

종합설계 프로젝트2

최종 결과 보고서

운동량 관리 어플리케이션

B311167 이정문

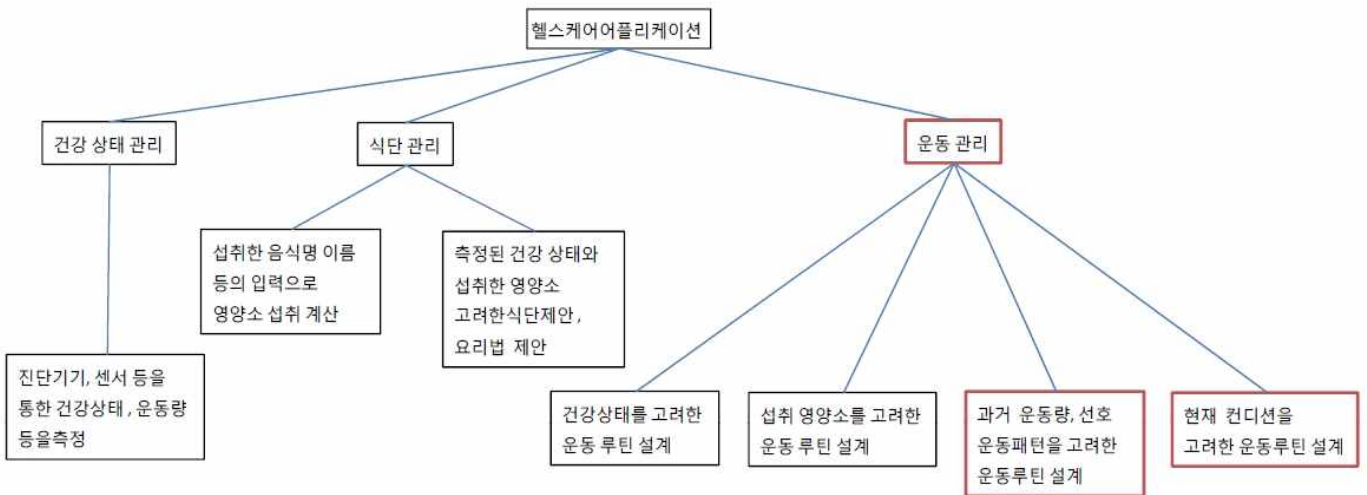
2016. 12. 14

목 차

1. 프로젝트 주제
2. 주제의 선정 이유와 타당성 분석
3. 프로젝트 결과물의 활용 방안과 기대효과
4. 전체 시스템 구성도
5. 제공되는 서비스의 내용과 서비스별 프로세스 다이어그램
6. 구현결과
7. 핵심 모듈의 소스 프로그램

1. 프로젝트 주제

① 프로젝트 주제 : 종합 헬스 케어 어플리케이션 (운동량 조절 구현) : 건강운동 도우미



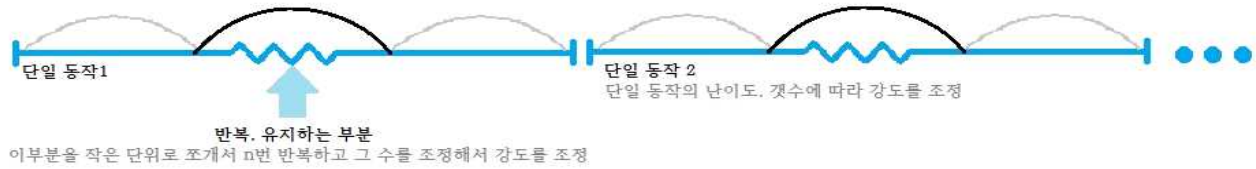
이 프로그램은 종합적인 헬스케어 어플리케이션으로 첫 번째 부분인 건강상태 관리는 웨어러블 디바이스로 매일 혈압, 혈당, 체중 등을 측정하여 사용자가 알 수 있게 해주고, 하루 동안의 운동량도 측정해 주어 항상 자신의 건강상태를 살필 수 있게 합니다. 하지만 이 기능은 현재 많은 소프트웨어와, 기기들이 개발되어 이번 프로젝트와 어울리지 않다고 생각되었습니다.

또한 식단관리 부분은 섭취한 음식은 간단히 입력함으로써 칼로리뿐만이 아니라 여러 가지 영양소들을 간단히 분석해 줍니다. 분석 후에는 하루 또는 이틀, 혹은 일주일 단위로 섭취량 중 부족한 영양소나 과다한 영양소가 무엇인지 경고를 주고, 이를 고려해서 식단과 요리법을 제안해 주도록 하였습니다. 이 기능은 앞선 건강 측정도 고려되어 식단과 요리법을 추천해 준다면 좀 더 유용해 질 것이라고 생각되었습니다. 하지만 이 부분은 단순히 텍스트 입력을 하기에는 사용자에게 오히려 불편을 줄 것 이라 생각되 구현하게 된다면 음성인식 등을 사용하는 편이 더 좋을 것이라 생각되었고, 이는 이번 프로젝트로 구현하기에는 너무 많은 내용이 될 것이라 생각했습니다.

마지막으로 이번에 구현할 부분인 운동량의 관리입니다. 하루 30분의 규칙적인 운동은 심폐기능을 개선시키고 근육을 단단하게 하는 등 우리 몸을 건강하게 하는 효과가 있습니다. 하지만 고혈압, 뇌졸중 같은 만성질환을 가지고 있는 사람의 운동이나 적절한 양의 휴식시간을 가지지 않고 하는 무리한 운동이나, 기온이 높거나 낮은 날의 많은 양의 운동은 도리어 해가 될 수 있습니다. 이렇게 본인에게 맞지 않는 양의 운동으로 인해 병을 얻게 되는 경우도 주변에 흔히 있는 경우입니다. 또한 전문가들은 운동의 효과를 보기 위해서는 꼭 운동의 목적을 가지고 운동량을 정해 규칙적으로 하되 단계적으로 운동량을 증가시켜서 하라고 합니다. 이렇게 운동을 건강하게 하기 위해서 고려해야 할 사항들은 많습니다. 이런 문제들을 해결하기 위해서 어플리케이션을 개발하게 되었습니다.

이 운동관리 부분은 개인의 여러 가지 상황을 고려하여 사용자만의 운동 동작들을 조합해 운동량에 맞게 그대로 따라 할 수 있게 도와주는 프로그램입니다. 구현계획의 초기와 달라진 점은 개인에 맞는 운동 동작에 초점을 맞추기 보다는 개인의 컨디션과 환경적 요인을 고려하여 운동량을 조절해 주는 것에 초점을 맞추었다는 점입니다. 그 이유는 운동의 동작의 추천은 알고리즘은 간단 할 수 있는데 비해서 운동학적인 지식이 더 많은 부분이 필요할 것이라고 생각되어 이 프로젝트의 목표와는 다소 멀게 느껴졌기 때문입니다. 그렇기 때문에 운동 동작의 선택은 다소 느슨하지만 운동량의 조절을 중심으로 한 기능은 4가지입니다.



- i. 개인을 위한 운동 동작 프로그래밍 : 선택된 주의해야 할 신체부위에 따라 운동 프로그래밍
- ii. 개인을 위한 운동 동작 프로그래밍2 : 사용자의 연령에 알맞은 강도의 운동 동작 선택
- iii. 운동 강도 조절 : 사용자가 선택한 주기에 맞게 각 운동 동작의 유지, 반복 시간을 증가시킴
- iv. 환경적 요소 고려 : 연령, 기온에 따라 자동으로 강도 증감
- v. 운동 기록 : 최근 10회간의 운동 시간 기록



강도의 조절은 아래의 그림처럼 단일 동작 의 반복, 유지 시간과 단일 동작들의 개수로 조정됩니다.

3. 주제의 선정이유와 타당성 분석

① 기존 시스템과 비교 - 기존 휘트니스 어플리케이션

Women Workout 	Sworakit 
<p>많은 운동동작의 동영상을 5분에서 60분 사이를 선택하여 운동루틴을 제공합니다. 유료 버전으로 동작 난이도 단계를 1~3까지 제공하고, 무료버전은 Free style 운동만을 제공합니다.</p> <pre> graph TD A[프로그램 시작] --> B[운동시간, 부위, 단계의 선택] B --> C[휴식시간 선택] C --> D[영상 재생 (skip을 누를 경우 다음 동작을 재생시킵니다)] D --> E[선택한시간이 종료되면 영상재생 종료] </pre>	<p>운동할 시간, 운동의 종류, 강화하고 싶은 운동의 부위를 선택할 수 있고, 사용자가 직접 운동순서를 제작해 개인에 맞춰 운동을 할 수 있습니다. 운동의 순서를 랜덤으로 바꿔줍니다.</p> <pre> graph TD A[프로그램 시작] --> B[빠른 5분운동 선택] A --> C[운동부위 선택] A --> D[맞춤식 운동 선택] B --> E[영상 재생 (전동작, 다음 동작으로 건너뛸 수 있다)] C --> F[운동 시간 선택] F --> E D --> G[원하는 동작들 선택] G --> E E --> H[선택한 시간 종료 또는 운동 동작들 종료 되면 재생 종료] </pre>

이 두 운동 어플리케이션에 비해 계획 중인 어플리케이션은 크게 4가지의 강점이 있습니다.

첫 번째로 이 어플리케이션들은 모두 운동 시간을 사용자가 실행할 때마다 선택하며, 운동시간에 대한 기록 기능이 없습니다. 이와 달리 건강운동도우미는 과거의 운동시간을 기록하고, 이에 따라 운동시간을 적절한 양을 자동으로 늘이거나 줄여줍니다. 운동량의 증가는 증가하는 주기를 사용자가 선택할 수 있게 해주었습니다. 그 후에 필요하다면 사용자가 운동 시간을 증가 혹은 감소시킬 수 있도록 조정할 수 있게 해주었습니다. 이렇게 함으로서 사용자는 과거의 운동량을 일일이 기록하거나 기억하지 않아도 쉽게 운동량을 규칙적으로 늘릴 수 있으며, 개인 사정에 따라 운동시간을 조정할 수도 있게 되어 효과적인 운동을 도와줍니다.

두 번째로 기존 어플리케이션들과 달리 건강 운동 도우미는 사용자의 연령대와 날씨(기온)에 맞게 강도(시간)를 조정해줍니다. 이로써 사용자들은 무리한 운동을 하여 운동의 부작용을 겪는 일을 줄일 수 있습니다. 먼저 모든 연령대에 대해서 기온이 적정치보다 낮은 날(추운 날)에 대해서 준비 운동 시간을 늘려 운동루틴을 짜주며, 연령대에 따라 40대 이상일 경우 기온이 적정치인 섭씨 20도 이상이거나, 섭씨 5도 이상이라면, 운동 시간을 줄여 프로그래밍 해줍니다. 또한 습도에 대한 고려도 사용자의 선택에 따라 추가할 계획입니다.

세 번째로, 건강 운동 도우미는 운동들의 선택에 있어서 불편한 특정 부위를 선택할 시에 이 부분에 무리가 되지 않는 운동들로 운동 루틴을 짜주는 기능이 있습니다. 물론 기존 프로그램이 자신에 상태에 맞게 운동 프로그램을 사용자에게 짤 수 있게 제공하는 기능이 있으나, 이 기능을 사용하는 사용자가 운동학적으로 모든 동작의 기능을 이해해야만 이런 기능을 사용할 수 있어 현실적으로 이런 고려가 불가능합니다. 이에 대해 사용자에게 무리가 가지 않았으면 하는 부위를 선택받고 이를 자동적으로 배재해 주는 기능을 추가합니다.

네 번째로, 건강 운동 도우미는 운동 동작별로 운동 난이도를 가지고 있어, 나이에 알맞은 운동 동작을 자동으로 선택해 주어, 사용자에게 더욱 세분화 된 맞춤 운동 서비스를 제공합니다. 위의 세 번째 강점과 동일하게 기존 어플리케이션들은 사용자가 운동 동작을 직접 고를 수는 있지만, 운동 동작에 대한 정보를 가진 사용자들이 아닐 경우 건강한 운동을 하기는 어렵습니다. 이 점을 감안해 본 어플리케이션은 운동동작의 난이도를 4가지로 구분하고, 입력받은 사용자의 나이를 토대로 상위 난이도의 동작들을 제외해 줍니다.

하지만 이 프로그램의 약점은 운동량의 측정은 시간과 운동의 강도 (운동시의 심박수)를 곱한값입니다. 이렇게 측정된 값을 운동량으로 기록하며, 이 운동량을 매일 증가시키고 날에 따라 조정해 주어야 운동학적인 충분한 효과를 기대할 수 있습니다. 하지만 운동시의 심박수의 측정은 이 프로그램의 환경에서 구현하지 못하였으므로, 운동량을 운동의 시간에 기반에서 조정하고 있습니다. 이는 전체 프로그램에서 웨어러블 디바이스와 결합되었을 때, 다른 헬스케어 프로그램과 비교해 큰 강점이 될 것이라고 생각합니다.

