모바일 프로그래밍 프로젝트

더 지니어스

[Team3]

김민정 김도은 박민서 오승준

INDEX

4 + 17 = ?

21 = ?

제한 시간 동안 카드를 뒤집어서 목표 값을 만들자!

더 지니어스

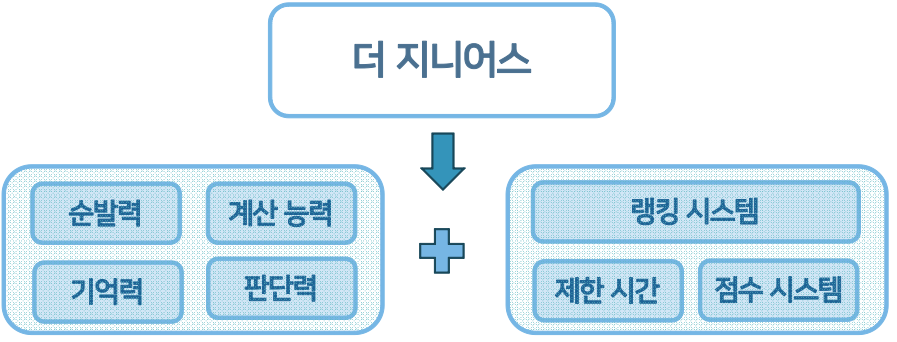
20초 동안 배치된 카드를 암기한다

3장의 가려진 카드를 뒤집는다

3장의 연산 결과가 목표 값과 일치하면 다음 라운드

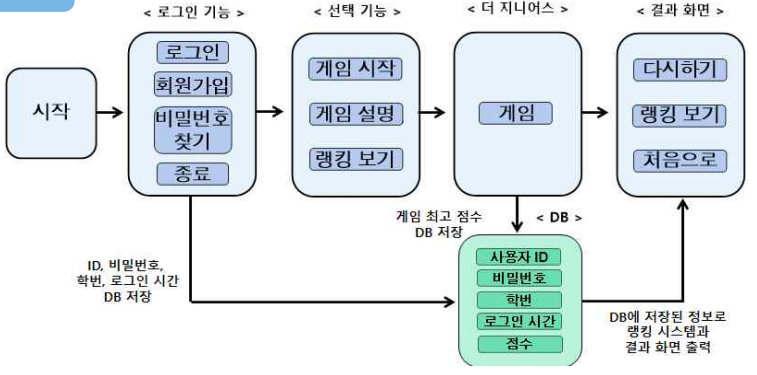
제한 시간 안에 성공하지 못하거나 5번을 틀리면 실패

제한 시간 동안 카드를 뒤집어서 목표 값을 만들자!



'추론 능력 향상'

App 기능 소개



로그인 시스템 | 게임 시스템 | 랭킹 시스템

App 페이지

더 지니어스

로딩 화면

로그인

회원가입

계정 찾기

선택 화면

게임 설명

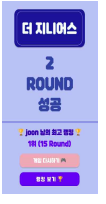
준비 화면

게임 화면1

게임 화면2

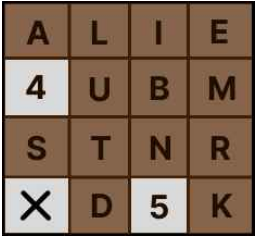
게임 화면3

결과 화면



랭킹 화면

활용 기술



번호	내용
1	자주 쓰이는 색, 메인 색 모듈화
2	어플 아이콘, 이름 변경
3	Loading 화면 추가
4	Handler를 활용한 시간 관리
5	실시간 정보가 반영되는 Custom Progress Bar
6	GridView를 기반한 Cell단위의 상호작용
7	조건에따른 Custom Grid Cell
8	Nested, Relative Layout
9	Frame Layout
10	intent의 finish()가 아닌 finishAffinity()
11	intent의 Flag (FLAG_ACTIVITY_CLEAR_TASK)
12	DB를 활용한 정보 관리
13	SHA-256을 활용한 DB에서의 비밀번호 hasing
14	Random과 Collection을 사용한 Shuffle
15	ListArray를 활용한 정보 관리
16	View를 활용한 적절한 정보 디자인

