에러가 발생하므로 Toast Message를 출력해주고 Update는 진행시키지 않는다. 만약 Id는 변하지 않았다면 다른 모든 값들의 변경을 확인, 변경되지 않았을 경우 StartActivity를 통해 login으로 나갈 수 있게 해주었고, 만약 하나라도 바뀌었을 경우 updateData를 활용, 입력된

변수들을 인자로서 해당 함수를 실행한다. 이후 startActivity를 이용해 login화면으로 나간다.



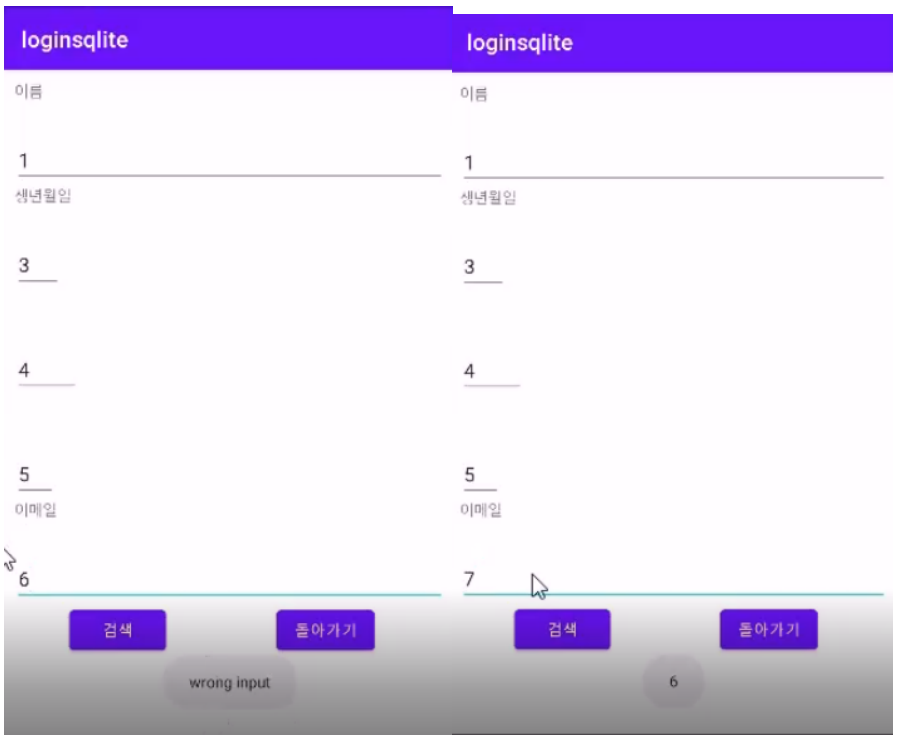
(추가구현)

(1)->(2)로 수정하고 나가기 버튼을 누른 후 다시 login한 모습이다. 상단의 이름은 물론 다른 정보들도 적절히 바뀐 모습이다.



Id를 변경한 경우 에러이므로 ToastMessage출력

-비밀번호 검색 화면(추가구현)



(좌)알맞지 않은 정보를 입력 시 wrong input를, (우)올바른 정보를 입력시 비밀번호를 toast message로서 띄워주는 것을 확인할 수 있다.

Activity_pass.xml

TextView와 EditText를 바탕으로 입력해야하는 것들을 표시하고 입력하는 값들을 받을 수 있게 해주었다.(이전 화면들에 서술)

PassActivity.java

EditText, Button class를 기반으로 하는 변수들을 생성, OnCreate내에서 findViewById를 통해 입력된 값들을 EditText 변수들에 받고 setonClickListener와 onClick을 통해 button클릭을 인식할 수 있게 해주었다. 또한 DBHelper 변수를 선언해주어 Database에 접근할 수 있게 해주었다.

이렇게 받은 값들에 대해 get.Text().toString()으로 입력된 값들을 받고 equals함수를 통해 빈 입력칸이 있는 지 확인하여 있을 경우 Toast message를 통해 빈 곳을 채우라는 message를 출력한다. 만약 빈속이 없다면 findpassword를 통해 적힌 값들을 인자로 받아 해당 정보들을 갖는 Data가 있는지 확인, 있을 경우 passwordprint2함수를 통해 password를 받아 toast message로서 출력한다. 만약 그런 데이터가 존재하지 않는 경우 wrong input이라는 toast message를 띄워준다. 또한 추가적으로 login화면으로 돌아갈 수 있는 button의 클릭 또한 같은 방법으로 감지하고 이후 startActivity를 통해 login화면으로 갈 수 있도록 해주었다.

-YouTube에 데모 영상 업로드 후 링크 첨부

TestBench와 결과는 다음과 같습니다.