② 주제선정 이유 및 중요성

이 주제의 선정이유는 평소에 항상 헬스케어나 건강에 관해 관심이 많았습니다. 그리고 그래서 저번 학기에 수강한 보건학 수업에서 꾸준한 운동의 필요성과 꾸준히 강도를 높여야 할 필요성을 깨닫게 되었습니다. 하지만 현실적으로 이렇게 규칙적으로 운동하는 것이 쉽지는 않다는 생각이 들었고 운동루틴을 자동으로 만들어주는 프로그램이 있다면 운동을 배우는 비용을 절약할 수 있고, 시간과 공간의 제약을 받지 않고 꾸준히 운동을 할 수 있게 될 수 있을 것이라는 생각이 들었습니다.

현재 BT·IT의 발전과 융합이 확대되고, 의료 패러다임이 질병 치료에서 예방과 관리로 전환됨에 따라 경쟁이 치열한 모바일 시장에서도 개인 맞춤형 ‘스마트 헬스케어 산업’이 급부상 하고 있습니다. 2015년 3월 2일 개최된 모바일 월드 콩그레스(MWC) 2015에서 전세계 모바일 기업들은 스마트 헬스케어를 주목하고 있고, 미국 정부는 원격의료 관련 광대역 서비스 보조금 등 예산으로 72억 달러 투자를 발표하고 디지털의료서비스에 대한 의료수가 체계를 적극적으로 마련하고 있으며 쿠팡, 구글 등 IT기업들이 디지털병원 설립 등 스마트헬스케어사업을 적극 추진 중인 등 2017년 스마트헬스케어 세계시장은 260억 달러 이상 성장 전망 되어지는 유망 사업입니다.

또한 치료의학 시대에서 예방의학 시대로 이행하는 전세계적인 추세에 따라 진단의학에 대한 수요는 크게 늘고 있으며, 글로벌 고령화 사이클과 맞물려 체외진단 시장의 급성장 또한 예상되고 있습니다. 최근의 헬스케어 트렌드는 단순히 오래 사는 것이 아니라 건강하게 오래 사는 것, 즉 삶의 질을 추구하고 있습니다. 21세기 헬스케어 패러다임은 과거 공중 보건의 시대를 거쳐 질병 치료의 시대에서 질병 예방과 관리를 통한 건강 수명 연장의 시대로 변모하고 있습니다. 현재 헬스케어의 개념은 병원 치료 중심에서 예방 및 건강관리 중심으로 발전하는 일상 관리화의 시대로써 의료산업과 건강관리 산업을 포괄하는 전 생애에 걸친 라이프 케어의 개념이라 할 수 있습니다.

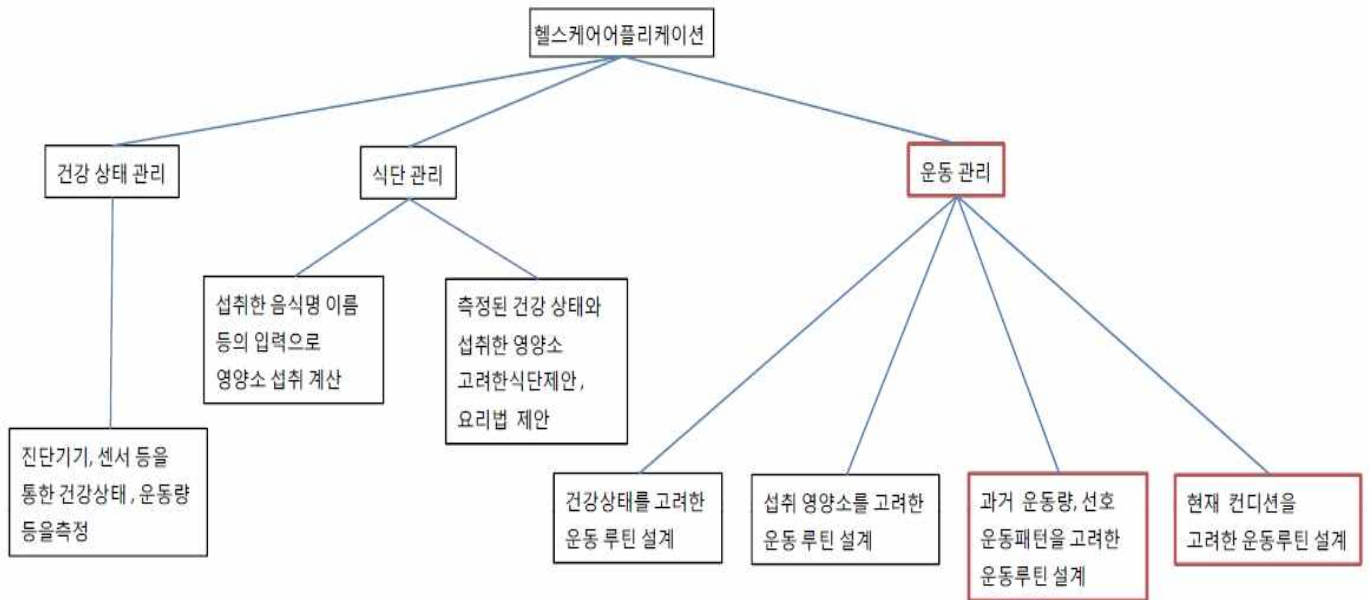
3. 프로젝트 결과물의 활용방안과 기대효과

이 프로젝트의 나머지 구현 하지 않을 부분인 건강 상태와 식단의 관리 프로그램이 완성되면 추가 된다면 단순한 체력관리나 체중관리 뿐만이 아니라 당뇨병, 고혈압, 비만, 고지혈증, 골다공증 등 여러 만성질환들을 예방, 관리하여 건강한 삶에 도움을 줄 수 있을 것이라고 기대합니다. 그렇게 된다면 현재 운동처방이나 개인의 종합적인 관리를 위해서 개인 트레이너나 전문가를 통해 관리하는 것에 비해 경제적인 해결책을 제시해 줄 수 있을 것이라고 생각합니다. 특히 운동량의 측정에 있어서 시간뿐만이 아니라 웨어러블 디바이스를 통한 심박수도 측정한다면 좀 더 정확한 운동량 관리를 도와줄 수 있어 충분한 건강관리 효과를 누릴 수 있을 것입니다.

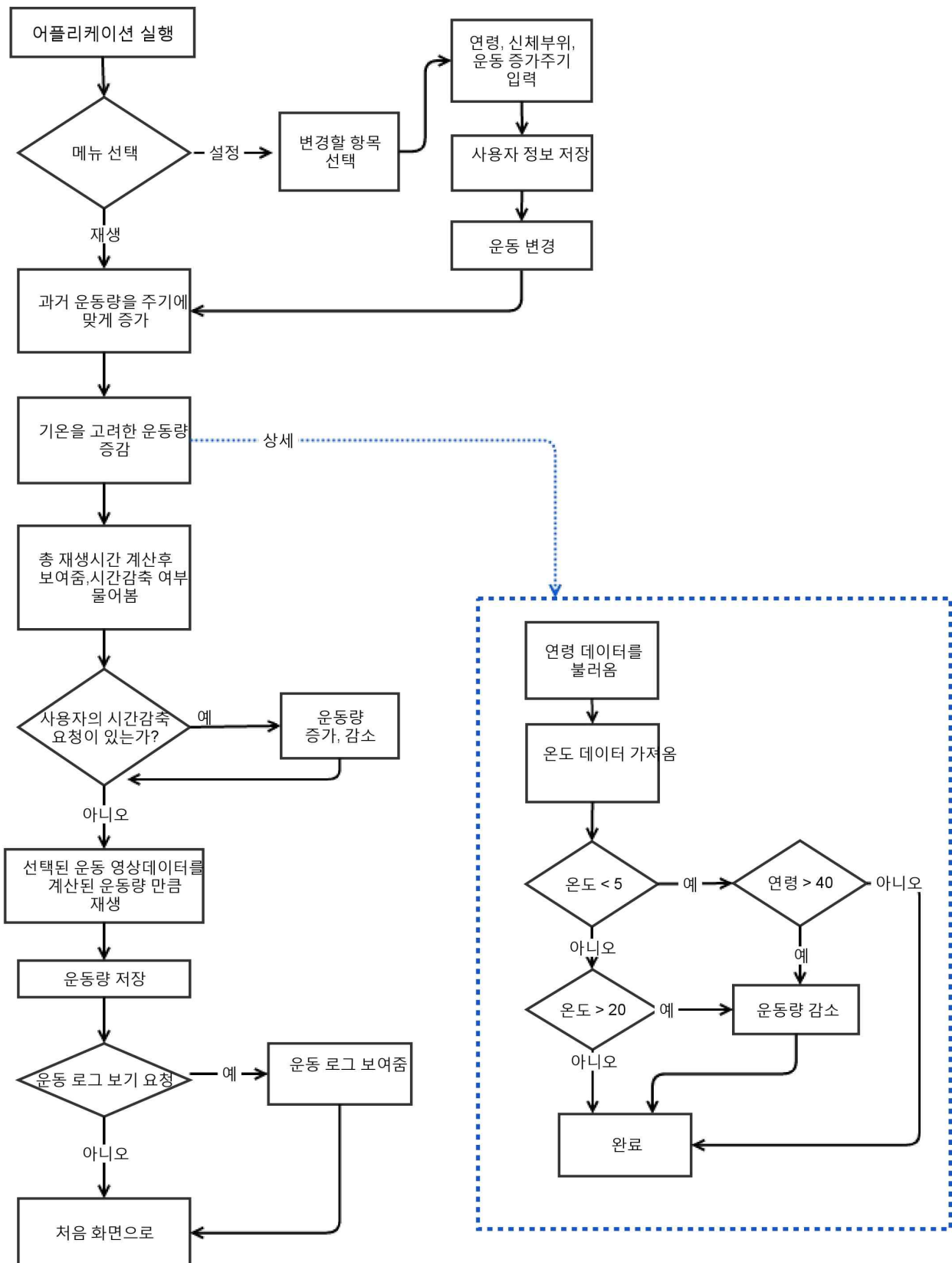
하지만 운동 관리부분만으로도 건강하고 안전한 규칙적인 운동을 돕는데 효과적일 것이라고 기대하며, 병원이나 피트니스 센터에서 응용되어 환자의 재활치료나 운동관리, 회원의 운동관리에 있어서 쓰일 수 있을 것입니다.