더 지니어스

역할 분담

아이콘 & 로딩 화면

테마 색

#31409F	#5A68C1	#795AC5	#B5BDF1

```
<color name="background">#B5BDF1</color>
<color name="fontcolor1">#31409F</color>
<color name="outline">#6B390F</color>
<color name="covercell">#D3BAA9</color>
<color name="realcell">#E4DDD6</color>
<color name="coverfont1">#5A68C1</color>
```

테마 폰트

KBO 다이아고딕체

제작 (사)한국야구위원회 조회수 151K 형태 고딕 굵기 3가지

던파 비트비트체

제작 (주)네오플 조회수 36.3K 형태 픽셀체

```
<font
    android:font="@font/kbo_dia_gothic_light"
    android:fontStyle="normal" />

<font android:fontStyle="normal"
    android:fontWeight="400"
    android:font="@font/dnf_bit" />
```

디자인 컨셉

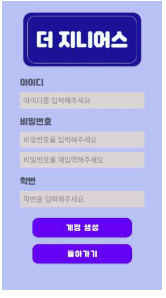
Manifest.xml에서 icon과 intent-filter의 'action', 'category'를 사용하여 아이콘과 로딩 화면을 구현

로그인 기능

로그인 & 회원 관리 액티비티

로그인

회원가입



비밀번호 찾기

비밀번호 찾기 (일치)

로그인 기능

입력 과정에서의 예외 처리

예외 발생 시, 토스트 메세지를 통해 알림

다양한 입력 예외 처리
아이디 미입력 상태
비밀번호 미입력 상태
아이디나 비밀번호에 공백이 존재하는 경우
(회원가입) 비밀번호 재입력 미입력 상태
(회원가입) 학번 미입력 상태
(회원가입) 학번이 정수가 아닌 경우
(회원가입) 비밀번호와 재입력의 불일치

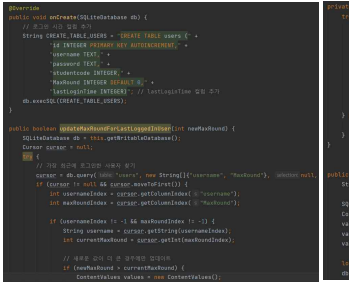
2

002 프로젝트 구현

로그인 기능

DB를 활용한 로그인 시스템

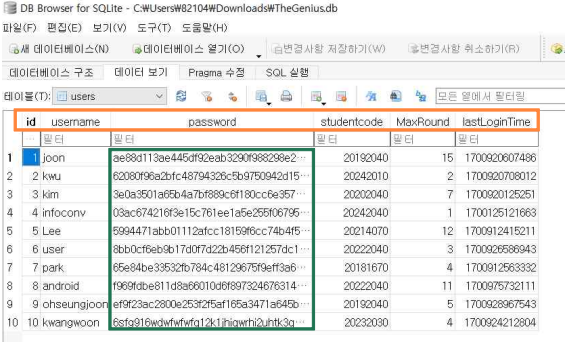
DB Class에 다양한 메소드를 구현 (SQLiteOpenHelper)



로그인 기능

DB를 활용한 로그인 시스템

DB Class에 다양한 메소드를 구현



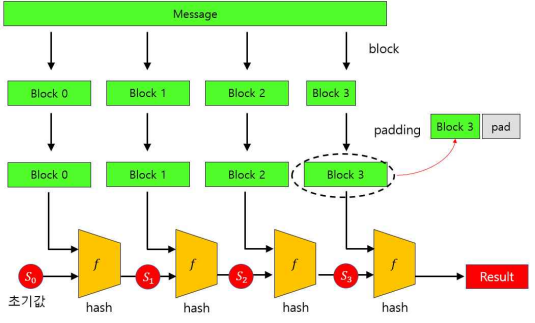
	id	username	password	studentcode	MaxRound	lastLoginTime
		필터	필터	필터	필터	필터
1	1	joon	ae88d113ae445df92eab3290f988288e2…	20192040	15	1700920607486
2	2	kwu	62080f96a2bfc48794326c5b9750942d15…	20242010	2	1700920708012
3	3	kim	3e0a3501a65b4a7bf889c6f180cc6e357…	20202040	7	1700920125251
4	4	infoconv	03ac674216f3e15c761ee1a5e255f06795…	20242040	1	1700125121663
5	5	Lee	5994471abb01112afcc18159f6cc74b4f5…	20214070	12	1700912415211
6	6	user	8bb0cf6eb9b17d0f7d22b456f121257dc1…	20222040	3	1700926586943
7	7	park	65e84be33532fb784c48129675f9eff3a6…	20181670	4	1700912563332
8	8	android	f969fdbe811d8a66010d6f897324676314…	20222040	11	1700975732111
9	9	ohseungjoon	ef9f23ac2800e253f2f5af165a3471a645b…	20192040	5	1700928967543
10	10	kwangwoon	6sfg916wdwfwfwfg12k1ihiqwrhi2uhtk3g…	20232030	4	1700924212804