1. admin화면을 로그인을 통해 접근, 비어있는 관리자 화면이 확인되며 나오는 것도 문제 없었다.
2. 회원가입 화면으로 이동, 빈 곳이 있을 경우와 중복을 누르지 않을 경우의 처리를 확인하고 올바른 입력 후 정상적으로 login화면으로 돌아가는 것을 확인했다.
3. login화면에서 비밀번호 찾기 화면으로 이동, 2번에서 설정된 유저에 대해 입력하여 비밀번호가 Toast message로 잘 뜨는지 확인하고 알맞지 않게 넣었을 때와 빈칸이 존재하는 경우에 대한 에러 메시지 처리가 되는 것을 확인했다. 이후 login 화면으로 돌아가는 버튼 또한 정상적으로 작동하는 것을 확인했다.
4. login화면에서 회원가입 화면으로 이동, 이전에 등록한 아이디를 다시 사용하려 하자 중복확인 버튼을 통해 사용이 차단됨을 확인했다. 이후 다른 값으로 변경 후 등록하니 정상적으로 작동하는 것을 확인했다.
5. 2번째로 등록한 유저에 대한 로그인을 진행, 유저화면으로 넘어가지는 것을 확인하였으며 내용 또한 등록했던 그대로 임을 확인했다.
6. 유저화면에서 나가기 버튼이 정상적으로 작동함을 확인했다.
7. 첫번째로 등록한 유저에 대한 로그인에 대해 비밀번호가 틀리게 시도, 에러 메세지가 출력됨을 확인했다. 이후 올바른 비밀번호를 치자 유저 화면으로 넘어가졌다.
8. 유저화면에서 아이디를 제외한 나머지 값들을 변경하고 나간 후 다시 로그인 해주었다. 유저화면에 정보들이 수정한 정보들로 바뀌었다.
9. 아이디 값을 바꾸는 것을 시도, 아이디 값을 바꾸자 화면에서 나갈 수 없어 졌으며 이에 맞는 toast 메시지가 출력됨을 확인했다. 원래 값으로 돌리고 나가기를 시도할 경우 올바르게 나가 짐을 확인했다.
10. Login 화면에서 admin계정으로의 접속을 시도, 관리자 화면에 이름/id/이메일 형식으로 등록/수정한 데이터들이 출력됨을 확인할 수 있었다.
11. 관리자 화면에서 radio버튼만 클릭했다. 화면이 재 시작되고 삭제는 진행되지 않았다. 이는 나갔다 다시 로그인해도 마찬가지이다.
12. radio버튼을 누르고 삭제 버튼을 눌렀다. 해당 데이터에 대한 list가 사라짐을 확인했다.
13. 삭제된 정보에 대한 login을 시도했다. 삭제가 되어 login이 안 됨을 확인했다.
14. 회원가입 화면에서 중복 확인을 통해 ID가 사용가능해짐을 확인했다.

15. 관리자 화면에서 처음 등록한 정보를 삭제하였다. 데이터가 없어도 관리자 화면이 잘 작동함을 확인했다.

해당 test에 대한 결과는 Youtube에 업로드하였습니다.(용량에 의해 사진을 첨부하지 못했습니다.)

유튜브링크: https://youtu.be/QahNadTcfv0

-고찰

해당 프로젝트는 안드로이드 스튜디오를 기반으로 자명하게 사용되는 로그인 시스템을 구현하고 이를 바탕으로 회원가입, 비밀번호 검색, 관리자 모드 등을 구현하는 것이었습니다. 이번 프로젝트의 경우 학기 커리큘럼으로 진행하는 프로젝트 중 가장 high level의 프로젝트 였습니다. 그 때문인지 기존에 low level위주로 작업하던 프로젝트에 비해 이질감이 들었습니다. 하지만 high level을 다루며 원래라면 한참 걸릴 구현을 단시간에 처리하며 high level의 장점을 확실하게 느낄 수 있었습니다. 또한 data base와의 연계는 확실히 프로그램을 구현함에 있어 보안 을 비롯한 다양한 면에서 이점이 많이 있음을 확인할 수 있었습니다. 결과적으로 새로운 경험을 통해 많은 것을 배울 수 있었습니다.

또한 이번 프로젝트의 경우 비교적 제안사항이 자유로운 편이었습니다.(예를 들어 radio버튼에 대하여 구체적인 처리 방안을 주지 않아 위 같은 처리가 가능했습니다.) 이러한 점은 구현할 때 스스로 설계할 필요가 있어 힘들었지만 역으로 자유도가 높아져 직접 설계를 해본다는 좋은 경험을 할 수 있었습니다.