4. 전체 시스템 구성도

① 시스템 전체 다이어그램

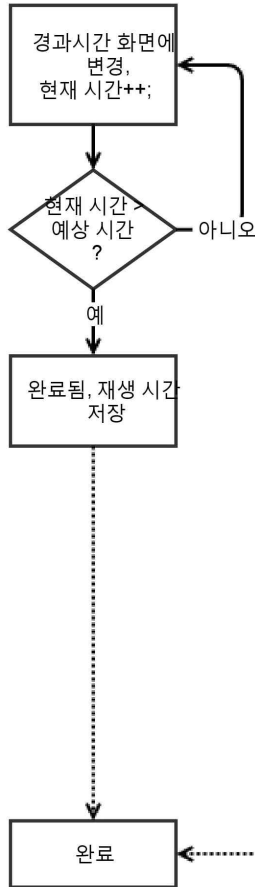


② 구현한 부분의 전체 플로우 차트

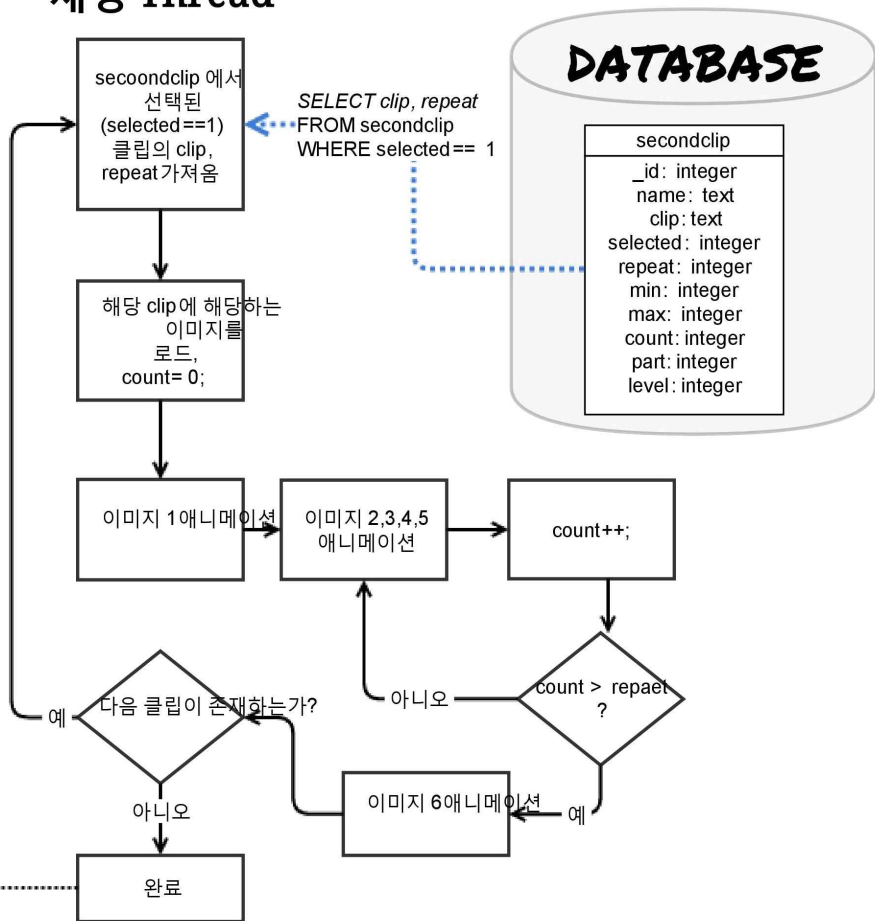


③ 재생 부분 플로우 차트

타이머 Thread

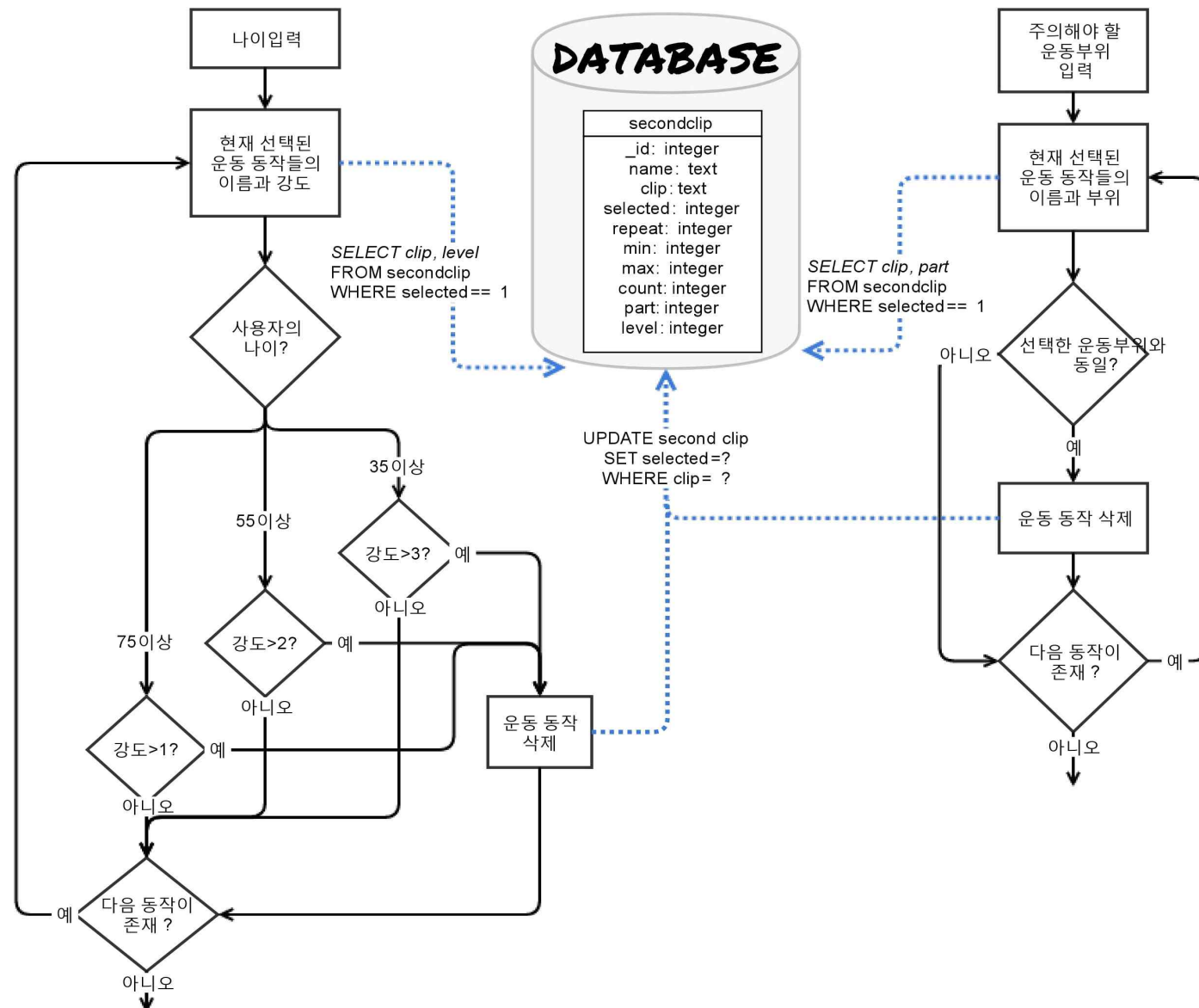


재생 Thread

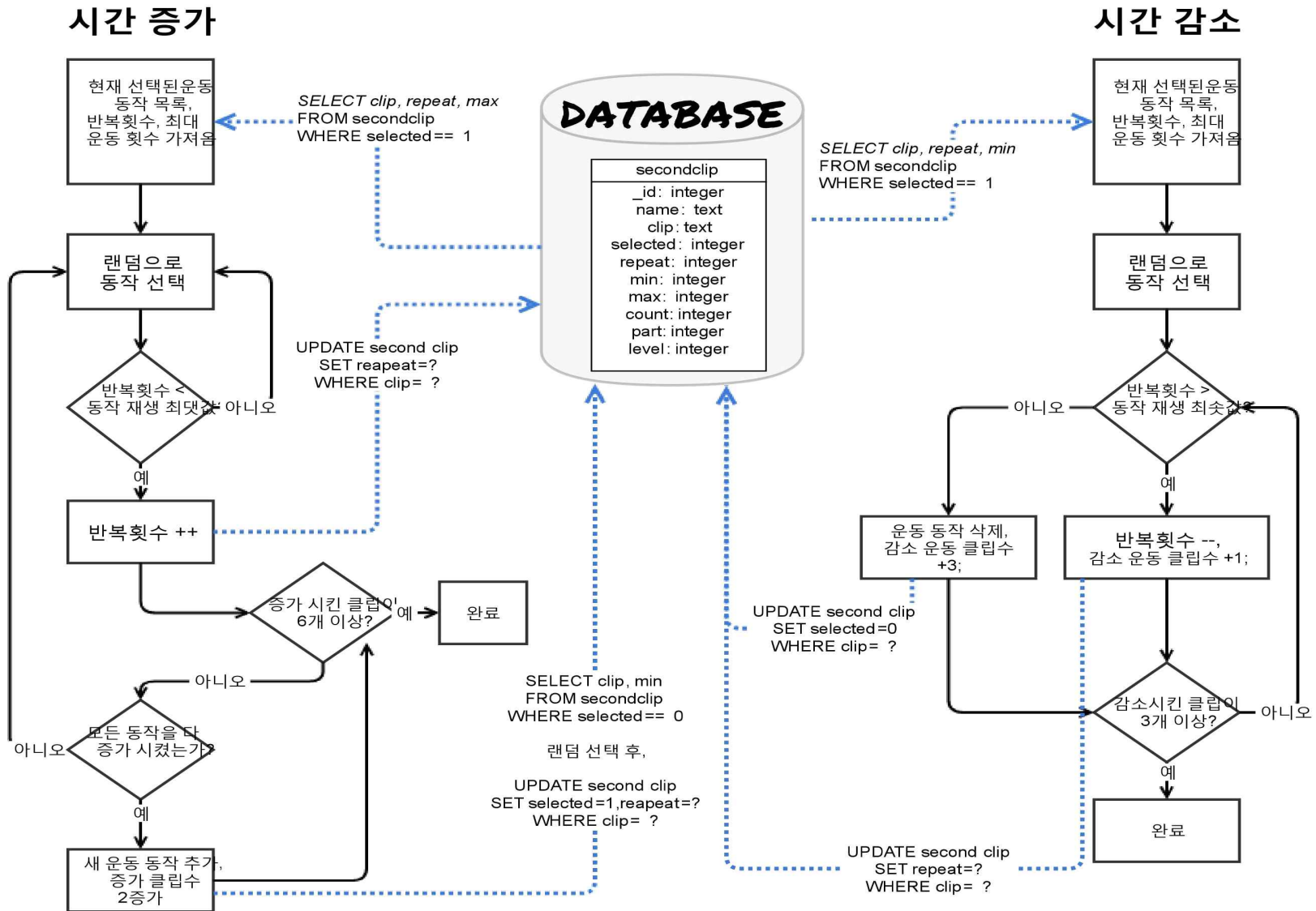


위 그림은 운동루틴 재생에 대한 플로우 차트로입니다. selected : integer 값은 현재 이 운동이 선택되었는가를 나타내고 있고, 선택되었다는 것은 운동 루틴에 현재 운동이 있다는 뜻입니다. 재생 전 clip 값을 이용해 해당되는 운동 동작 이미지 6개를 불러오고, 1, (2,3,4,5 의 반복), 6 순서로 화면을 갱신하여 애니메이션 형태로 보여줍니다. 반복의 순서는 repeat: integer 에 저장되어 있어 이 값으로 운동의 강도를 조정합니다.

④ 나이, 운동 부위에 따른 운동 동작 조정



⑤ 운동량 증가, 감소 부분 플로우 차트

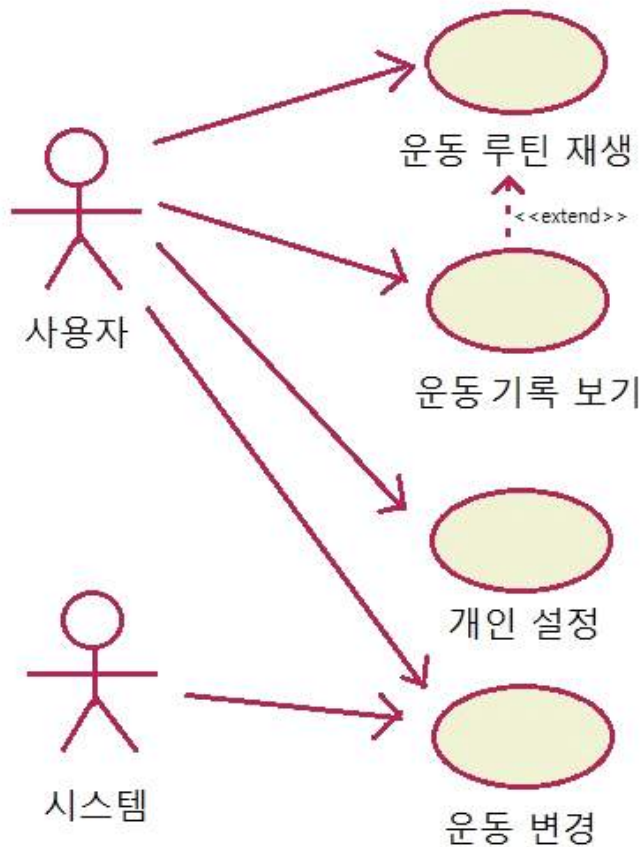


4. 제공되는 서비스의 내용과 서비스별 프로세스 다이어그램

① Requirement List

No.	Requirement	use case
1	사용자는 시작버튼을 눌러 오늘의 운동루틴을 재생할 수 있다.	운동루틴 재생
2	사용자는 주의해야할 신체부위(허리, 무릎, 목)를 선택할 수 있다.	개인 설정
3	사용자는 운동 주기(2일 ~ 14일)를 변경할 수 있다.	개인 설정
4	시스템은 운동주기에 맞춰 운동량을 증가시켜 준다.	운동 변경
5	기온, 습도에 따라 연령을 고려해 운동량이 변경된다.	운동 변경
6	사용자는 하루의 운동량을 선택에 따라 감축한다.	운동 변경
7	연령에 따라 알맞은 강도의 운동 동작을 선정한다.	개인 설정
8	사용자가 선택한 신체부위에 알맞은 운동 동작을 선정한다.	개인 설정
9	최근 1주일간의 운동 시간이 기록되고 보여진다.	기록 보기

② Usecase



③ Usecase Description (설계 변경 사항은 ⑦ 참조)

Use-case Description: 운동루틴 재생	
Actor action	System response
1. 재생 버튼을 누른다.	2. 운동 내용을 가져와서 예상 재생시간을 띄워준다.
3. 운동 시간을 감소 버튼 누른다.	4. 운동 반복횟수 감소 시켜 시간을 5분 단축시키도록 한다.
5. 확인을 누른다.	6. 재생, 재생이 끝나면 소요된 시간을 저장한다.
Alternative Courses <ul style="list-style-type: none"> - 단계 3에서 확인 버튼을 누를 경우 단계 5로 갈 수 있다. - 단계 5에서 다시 감소 버튼을 누를 경우 단계 4로 간다. - 단계 6이후 운동 기록 보기를 선택하면 운동기록을 보여준다. 	