DB에 추가된 기능
생성자
User 추가 기능
비밀번호 Hashing / 변경
사용자 검색 기능(다조건)
마지막 로그인 유저 점수 갱신
마지막 로그인 유저 최고 랭킹
Top 10 유저 리스트

로그인 기능

‘SHA-256’ 해시 암호화

비밀번호는 해시 암호화되어 DB에 저장



- 문자열을 0과 1로 변환 (512bits 블럭들로 변환)

- 이 블럭에 대해 해쉬값을 구하고 이 해쉬값과 이전 블럭을 잘라서 붙인 다음 다시 해쉬값을 붙인다.

- 이 과정을 64번 반복하여 256bits의 해쉬값으로 만든다.

DB에 저장된 값이 유출되어도
원본 Password를 알기 어렵다! (보안성 강화)

더 지니어스

선택 화면과 게임 설명

- 게임 설명 -

- 게임이 시작되면 20초 동안 4x4 칸에 무작위로 배치된 숫자와 기호가 제시된다.

- 20초 이후엔 제시된 값들이 모두 가려지며, 1분 동안 목표 값과 연산 결과가 일치하는 3개 카드를 찾아야 한다.
(5라운드는 50초, 10라운드는 40초, 15라운드 이상은 30초)

숫자는 1~12 사이의 수만 존재하며, 기호는 +-*/가 존재한다.

연산 결과가 목표 값과 다르면 하트가 하나씩 소모되며, 하트가 0개가 되거나 시간을 초과하면 게임이 종료된다.

준비 화면

준비 화면

더 지니어스

FrameLayout을 활용하여 MainActivity에 해당하는 Layout 위에 겹쳐서 준비 화면 Layout이 보여지도록 한다.

```
<FrameLayout
    android:id="@+id/overlayLayout"
    android:layout_width="420dp"
    android:layout_height="790dp"
    android:layout_marginTop="0dp"
    android:background="#B3000000"
    android:visibility="visible">
    <TextView
        android:id="@+id/currentRound"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/dnf_bit"
        android:layout_gravity="center"
        android:textColor="@color/white"
        android:text="N Round"
        android:layout_centerHorizontal = "true"
        android:textColorLink="#5C3A3A"
        android:layout_marginBottom="55dp"
        android:textSize="50dp"
        />
```

암기 화면

더 지니어스

암기 화면


```
private void initializeGame() {
    ProgressBar progressBar = findViewById(R.id.progressBar);
    TextView left_time = findViewById(R.id.leftTime);
    TextView Title = findViewById(R.id.Title_txt);
    selectedValues = new ArrayList<>();

    final Integer Target_num = generateValidResult();

    // Round와 Life 상태 업데이트
    updateGameStatus();

    // Real GridView 설정
    gridView_Real = findViewById(R.id.Real_gridView);

    // 20초 동안 진행 상태 업데이트
    for (int i = 0; i <= 20; i++) {
        int progress = 20 - i;
        int secondsLeft = i;

        handler1.postDelayed(() -> {
            progressBar.setProgress(progress);

            // 남은 시간을 TextView에 표시
            left_time.setText("남은 시간 : " + String.valueOf(20 - secondsLeft) + "초");
        }, delayMillis: i * 1000);
    }

    // 20초 후에 Real GridView와 ProgressBar 숨김
    handler1.postDelayed(() -> {
        gridView_Cover.setVisibility(View.VISIBLE);
        Title.setText("목표 값 : " + String.valueOf(Target_num));
```

- Round는 이전의 intent로부터 전달받아 갱신

- 4x4의 GridView를 만들고,
각 Cell에 1~12, +-*/를 랜덤으로 할당한 뒤 출력

- Handler를 사용하여 반복문이 한 번 돌 때 마다
1초씩 Delay 시키면서 Progress Bar를
Update한다.