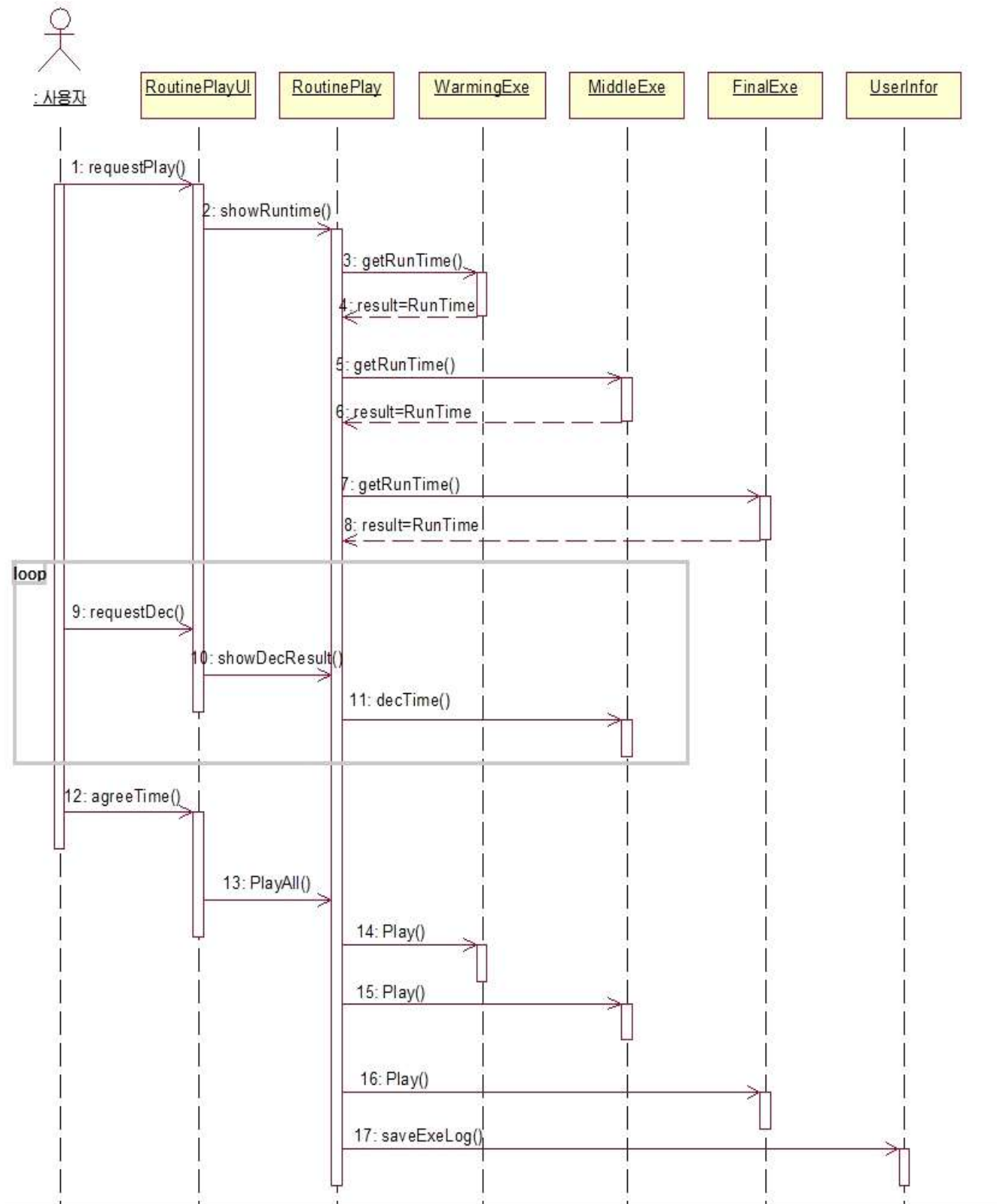
Use-case Description: 운동기록 보기	
Actor action	System response
1. 운동 기록 보기를 요청한다	2. 운동 기록을 가져와 보여준다

Use-case Description: 개인 설정	
Actor action	System response
1. 개인 설정을 요청한다	2. 개인 설정 사항들을 보여준다
3. 변경하고 싶은 개인 설정 항목을 선택하고 선택 값을 입력한다	4. 설정을 변경한다. 5. 설정에 알맞게 운동 루틴을 변경한다

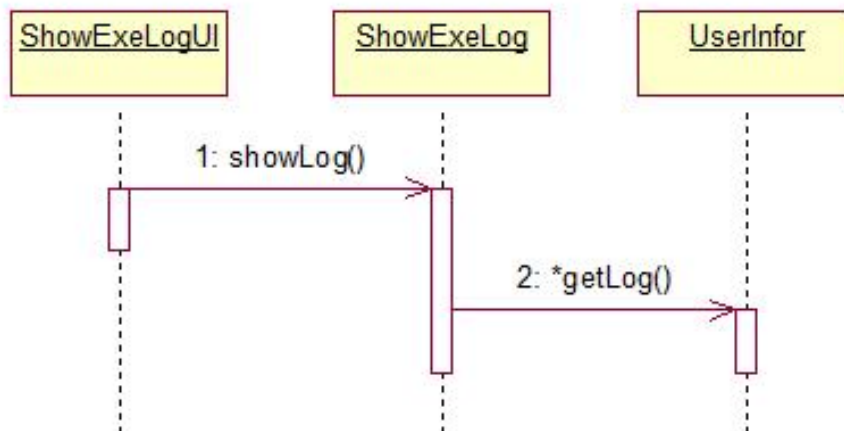
Use-case Description: 운동량 변경	
Actor action	System response
1. 운동량 변경을 요청한다	2. 주기를 확인하여 운동량을 증가시킨다 3. 날씨 데이터를 가져온다 4. 날씨에 따라 운동량을 변경한다

④ Sequence Diagram (설계 변경 사항은 ⑦ 참조)

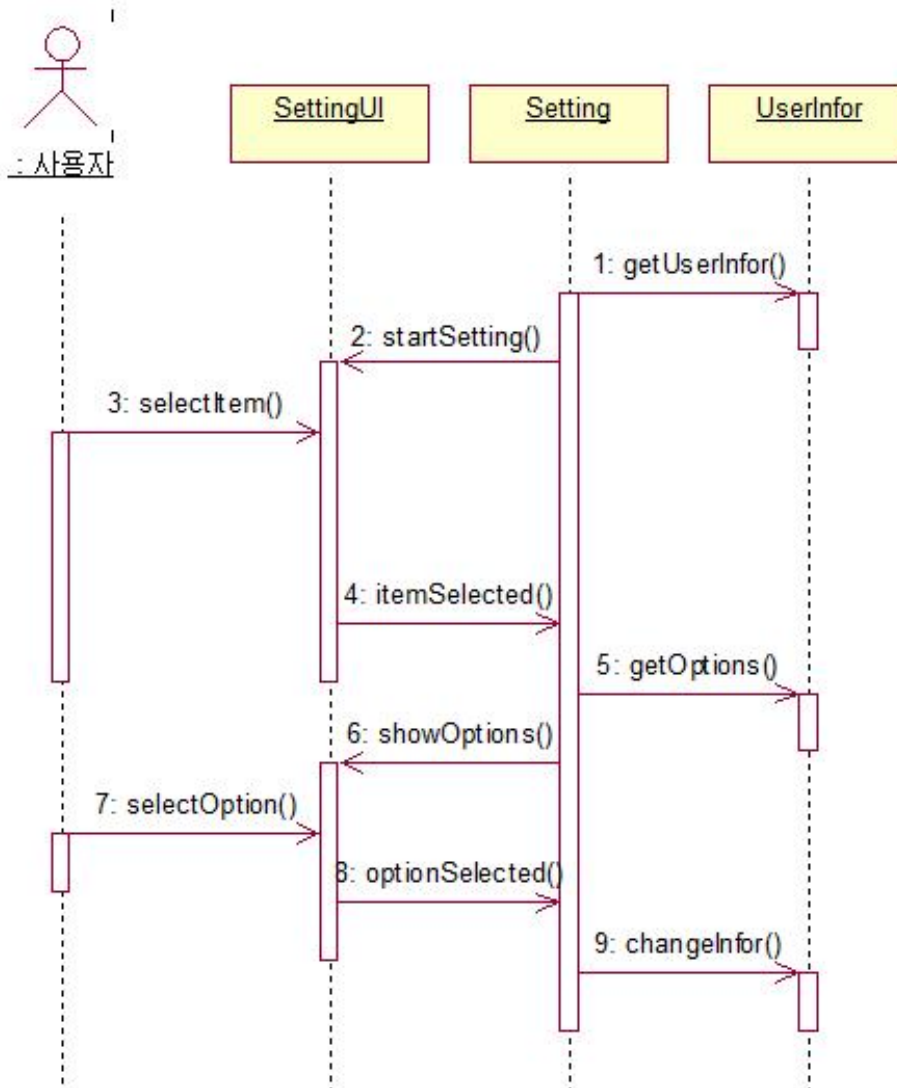
1. 운동루틴 재생



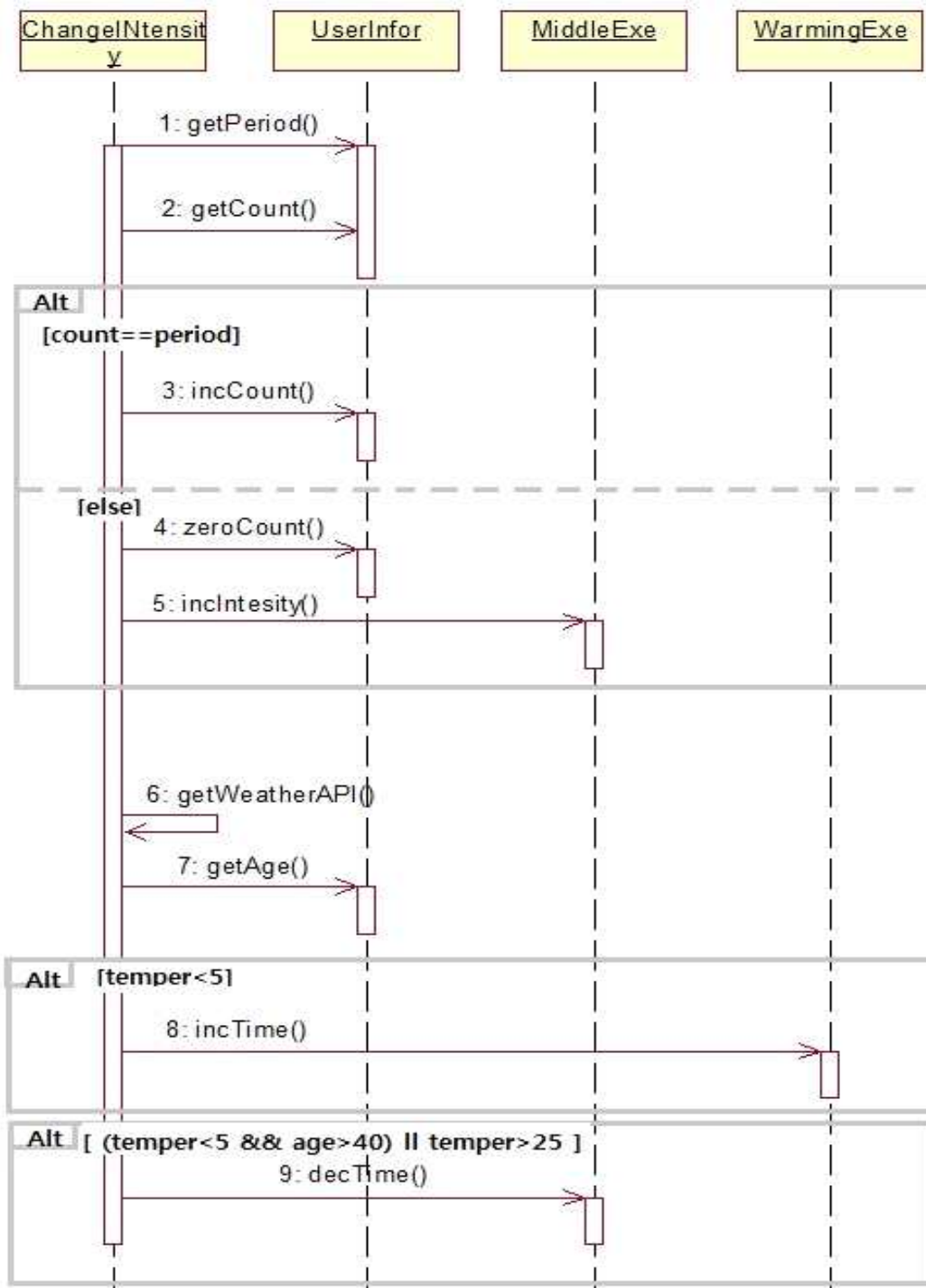
2. 운동기록 보기



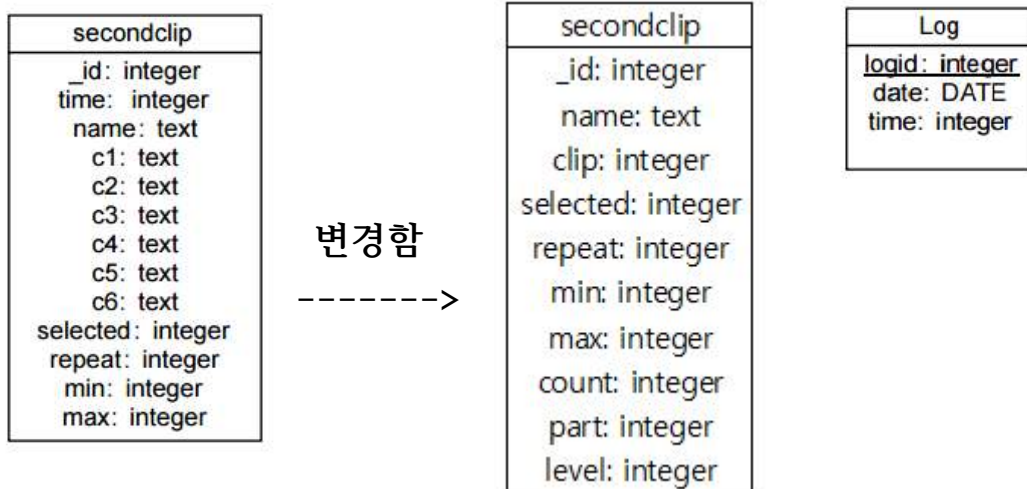
3. 개인 설정



4. 운동량 변경



⑤ ER 다이어그램

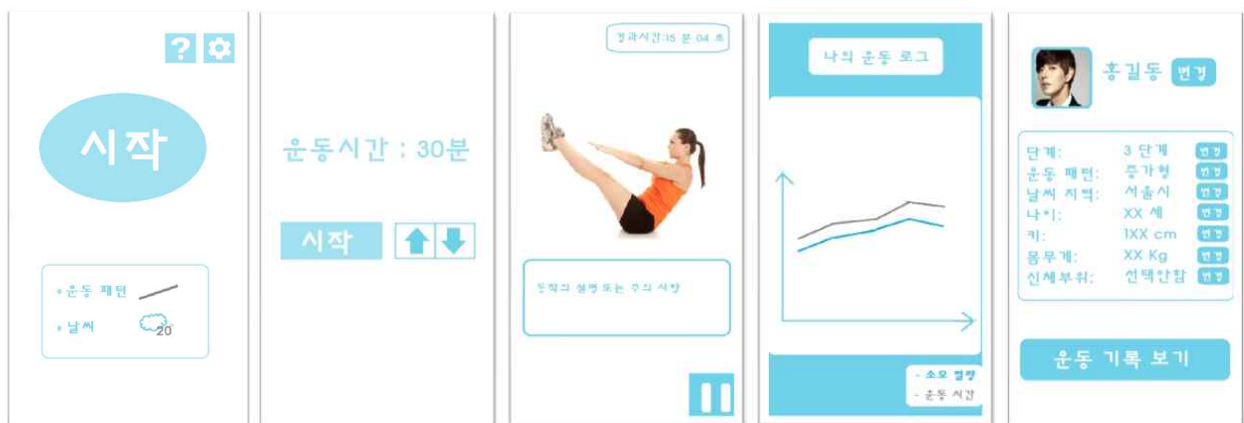


※ 사용자 정보 (이름, 나이, 운동 주기, 운동 부위) 는 shared preference를 사용

<Schema>

- secondclip (_id, name, clip, selected, repeat, min, max, count, part, level)
- log (logid, date, time, userlog)

⑥. UI 설계



⑦ 구현하면서 설계가 변경된 점

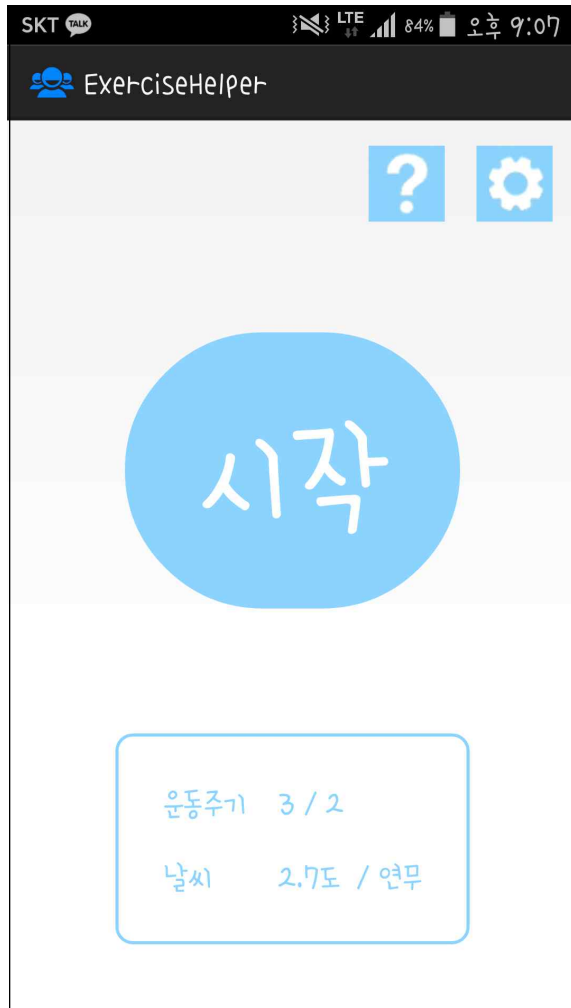
- mysql 대신 안드로이드 내부의 sqlite를 사용하였습니다.
- 데이터베이스의 설계를 매 질의마다 join연산이 일어나지 않도록 바꾸었습니다.
- 사용자의 설정 정보의 저장을 database가 아닌 shared preference를 사용하도록 변경하였습니다.
- 운동 데이터베이스의 삽입을 sqlite manager를 사용해 만든 sqlite 파일을 assert 파일에 저장해, SD카드로 복사하는 방법으로 변경하였습니다.
- 데이터 베이스 테이블의 스키마를 변경하였습니다. (count, part, level 행 추가)
- 알고리즘 수정 : 나이와 운동 시 주의해야 할 신체부위를 선택 하면 데이터 베이스에 따라 운동이 변경됩니다.
- 나이 선택 : 데이터 베이스에 운동 동작의 난이도 (level) 행에 따라 35세 이상은 난이도 4이상, 55세 이상을 난이도 3이상, 75세 이상은 난이도 2이상의 운동 동작을 삭제합니다.
- 신체 부위 : 선택 시 현재 선택되어 있는 운동들 중 주의해야 할 부위와 같은 운동(part 행)을 삭제합니다.
- 운동 동작의 재생을 서피스뷰를 사용한 이미지의 애니메이션으로 사용하였습니다.
- 운동 재생 전 재생 할 운동 동작의 이름과 운동 시 예상 소모 칼로리를 보여주었습니다.
- 경과시간을 보여주기 위한 스레드를 추가하였습니다.
- 날씨 api 파싱 방법을 xmlPullParser를 사용하였습니다.

5. 시스템 구현 환경

- 개발환경 : 안드로이드 4.2.2 (google API 17)
- 툴 : 이클립스

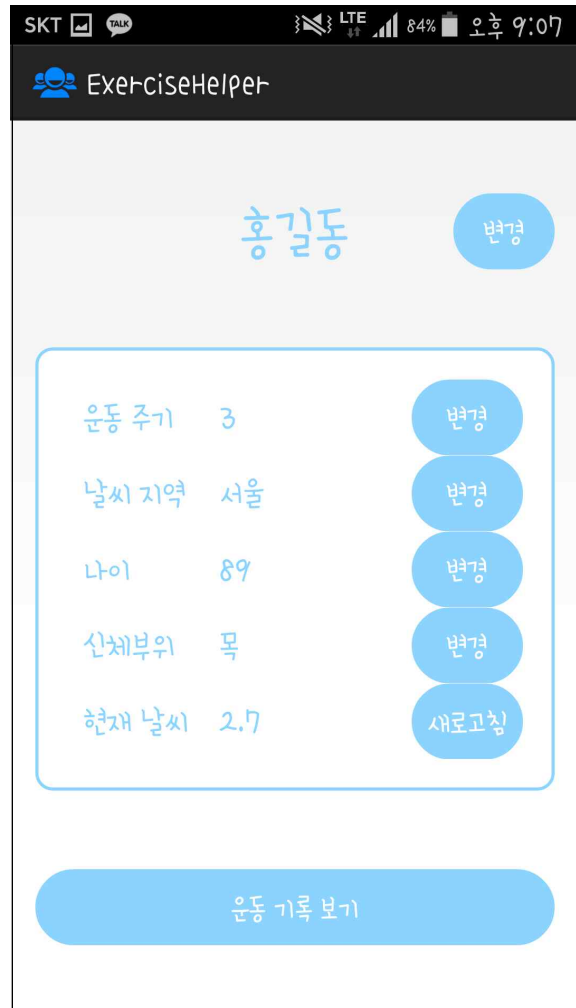
6. 구현결과

1) 어플리케이션 실행 화면



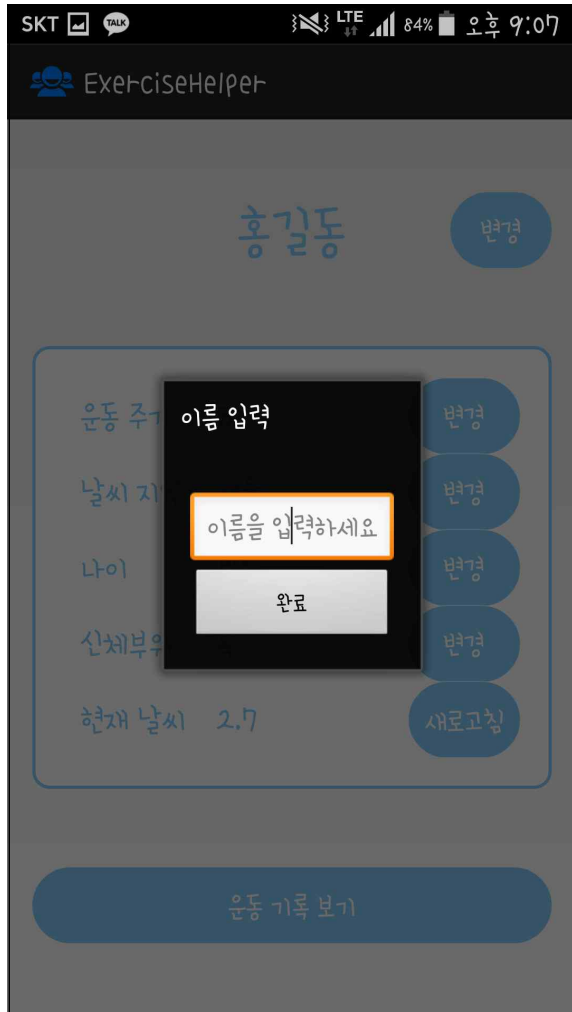
① 시작 화면

- 설정 버튼이 우측 상단에 있고, 클릭 할 경우 화면 ②로 넘어간다.
- 시작 버튼을 누를 경우 ⑤로 넘어간다.
- 현재 선택된 운동 주기와 운동 횟수, 날씨를 보여준다.



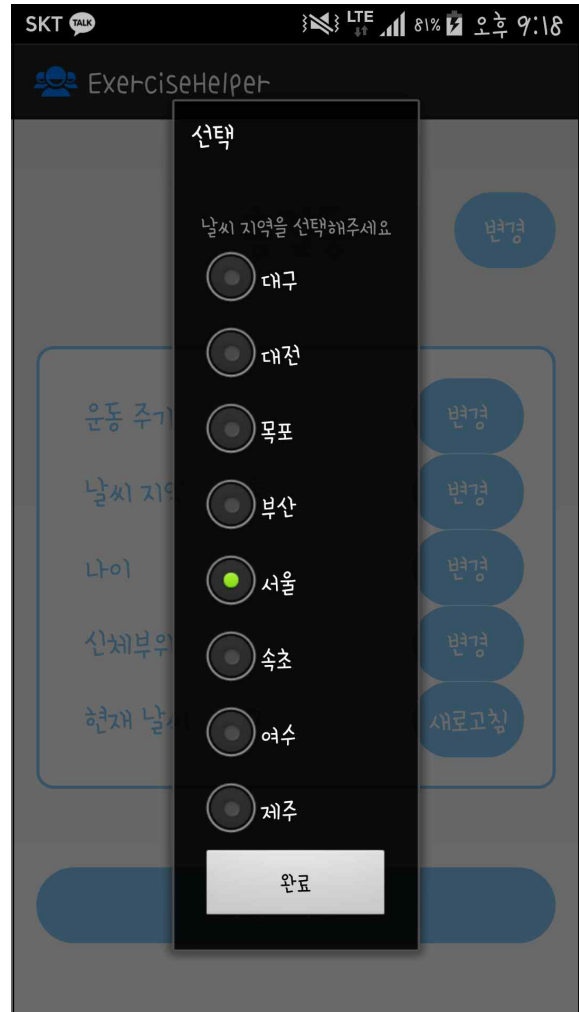
② 설정

- 현재 설정 된 내용들을 보여주는 동시에 변경 할 수 있게 해준다.
- 우측의 변경버튼을 누를 경우 ③ 또는 ④ 의 화면으로 넘어간다.
- 현재 날씨 새로 고침버튼을 누르면 현재 기온 이 업데이트 된다. (기상청 데이터 기준)
- 운동 기록 보기 버튼을 누르면 ⑥으로 넘어간다.



③ 설정 화면1

- 이름을 입력 받아 저장한다.
- 운동주기와 나이 모두 동일한 UI를 사용한다. (생략)
- 나이를 선택 할 경우 35세 이상은 운동 동작의 난이도 4이상, 55세 이상을 난이도 3이상, 75세 이상은 난이도 2이상의 운동 동작을 삭제합니다.
- 신체부위를 선택할 경우 현재 선택 되어 있는 운동들 중 선택한 운동 부위에 해로운 운동 동작들을 삭제합니다.