선택 화면

선택 화면

더 지니어스

- 주어진 값들 내에서 랜덤하게 목표 값을 설정한다.

- Progress Bar는 앞선 과정과 동일하게
Handler를 활용하여 갱신 및 시간을 측정하고,
Time Over가 되면 Life를 0으로 설정한다.

- 앞선 GridView와 동일하되, 알파벳이 적힌 GridView를 하나 더
만든 다음 RelativeLayout을 사용하여 두 Grid를 겹치게 배치한다.

- 두 GridView는 index가 공유되므로 이를 활용하여, 실제 값의 Datatype에
따라 Cover Grid의 글씨 색을 다르게 설정한다.

선택 화면

선택 화면

더 지니어스

- 사용자가 Cover Grid Cell을 클릭하면 해당 Grid Cell을 투명하게 바꾸어 아래의 Cell이 보이도록 설정한다. (선택된 Cell 아래의 값은 List에 저장)

- 선택된 Cell이 3개가 되면 사칙연산을 수행하고 결과를 비교한다. (Cover Grid Cell 역시 원상복구한다.)

- 만약 목표 값과 연산 결과가 다르다면 Life를 하나 감소시키고 이에 따라 UI를 갱신한다. (Life가 0이 되면 현 Round를 ResultActivity로 전달)

- 만약 목표 값이 연산 결과와 같다면 Round를 하나 증가시키고, 이를 Intent로 넘기면서 현재 액티비티를 재시작한다. (액티비티의 재귀적 구성)

- 액티비티를 재시작하는 경우에 handler는 액티비티가 아닌, Thread 단위로 동작하기에 removeCallbacksAndMessages(null); 를 사용하여 Thread에서 Handler를 내리고, 재귀적인 액티비티를 사용하기에 intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);를 사용하여 초기화한다.

선택 화면

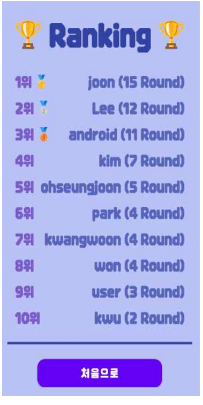
이외 코드들은 Q&A 시간을 활용해주시면 감사하겠습니다.

```
gridView_Cover.setOnItemClickListener((parent, view, position, id) -> {
    if (!isClickable) {
        return; // 클릭이 비활성화된 경우 처리하지 않음
    }

    if (adapter.getSelectedCount() == 0) {
        selectedValues.clear(); // 선택된 셀이 없을 때 리스트를 초기화
    }

    if (adapter.selectedCells.contains(position)) {
        return; // 이미 선택된 셀은 경우 추가적인 동작을 수행하지 않음
    }

    adapter.toggleCellState(position); // 셀 상태 변경
    adapter.notifyDataSetChanged(); // 데이터 변경 알림

    if (adapter.getSelectedCount() == 0) {
        selectedValues.clear(); // 선택된 셀이 없을 때 리스트를 초기화
    }

    if (adapter.getSelectedCount() == 3) {
        isClickable = false;
        selectedValues.clear();
        for (int pos : adapter.clickedPositions) {
            String realGridValue = (String) realAdapter.getItem(pos);
            selectedValues.add(realGridValue);
        }

        // 마지막 3개 원소만 유지
        if (selectedValues.size() > 3) {
            selectedValues = selectedValues.subList(fi selectedValues.size() - 3, selectedValues.size());
```

```
private void restartGame() {
    Intent intent = new Intent( packageContext: MainActivity.this, MainActivity.class);
    intent.putExtra( name: "Round", Round); // Round 값을 Intent에 추가
    intent.putExtra( name: "MaxTime", Max_time); // Maxtime 값을 Intent에 추가
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent); // 새로운 Activity 시작
}

private void checkGameStatus(int operationResult,int Target_num) {
    if (operationResult == Target_num) {
        Round++;
        handler1.removeCallbacksAndMessages( token: null);
        playtime.removeCallbacksAndMessages( token: null);
        finish();
        restartGame(); // 새로운 게임 시작
    } else {
        Life--;
        if (Life < 1) {
            handler1.removeCallbacksAndMessages( token: null);
            playtime.removeCallbacksAndMessages( token: null);
            // Life가 0이 되면 다음 Intent로 넘어감
            Intent nextIntent = new Intent( packageContext: MainActivity.this, ResultActivity.class);
            nextIntent.putExtra( name: "MaxRound", Round);
            finish(); // 현재 Activity 종료
            startActivity(nextIntent);
        } else {
            updateGameStatus();
        }
    }
```

더 지니어스

결과 화면과 랭킹 시스템



- 결과 화면 -

- MainActivity에서 최종적으로 전달받은 Round를 바탕으로 현재 Round를 출력

- 전달받은 Round가 DB에 저장된 Round보다 큰 경우 DB를 갱신

- DB에서 마지막으로 로그인 한 user의 username과 최고 기록을 가져옴

- 랭킹 화면 -

- DB에서 MaxRound를 기준으로 Top 10 User에 대한 정보 (Username, MaxRound)를 ArrayList로 가져온 다음 각 등수에 맞게 SetText

예상 기대 효과 및 활용 분야

프로젝트 단위

- 두뇌 활동과 문제 해결 능력 강화에 도움
- 재미있는 사용자 경험
- 플래그쉽 스토어, 팝업 스토어에서의 게임 이벤트

사용 기술 단위

- 사용자에게 준비 시간을 주어야 하는 경우
  FrameLayout
- 복잡한 Activity의 실행 결과가 단순할 경우
  재귀적 Activity 구성
- Grid View를 겹쳐서 Index를 공유하는 알고리즘
- Handler를 사용한 타임어택 UI 갱신
  (메인 스레드에서의 갱신)

004 시연 영상


4:40
Tue, Nov 23
Team3
더 지니어스


4:49
더 지니어스
A
L
Click
4
U
B
M
S
T
N
R
D
5
K
게임 시작 🎮
랭킹 보기 🏆
게임 끝내기

Q&A

Thank you

Appendix

A

Appendix

로그인 기능

DB를 활용한 로그인 시스템 구현

생성자, 가장 최근 로그인한 사람의 MaxRound Update

```
@Override
public void onCreate(SQLiteDatabase db) {
    // 로그인 시간 컬럼 추가
    String CREATE_TABLE_USERS = "CREATE TABLE users (" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "username TEXT," +
            "password TEXT," +
            "studentcode INTEGER," +
            "MaxRound INTEGER DEFAULT 0," +
            "lastLoginTime INTEGER)"; // lastLoginTime 컬럼 추가
    db.execSQL(CREATE_TABLE_USERS);
}

public boolean updateMaxRoundForLastLoggedInUser(int newMaxRound) {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = null;
    try {
        // 가장 최근에 로그인한 사용자 찾기
        cursor = db.query( table: "users", new String[]{"username", "MaxRound"},  selection: null,
        if (cursor != null && cursor.moveToFirst()) {
            int usernameIndex = cursor.getColumnIndex( s: "username");
            int maxRoundIndex = cursor.getColumnIndex( s: "MaxRound");

            if (usernameIndex != -1 && maxRoundIndex != -1) {
                String username = cursor.getString(usernameIndex);
                int currentMaxRound = cursor.getInt(maxRoundIndex);

                // 새로운 값이 더 큰 경우에만 업데이트
                if (newMaxRound > currentMaxRound) {
                    ContentValues values = new ContentValues();
```

Hash 암호화, User 추가

```
private String hashPassword(String password) {
    try {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(password.getBytes());
        byte[] bytes = md.digest();

        StringBuilder sb = new StringBuilder();
        for (byte aByte : bytes) {
            sb.append(Integer.toString( i: (aByte & 0xff) + 0x100,  radix: 16).substring(1));
        }
        return sb.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException("비밀번호 해시 중 오류 발생", e);
    }
}


public boolean addUser(String username, String password, int studentcode) {
    String hashedPassword = hashPassword(password);

    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("username", username);
    values.put("password", hashedPassword);
    values.put("studentcode", studentcode);

    long result = db.insert( table: "users",  nullColumnHack: null, values);
    db.close();
```