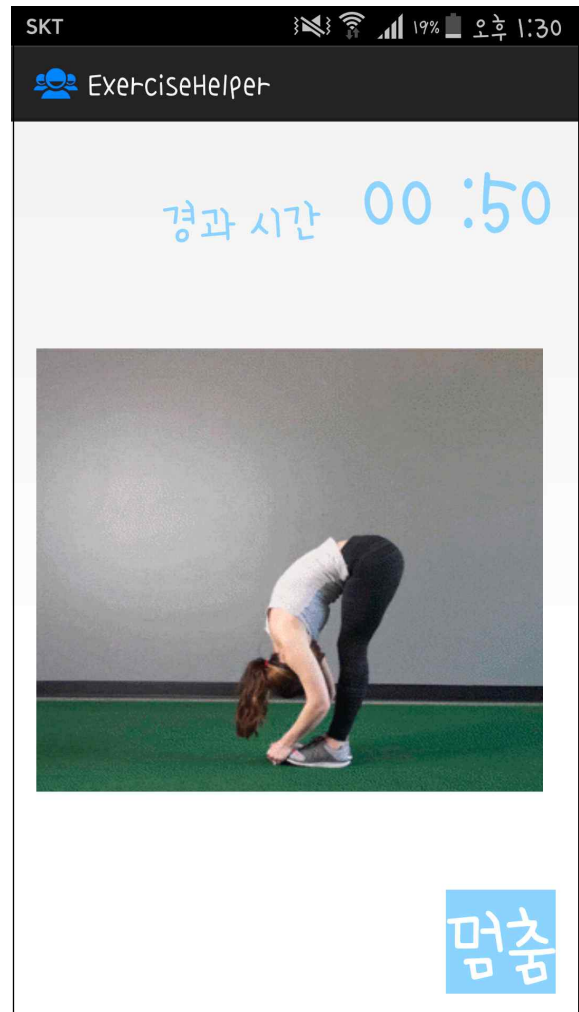
④ 설정 화면2

- 날씨를 불러올 지역을 선택 하도록 한다.
- 신체부위도 동일한 UI를 사용한다. (생략)



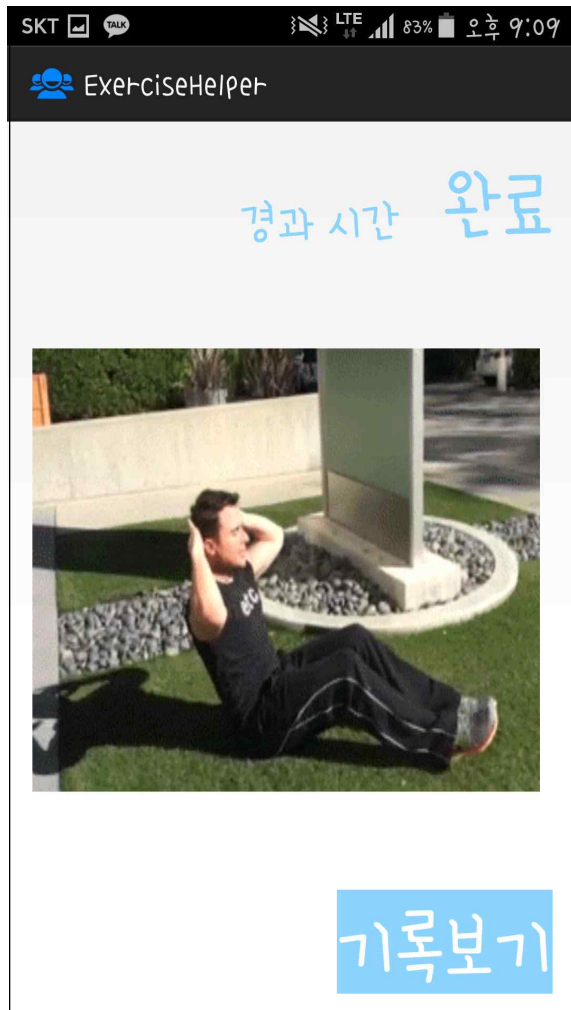
⑤ 준비 화면

- 저장된 사용자의 정보와 날씨에 의해 선정하고 이에 따라 재생 될 운동 시간과 예상 소모 칼로리를 보여준다.
- 시작을 누를 경우 ⑥ 으로 넘어가 재생을 시작한다.
- 증가/ 감소를 누를 경우 재생할 운동들의 강도, 개수를 수정하여 변경된 운동 시간과 예상 소모 칼로리를 보여준다.
- 재생될 운동들의 이름을 보여준다.
- 설정된 사용자의 정보를 보여준다.



⑥ 재생1

- 제일 상단에 현재까지 운동 시간을 보여준다.
- 운동 동작을 애니메이션 형태로 보여준다.
- 좌측 하단에 멈춤 버튼이 있다.



⑦ 재생2

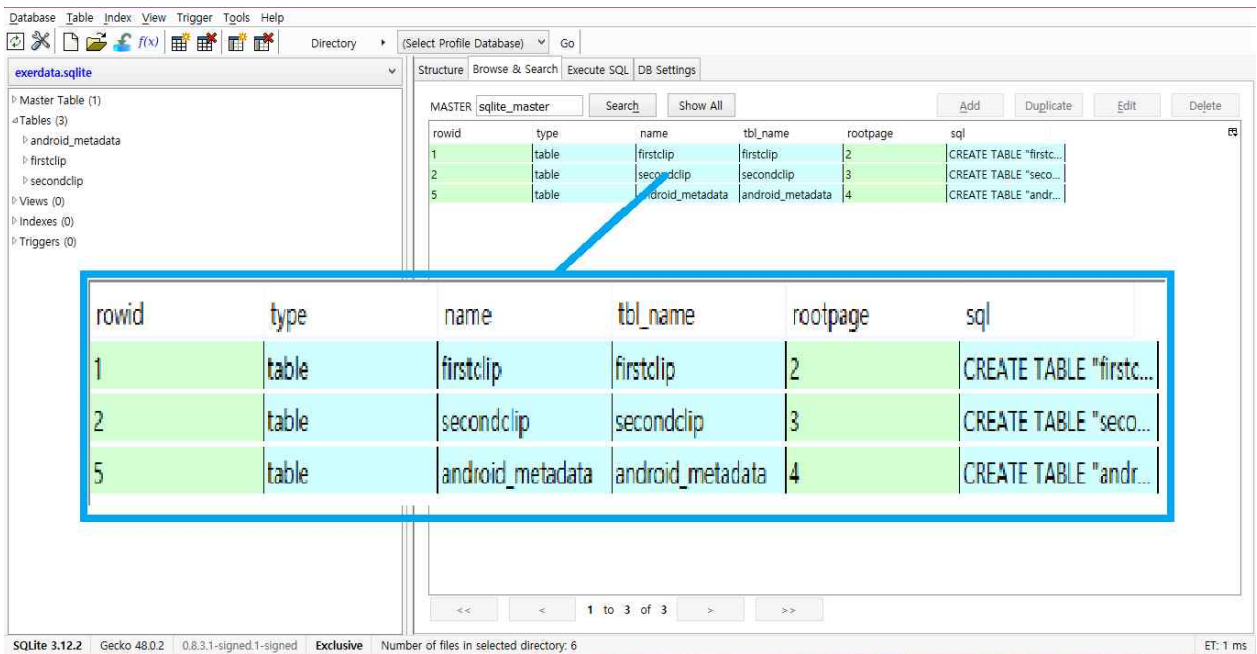
- 재생이 완료 (운동이 종료) 될 경우 경과시간 대신 '완료' 메시지를 띄워 주고, 재생 된 시간을 저장한다.
- '멈춤' 버튼이 '기록보기' 버튼으로 변경된다.
- 기록보기 버튼을 누를 경우 ⑧로 이동한다.



⑧ 기록 보기

- 최근 10회까지의 운동 기록을 그래프 형태로 보여준다.
- 재생을 끝까지 완료한 경우에만 기록이 저장되어 보인다.

2) 데이터 베이스



▲ 데이터 베이스 내의 테이블들 : firstclip(빈 테이블), secondclip, android_metadata 테이블이 존재

	B	C	D	E	F	G	H	I	J
1	name	clip	selected	repeat	min	max	count	part	level
2	cat-cow	c0000	0	4	2	15	0	1	1
3	standing forward bend	c0001	1	5	2	20	0	0	2
4	modified bicycle crunch	c0002	0	5	2	20	0	0	3
5	standing bicycle launches	c0003	1	5	2	20	0	0	4
6	tuck jump	c0004	1	4	2	15	0	3	1
7	mountain pose	c0005	1	4	2	25	0	0	2
8	downward-facing dog	c0006	0	5	2	25	0	0	3
9	reverse fly	c0007	0	4	2	15	0	0	4
10	sit-up	c0008	0	5	2	30	0	0	1
11	sit-up2	c0009	0	4	2	15	0	1	2
12	sit-up3	c0010	1	5	2	20	0	0	3
13	twisting	c0011	0	5	2	20	0	2	4
14	stretching arm	c0012	1	5	2	20	0	0	1
15	standing forward	c0013	1	4	2	15	0	0	2
16	arm forward	c0014	1	4	2	25	0	0	3
17	twisting legs	c0015	0	5	2	25	0	0	4
18	flogs	c0016	0	4	2	15	0	0	1
19	bend forward	c0017	0	5	2	30	0	0	2
20	situp straight	c0018	0	3	2	15	0	0	3
21	mountain pose2	c0019	0	3	2	25	0	0	4
22	twisting bend	c0020	0	4	2	15	0	1	1

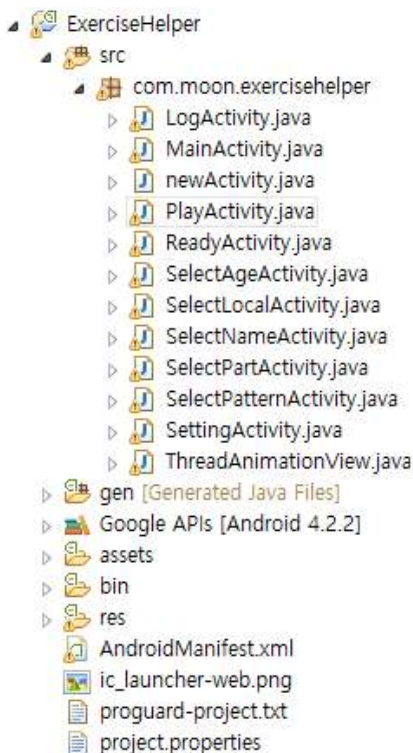
▲ 테이블 secondclip 내의 데이터

3) 구현하면서 설계가 변경된 점

- mysql 대신 안드로이드 내부의 sqlite를 사용하였습니다.
- 데이터베이스의 설계를 매 질의마다 join연산이 일어나지 않도록 바꾸었습니다.
- 사용자의 설정 정보의 저장을 database가 아닌 shared preference를 사용하도록 변경하였습니다.
- 설정 화면에서 운동 부위의 선택을 라디오 버튼으로 변경하였습니다.
- 운동 데이터베이스의 삽입을 sqlite manager를 사용해 만든 sqlite 파일을 넣는 방법으로 변경하였습니다.
- 데이터 베이스 테이블의 스키마를 변경하였습니다. (count, part, level 칼럼 추가)
- 알고리즘 수정 : 나이와 운동 시 주의해야 할 신체부위를 선택 하면 데이터 베이스에 따라 운동이 변경됩니다.
- 나이 선택 : 데이터 베이스에 운동 동작의 난이도 (level) 행에따라 35세 이상은 난이도 4이상, 55세 이상을 난이도 3이상, 75세 이상은 난이도 2이상의 운동 동작을 삭제합니다.
- 신체 부위 : 선택 시 현재 선택되어 있는 운동들 중 주의해야 할 부위와 같은 운동을 삭제합니다.

7. 핵심 모듈의 소스 프로그램

①전체 모듈



- 코드의 길이 상, 아래 코드는 PlayActivity.java, ReadyActivity.java, ThreadAnimationView.java 와 Setting.java 의 소스 파일 중 일부만 첨부하였습니다.

②ReadyActivity.java

```
/*
 * 클래스 : ReadyActivity
 * 운동 동작의 데이터 베이스 SD카드에 복사, 재생 할 운동 계산 및 보여줌, (본 문서 18page 의) 화면⑤ 띄워줌
 */
public class ReadyActivity extends Activity {

    private static String DB_PATH="/sdcard/";
    private static String DB_NAME="exerdata.sqlite";
    SQLiteDatabase db = null;

    SharedPreferences pref;
    SharedPreferences.Editor editor;
    int periods=0;
    int final_totaltime=0;

    /*
     * 함수 : onCreate()
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ready);

        String myPath = DB_PATH + DB_NAME;

        /*과거의 기록과 운동주기를 비교, 날씨데이터를 이용해서 시간조정 후 시간 띄워줌, 사용자 정보 띄워줌*/
        cmpHistory();
        adjustByWheather();
        setTime();
        setInformation();

        Button startBtn = (Button) findViewById(R.id.startBtn);
        Button upBtn = (Button) findViewById(R.id.upBtn);
        Button downBtn = (Button) findViewById(R.id.downBtn);
        startBtn.setEnabled(true);
    }
}
```

```

/* 버튼 클릭 리스너 (시작, 증가, 감소 버튼) */
startBtn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        if(final_totaltime < 0) {
            Toast.makeText(getApplicationContext(),"증가버튼을 눌러주세요", Toast.LENGTH_LONG).show();
        }
        else{
            Intent myIntent = new Intent(getApplicationContext(), PlayActivity.class);
            myIntent.putExtra("totaltimeget", final_totaltime);
            startActivity(myIntent);
            //재생 액티비티를 띄워주면서 전체 재생 시간을 넘겨줌
        }
    }
});

upBtn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        incIntensity();
        setTime();
        //운동 시간을 증가 시키고 화면 갱신
    }
});

downBtn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        decIntensity();
        setTime();
        //운동 시간을 감소 시키고 화면 갱신
    }
});

}

/*
 * 함수 : adjustByWheather()
 * 온도가 5도이하, 25도 이상일 경우 운동 시간 감소시킴
 */

public void adjustByWheather() {
    pref = getSharedPreferences("prefs", 0);

    String m_temperature;
    float m_temperature1=0;

```

```

int m_temperature2=0;

m_temperature = pref.getString("temp", "20");
//Toast.makeText(getApplicationContext(),m_temperature, Toast.LENGTH_LONG).show();
m_temperature1 = Float.parseFloat(m_temperature);
m_temperature2 = (int) m_temperature1;

if((m_temperature2 > 25) || (m_temperature2 < 5)) {
    decIntensity();
}

}

/*
 * 함수 : setTime()
 * 현재 데이터 베이스에서 선택된 운동, 반복 시간을 가져와서 시간을 계산후 텍스트뷰 설정해줌
 */
private void setTime () {

    TextView testship = (TextView) findViewById(R.id.testship);
    TextView calText = (TextView) findViewById(R.id.calText);
    testship.setText("운동 : \n");
    int first = 0;

    String myPath = DB_PATH + DB_NAME;
    db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READONLY | SQLiteDatabase.NO_LOCALIZED_COLLATORS);

    Cursor c1 = db.rawQuery("select name, repeat from secondclip where selected =1 ",null);

    int recordCount = c1.getCount();
    int totalTime=0;
    float totalCalory=0;

    for (int i=0; i<recordCount;i++){
        c1.moveToNext();
        String name = c1.getString(0);
        int repeating = c1.getInt(1);

        if(first == 0) {
            first = 1;
            testship.append(name);
        }
        else {

```

```

        testship.append(", " + name);
    }
    totalTime = totalTime+ 2 + 4*repeating;    //시간계산
    totalCalory = (float) ((0.097* (float)totalTime) + 0.194);

}
c1.close();

db.close();

TextView timeText = (TextView) findViewById(R.id.timeText);

totalTime = totalTime*12;
totalTime = totalTime/10;

int min = totalTime / 60;
int sec = totalTime % 60;
String totime = String.format("%02d 분    %02d 초", min, sec);

final_totaltime = totalTime;

timeText.setText(totime);
timeText.invalidate();
calText.setText("( " + totalCalory + " Kcal )");
calText.invalidate();

}

/*
 * 함수 : setInformation()
 * 현재 개인 설정 정보를 띄워준다.
 */
public void setInformation () {

    String tempo;
    TextView descriptionTxt = (TextView) findViewById(R.id.descriptionTxt);

    descriptionTxt.setText("*사용자 정보₩n");

    pref = getSharedPreferences("prefs", 0);

    tempo = pref.getString("age", "ND");
    if(tempo.equals("ND")) {}
    else {

```

```

        descriptionTxt.append("나이 : " + tempo + "\n");
    }
    tempo = pref.getString("temp", "ND");
    if(tempo.equals("ND")) {}
    else {
        descriptionTxt.append("기온 : " + tempo + "\n");
    }
    tempo = pref.getString("part_txt", "ND");
    if(tempo.equals("ND")) {}
    else {
        descriptionTxt.append("주의 해야할 부분 : " + tempo);
    }
    descriptionTxt.invalidate();

}

/*
 * 함수 : checkDatabase()
 * SD 카드에 운동동작 관련 DB파일이 존재 하는지를 확인한다.
 */
private boolean checkDatabase(){
    SQLiteDatabase checkDB = null;
    try {
        String myPath = DB_PATH + DB_NAME;
        checkDB = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READONLY);
    }catch(SQLiteException e){}

    if (checkDB != null) {checkDB.close();}
    return checkDB != null? true:false;
}

/*
 * 함수 : copyDatabase()
 * SD카드에 assets에 등록된 DB파일을 복사한다.
 */
private void copyDatabase() throws IOException{
    InputStream myInput = this.getAssets().open(DB_NAME);
    String outFileName = DB_PATH + DB_NAME;
    OutputStream myOutput = new FileOutputStream(outFileName); //이부분에서 에러!!!!-->고침 메니페스트 등록

    byte[] buffer = new byte[2048];

```

```

        int total_length = 0;
        int length;

        while ((length = myInput.read(buffer))>0){
            myOutput.write(buffer, 0, length);
            total_length += length;
        }

        total_length+=length;
        myOutput.flush();
        myOutput.close();
        myInput.close();
    }

    /*
    * 함수 : createDatabase()
    * SD카드에 DB파일이 있는지 확인하고 없을 경우 assets에 등록된 DB파일을 복사한다
    */
    public void createDatabase() throws IOException {
        boolean dbExist = checkDatabase();

        if (dbExist) {
            /*디비가 존재한다면 기존 데이터 베이스 삭제*/
            /*
            SQLiteDatabase checkDB = null;
            String myPath = DB_PATH + DB_NAME;
            this.deleteDatabase(myPath);
            Toast.makeText(getApplicationContext(), "DB 삭제", Toast.LENGTH_LONG).show();*/
        }
        else {
            /*데이터 베이스 생성*/
            try {
                copyDatabase();
            } catch (IOException e){
                throw new Error ("Error copying database");
            }
        }
    }
}

```

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}

```

```

/*
 * 함수 : cmpHistory
 * 주기와 운동 기록을 비교하여 운동량을 증가시킴
 */

```

```

public void cmpHistory () {

    pref = getSharedPreferences("prefs", 0);

    String m_period;
    String m_count;
    int m_period2;
    int m_count2;

    m_period = pref.getString("period", "5");
    m_count = pref.getString("count", "0");

    m_period2 = Integer.valueOf(m_period);
    m_count2 = Integer.valueOf(m_count);

    if(m_period2 <= m_count2) {
        incIntensity();
        //이제 count=0으로..
        editor = pref.edit();
        editor.putString("count", "0");
        editor.commit();
    }
}

```

```

    }

}

/*
 * 함수 : incIntensity()
 * 재생시간 (운동의 강도)를 증가시킴 : 현재 선택 되어있는 임의의 운동을 선택해서 최대 운동 횟수와 비교, 더 작을 경우 증가시킴(이를
반복함), 반복하여 6개 이상의 운동 동작을 증가 시키지 못하였을 경우 새로운 운동 동작을 추가한다.
 */
public void incIntensity() {

    Random random = new Random();
    int rand=0;
    int incNumber=0;
    int tooSmall=0;

    String myPath = DB_PATH + DB_NAME;
    db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE | SQLiteDatabase.NO_LOCALIZED_COLLATORS);

    Cursor c1 = db.rawQuery("select repeat, max, clip from secondclip where selected =1 ",null);

    int recordCount = c1.getCount();

    if(recordCount < 6) {
        tooSmall=1;
    }

    rand = random.nextInt(recordCount);

    c1.moveToFirst();

    for (int i=0; i<rand; i++){
        c1.moveToNext();
    }
    //시작점

    int repeating;
    int maxtime;
    int position;
    String clip1;

    for(int j=0; (j<recordCount) && (incNumber < 3); j++) {

```



```

        if(tooSmall==1) {
            j = j+3;
        }

        repeating = c1.getInt(0);
        maxtime = c1.getInt(1);
        clip1 = c1.getString(2);
        if(repeating < maxtime) { //증가

            int repeatToUpdate = repeating +1;

            //UPDATE [테이블] SET [열]= '변경할값' WHERE [열] is null

            String SQL = "UPDATE secondclip SET repeat=? WHERE clip=?";
            String args[] = {"", ""};

            args[0] = String.valueOf(repeatToUpdate);
            args[1] = clip1;

            db.execSQL(SQL, args);

            incNumber++;
        }
        else { //증가불가
        }

        position = c1.getPosition();
        if ((position+1) == recordCount) { //마지막거라면 맨 처음으로
            c1.moveToFirst();
        }
        else {
            c1.moveToNext();
        }
    }

    c1.close();
    db.close();

    if(incNumber < 3) {

```

```

        //새로운 운동 추가가 필요하다면
        db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE
SQLiteDatabase.NO_LOCALIZED_COLLATORS);

        c1 = db.rawQuery("select repeat, min, clip from secondclip where selected =0 ",null);
        recordCount = c1.getCount();
        if(recordCount < 1) {
            //증가불가
            Toast.makeText(getApplicationContext(), "더이상 증가가 불가합니다", Toast.LENGTH_LONG).show();
        }
        else {
            rand = random.nextInt(recordCount);

            c1.moveToFirst();
            for (int i=0; i<rand; i++){
                c1.moveToNext();
            }

            repeating = c1.getInt(0);
            int mintime = c1.getInt(1);
            clip1 = c1.getString(2);

            String SQL = "UPDATE secondclip SET selected=1, repeat=? WHERE clip=?;";
            String args[] = {"", ""};

            args[0] = String.valueOf(mintime);
            args[1] = clip1;

            db.execSQL(SQL, args);

            c1.close();
        }
        db.close();
    }
}

```

```
}
```

```
/*  
 * 함수 : incIntensity()  
 * 재생시간 (운동의 강도)를 감소시킴 : 운동 강도 증가의 방법과 동일  
 */
```

```
public void decIntensity() {
```

```
    Random random = new Random();  
    int rand=0;  
    int decNumber=0;
```

```
    String myPath = DB_PATH + DB_NAME;  
    SQLiteDatabase db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE |  
    SQLiteDatabase.NO_LOCALIZED_COLLATORS);
```

```
    Cursor c1 = db.rawQuery("select repeat, min, clip from secondclip where selected =1 ",null);
```

```
    int recordCount = c1.getCount();
```

```
    if(recordCount == 0) {  
        //0개라면 감소 불가  
        Toast.makeText(getApplicationContext(), "더이상 감소가 불가능합니다", Toast.LENGTH_LONG).show();  
    }
```

```
    else { //1개이상 클립이 남아있다면 감소가 가능하다  
        rand = random.nextInt(recordCount);
```

```
        c1.moveToFirst();  
        for (int i=0; i<rand; i++){  
            c1.moveToNext();  
        } //랜덤수 만큼 점프
```

```
        int repeating;  
        int mintime;  
        int position;  
        String clip1;
```

```

        Toast.LENGTH_LONG).show();

        for(int j=0; (j<recordCount) && (decNumber < 3); j++) {    //slected가 1인거 돌면서 3개를 감소시킴

            repeating = c1.getInt(0);
            mintime = c1.getInt(1);
            clip1 = c1.getString(2);

            //Toast.makeText(getApplicationContext(), clip1 + "을 변경하려 합니다. 변경전" + repeating,

            if(repeating > mintime) { //repeat를 감소

                int repeatToUpdate = repeating -1;

                //UPDATE [테이블] SET [열]= '변경할값' WHERE [열] is null

                String SQL = "UPDATE secondclip SET repeat=? WHERE clip=?;";
                String args[] = {"", ""};

                args[0] = String.valueOf(repeatToUpdate);
                args[1] = clip1;

                db.execSQL(SQL, args);

                decNumber++;
            }
            else if (repeating == mintime){    //최소운동중 삭제!!

                if(recordCount==1) {
                    Toast.makeText(getApplicationContext(), "더이상 감소가 불가능합니다",
                    Toast.LENGTH_LONG).show();
                }
                else {
                    String SQL = "UPDATE secondclip SET selected =0 WHERE clip=?;";
                    String args[] = {""};
                    args[0] = clip1;
                    db.execSQL(SQL, args);
                    decNumber = decNumber+3;
                }
            }
        }
    }

```

```

        else {
        }

        position = c1.getPosition();
        if ((position+1) == recordCount) { //마지막거라면 맨 처음으로
            c1.moveToFirst();
        }
        else {
            c1.moveToNext();
        }
    }
}

c1.close();
db.close();
}
}

```

```

public class PlayActivity extends Activity {

    ThreadAnimationView mySurfaceView;
    int totalTime=0;
    Button gotoLogs;
    Button stopButton;
    boolean backClicked = false;

    private static final int SEND_THREAD_INFORMAION =0;
    private static final int SEND_THREAD_STOP_MESSAGE =1;
    private static final int SEND_THREAD_FINAL =2;

    private SendMessageHandler mMainHandler = null;
    private CountThread mCountThread = null;

    int playtime = 10000;

```

③ PlayActivity.java

```
/*
 * 클래스 : PlayActivity
 * 재생화면 (화면 ⑥)을 띄워주고, 리스너 동작
 */
public class PlayActivity extends Activity {