A

Appendix

로그인 기능

DB를 활용한 로그인 시스템 구현

DB에 존재하는 User인지 확인 (ID, Password)

```
public boolean checkUser(String username, String password) {
    String hashedPassword = hashPassword(password);
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query( table: "users",
            new String[]{"id"},
             selection: "username=? AND password=?",
            new String[]{username, hashedPassword},
             groupBy: null,  having: null,  orderBy: null);

    boolean loginSuccessful = cursor.getCount() > 0;
    cursor.close();

    if (loginSuccessful) {
        // 로그인 성공 시 로그인 시간 업데이트
        long currentTime = System.currentTimeMillis();
        ContentValues values = new ContentValues();
        values.put("lastLoginTime", currentTime);
        db.update( table: "users", values,  whereClause: "username=?", new String[]{username});
    }

    db.close();
    return loginSuccessful;
}
```

User 확인 (ID, 학번), Password Update

```
public boolean checkUser_find(String username, Integer studentcode) {
    SQLiteDatabase db = this.getReadableDatabase();

    // Integer를 String으로 변환
    String studentCodeStr = Integer.toString(studentcode);

    Cursor cursor = db.query( table: "users",
            new String[]{"id"},
             selection: "username=? AND studentcode=?",
            new String[]{username, studentCodeStr},
             groupBy: null,  having: null,  orderBy: null);

    int cursorCount = cursor.getCount();
    cursor.close();
    db.close();

    return cursorCount > 0;
}

public boolean updatePassword(String username, String newPassword) {
    String hashedNewPassword = hashPassword(newPassword);
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("password", hashedNewPassword); // 중복된 해시 처리 수정

    // 사용자명을 기준으로 비밀번호 업데이트
    int result = db.update( table: "users", values,  whereClause: "username=?", new String[]{username});
    db.close();
```

A

Appendix

로그인 기능

DB를 활용한 로그인 시스템 구현

MaxRound 기준 Top10 User 정보

```
public List<Pair<String, Integer>> getTopTenMaxRoundUsers() {
    List<Pair<String, Integer>> topUsers = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = null;
    try {
        // MaxRound를 기준으로 정렬하여 상위 10개의 데이터 조회
        cursor = db.query( table: "users", new String[]{"username", "MaxRound"},  selection: null,  selectionArgs: null,  groupBy: null,
        while (cursor != null && cursor.moveToNext()) {
            String username = cursor.getString(cursor.getColumnIndex( s: "username"));
            int maxRound = cursor.getInt(cursor.getColumnIndex( s: "MaxRound"));
            topUsers.add(new Pair<>(username, maxRound));
        }
    } finally {
        if (cursor != null) {
            cursor.close();
        }
        db.close();
    }
    return topUsers;
}
```

로그인중인 User의 최고 랭킹과 Round

```
public Pair<Integer, Pair<String, Integer>> getMostRecentUserRank() {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = null;
    try {
        // 가장 최근에 로그인한 사용자 찾기
        cursor = db.query( table: "users", new String[]{"username", "MaxRound"},  selection: null,  selectionArgs: null,
        if (cursor != null && cursor.moveToFirst()) {
            String recentUsername = cursor.getString(cursor.getColumnIndex( s: "username"));
            int recentMaxRound = cursor.getInt(cursor.getColumnIndex( s: "MaxRound"));

            // 전체 사용자를 MaxRound 기준으로 정렬하여 등수 계산
            try (Cursor allUsersCursor = db.query( table: "users", new String[]{"username"},  selection: null,  selecti
                int rank = 1;
                while (allUsersCursor.moveToNext()) {
                    String username = allUsersCursor.getString(allUsersCursor.getColumnIndex( s: "username"));
                    if (recentUsername.equals(username)) {
                        return new Pair<>(rank, new Pair<>(recentUsername, recentMaxRound));
                    }
                    rank++;
                }
            }
        }
    } finally {
        if (cursor != null) {
            cursor.close();
        }
        db.close();
    }
    return null; // 가장 최근에 로그인한 사용자가 없거나 찾을 수 없는 경우
}
```