    ThreadAnimationView mySurfaceView;
    int totalTime=0;
    Button gotoLogs;
    Button stopButton;
    boolean backClicked = false;

    private static final int SEND_THREAD_INFORMAION =0;
    private static final int SEND_THREAD_STOP_MESSAGE =1;
    private static final int SEND_THREAD_FINAL =2;

    private SendMessageHandler mMainHandler = null;
    private CountThread mCountThread = null;

    int playtime = 10000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_play);
        mySurfaceView = new ThreadAnimationView(this);
        //여기서애러 mySurfaceView.thread.setImage("c00000.png", "c00001.png", "c00002.png", "c00003.png", "c00004.png", "c00005.png");
        //mySurfaceView.startThread(this, 3);
        stopButton = (Button) findViewById(R.id.stopButton);
        gotoLogs = (Button) findViewById(R.id.gotoLogs);
        //TextView runtimeing = (TextView) findViewById(R.id.RunTime);

        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON); //화면꺼짐 방지

        setContentView();

        //메인 핸들러 생성
        mMainHandler = new SendMessageHandler();
        mCountThread = new CountThread();
        mCountThread.start();

        Intent myintent = getIntent();
```

```

playtime = myintent.getIntExtra("totaltimeget", 10000);

stopButton.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        //정지 버튼이 눌림

        mySurfaceView.thread.interrupt();
        Toast.makeText(getApplicationContext(),"버튼이 눌렸어요~", Toast.LENGTH_LONG).show();
    }
});

gotoLogs.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        //로그로 가기

        mySurfaceView.thread.interrupt(); //인터럽트 신호보내 스레드 종료
        CountThread.interrupted();
        Intent myIntent = new Intent(getApplicationContext(), LogActivity.class);
        startActivity(myIntent);

        finish(); //액티비티 종료
    }
});
}

```

```

/*
 * 클래스 : SendMessageHandler
 * 1초가 지났다는 메시지를 받아 처리하는 메시지 핸들러
 * 메시지를 받으면 경과시간을 갱신해준다, 완료 메시지를 받으면 완료 메시지를 띄워준다
 */
class SendMessageHandler extends Handler {

```

```

    @Override
    public void handleMessage (Message msg) {
        super.handleMessage(msg);

        int times;

```

```

switch (msg.what) {
case SEND_THREAD_INFORMAION:
    //times = msg.arg1;
    totalTime = msg.arg1;
    int min = totalTime / 60;
    int sec = totalTime % 60;
    String strTime = String.format("%02d :%02d", min, sec);

    runningTime.setText(strTime);
    break;

case SEND_THREAD_STOP_MESSAGE:
    break;

case SEND_THREAD_FINAL:
    if (backClicked == false) {
        runningTime.setText("완료");
        incCounter();
        makeLog(playtime);
        gotoLogs.setVisibility(View.VISIBLE);
        stopButton.setVisibility(View.INVISIBLE);
    }
    break;

default:
    break;
}

};

/*
 * 클래스 : CountThread
 * 1초에 한번씩 메시지를 보내는 스레드
 * 시간이 완료되었을 경우에 완료 메세지를 보낸다
 */
class CountThread extends Thread implements Runnable {

    private int running = 1;
    public CountThread() {

    }

    @Override
    public void run() {

```



```

        super.run();

        int i=-2;
        int j=0;
        while((j<playtime+2) && (!Thread.currentThread().isInterrupted())) {
            try {
                i++;
                Message msg = mMainHandler.obtainMessage();
                msg.what = SEND_THREAD_INFORMAION;
                msg.arg1 = i;

                mMainHandler.sendMessage(msg);

                //try {Thread.sleep(1000);}
                //catch (InterruptedException e) {e.printStackTrace();}
                Thread.sleep(1000);

                j++;
            } catch (InterruptedException e) {
            }
            finally {
                if(j == playtime+2) {
                    Message msg = mMainHandler.obtainMessage();
                    msg.what = SEND_THREAD_FINAL;
                    mMainHandler.sendMessage(msg);
                }
            }
        }
    }

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long

```

```

        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    /*
     * 함수 : onKeyDown
     * 뒤로 가기 버튼이 눌렸을 경우에 스레드를 종료해 주고 액티비티 끝냄
     */
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        // TODO Auto-generated method stub

        if(keyCode == KeyEvent.KEYCODE_BACK) { //이 키가 뒤로가기 버튼이라면

            mySurfaceView.thread.interrupt(); //인터럽트 신호보내 스레드 종료
            mCountThread.interrupt();
            backClicked = true;
            Toast.makeText(getApplicationContext(),"총   운동시간:   " + totalTime + "초   :플레이를   종료합니다",
Toast.LENGTH_LONG).show();

            finish(); //액티비티 종료
        }
        return false;
    }

    private TextView runningTime = null;

    public void setLayout() {
        runningTime = (TextView) findViewById(R.id.RunTime);
    }

    /*
     * 함수 : incCounter
     * 운동 횟수 값인 counter를 1 증가시킨다
     */
    public void incCounter () {
        SharedPreferences pref;
        SharedPreferences.Editor editor;

```

```

    pref = getSharedPreferences("prefs", 0);
    editor = pref.edit();

    String pre_count = pref.getString("count", "0");
    int m_count = Integer.valueOf(pre_count);
    m_count++;
    pre_count = String.valueOf(m_count);

    editor.putString("count", pre_count);
    editor.commit();
}

/*
 * 함수 : makeLog
 * 운동 시간을 데이터 베이스에 저장한다
 */
public void makeLog (int m_amount) {

    String DB_PATH2="/sdcard/";
    String DB_NAME2="exerdata.sqlite";

    SQLiteDatabase db = null;

    String myPath = DB_PATH2 + DB_NAME2;
    db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE | SQLiteDatabase.NO_LOCALIZED_COLLATORS);

    String sql = "CREATE TABLE IF NOT EXISTS "+ "log" + " ("
        + "_id integer PRIMARY KEY autoincrement, "
        + "date text NOT NULL, "
        + "time text NOT NULL, "
        + "amount integer);";
    db.execSQL(sql); //테이블 생성

    Calendar cal = Calendar.getInstance();
    String dateToString, timeToString;

    dateToString = String.format("%04d-%02d-%02d", cal.get(Calendar.YEAR), cal.get(Calendar.MONTH)+1,
cal.get(Calendar.DAY_OF_MONTH));
    timeToString = String.format("%02d:%02d:%02d", cal.get(Calendar.HOUR_OF_DAY), cal.get(Calendar.MINUTE),
cal.get(Calendar.SECOND));

    sql = "INSERT INTO log VALUES (null, ?, ?, ?);";
    String args[] = {"", "", ""};

```

```
args[0] = dateToString;  
args[1] = timeToString;  
args[2] = String.valueOf(m_amount);
```

```
db.execSQL(sql, args);
```

```
//Toast.makeText(getApplicationContext(),"insert 실행 (넣은 데이터): " + args[0] + args[1] + args[2] , Toast.LENGTH_LONG).show();
```

```
db.close();
```

```
}
```

```
}
```

④ ThreadAnimationView.java

```
/*
 * 클래스 : ThreadAnimationView
 * 영상 재생 스레드, 데이터 베이스에서 클립 아이디를 가져와서 해당 이미지를 로컬에 저장, 재생
 */

public class ThreadAnimationView extends SurfaceView implements Callback {

    ImageThread thread;
    Context con;

    /*생성자*/
    public ThreadAnimationView(Context context, AttributeSet attrs) {
        super(context, attrs);

        init(context);
        con=context;
    }

    public ThreadAnimationView(Context context) {
        super(context);
        init(context);
        con=context;
    }

    private void init (Context context) {
        SurfaceHolder holder = getHolder();
        holder.addCallback(this);

        /*디비에서 레코드의 수를 가져와서 스레드의 생성자에 넘겨준다*/
        SQLiteDatabase db = null;

        String myPath = "/sdcard/" + "exerdata.sqlite";
        db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READONLY | SQLiteDatabase.NO_LOCALIZED_COLLATORS);
        Cursor cu = db.rawQuery("select _id from secondclip where selected =1",null);
        int recordCount = cu.getCount();
        cu.close();
        db.close();

        //스레드 생성 (레코드의 수를 넘겨줌)
        thread = new ImageThread(context, holder, 3, recordCount);
    }
}
```

```

@Override
public void surfaceCreated(SurfaceHolder holder) {
    thread.setRunning(true);
    thread.start();
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    // TODO Auto-generated method stub
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    thread.setRunning(false);
}

/*
 * 클래스 : ImageThread
 * 스레드 클래스 - 이미지를 가져와 저장, 재생
 */
class ImageThread extends Thread {

    /*멤버 변수*/
    private SurfaceHolder mHolder;
    private int mImageWidth;
    private int mImageHeight;
    private int mCount = 0;
    private boolean myThreadrun = false;
    private int repeating;
    private int frame;

    private static final String DB_PATH="/sdcard/";
    private static final String DB_NAME="exerdata.sqlite";
    SQLiteDatabase db = null;
    private Drawable mFrontImage[];
    private int repeat[];

    /*생성자*/
    public ImageThread (Context context, SurfaceHolder surfaceHolder, int times, int frames) {
        mHolder = surfaceHolder;
        Resources res = context.getResources();
        repeating = times;
    }
}

```

```

        frame=frames;

        mFrontImage = new Drawable[frames*6]; //이미지 만큼 할당
        repeat = new int[frames];

        String myPath = DB_PATH + DB_NAME;
        SQLiteDatabase db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READONLY |
        SQLiteDatabase.NO_LOCALIZED_COLLATORS);

        Cursor cu = db.rawQuery("select clip, repeat from secondclip where selected = 1",null);

        int recordCount = cu.getCount();
        System.out.println("레코드카운트 출력!!!!!!!!!!" + recordCount);

        for (int i=0; i<recordCount;i++){
            cu.moveToNext();
            String clip1 = cu.getString(0);
            System.out.println("클립1!!!!!!!!!!" + clip1);

            this.setImage(clip1, i);

            repeat[i] = cu.getInt(1); //repeat 저장

        }
        cu.close();

        db.close();

        //

        mImageWidth = mFrontImage[1].getIntrinsicWidth();
        mImageHeight = mFrontImage[1].getIntrinsicHeight();

    }

    /*
     * 함수 : setImage
     * 이미지를 저장
     * 입력 : 이미지 아이디, 저장할 인덱스
     */
    public void setImage (String cname, int index){

```

```

        int n = index*6;
        AssetManager assetMgr = getContext().getAssets();
        InputStream inputStream = null;

        try {
            inputStream = assetMgr.open("drawables/" + cname + "0.png" );
            mFrontImage[n+0] = Drawable.createFromStream(inputStream, null);

            inputStream = assetMgr.open("drawables/" + cname + "1.png" );
            mFrontImage[n+1] = Drawable.createFromStream(inputStream, null);

            inputStream = assetMgr.open("drawables/" + cname + "2.png" );
            mFrontImage[n+2] = Drawable.createFromStream(inputStream, null);

            inputStream = assetMgr.open("drawables/" + cname + "3.png" );
            mFrontImage[n+3] = Drawable.createFromStream(inputStream, null);

            inputStream = assetMgr.open("drawables/" + cname + "4.png" );
            mFrontImage[n+4] = Drawable.createFromStream(inputStream, null);

            inputStream = assetMgr.open("drawables/" + cname + "5.png" );
            mFrontImage[n+5] = Drawable.createFromStream(inputStream, null);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void setRunning(boolean b) {
        myThreadrun = b;
    }

    /*
    * 함수 : run
    * 화면을 갱신하면서 영상재생
    */
    public void run() {
        Canvas c = null;
        int i=0;
        int j=0;

```


반복한다.

```
int tmp=0;
int tmp2=0;

int duration = this.repeating*6*this.frame;

/*인터럽트가 들어오기 전까지 실행*/
while(!Thread.currentThread().isInterrupted()) && ( i < this.frame ) { //칼럼의 로우만큼, 즉 운동의 수만큼

    for (j=0; j< (2+(this.repeat[i]*4)) ; j++){ //반복횟수 만큼 반복한다

        try {

            c = mHolder.lockCanvas(null);

            synchronized (mHolder) {
                doDraw(c, i, tmp);
                mCount++;
                if((j==0) && (i!=0)){ //첫번째 프레임 이라면
                    //sleep(5000);
                }
                sleep(1200);
            }

            tmp ++;
            //0 1234 1234 1234 5 로 반복함
            if (tmp == 5) {
                tmp2 ++;

                if (tmp2 == repeat[i]) {
                }
                else {
                    tmp = 1;
                }
            }

        } catch (InterruptedException ex) {
            Log.e("ThreadAnimationView", "Exception in thread",ex);
        } finally {
            if (c != null) {
                mHolder.unlockCanvasAndPost(c);
            }
        }
    }
}
```

```

        i++; //i번반복
        tmp=0;
        tmp2=0;
    }
    System.out.println("Thread is dead!!");
}

private void doDraw (Canvas canvas, int index, int count) {
    if (canvas != null){
        int ind = index*6;

        //갤럭시 s4
        mFrontImage[count+ind].setBounds(0, 0, 980, 850);
        mFrontImage[count+ind].draw(canvas);

        //갤럭시 s3
        mFrontImage[count+ind].setBounds(0, 0, 640, 560);
        mFrontImage[count+ind].draw(canvas);
    }
}
}

```

⑤ConnectThread (SettingActivity.java)

```
/*
 * 클래스 : ConnectThread
 * http://www.kma.go.kr (기상청) 으로부터 날씨 데이터를 가져와 파싱 (XmlPullParser 사용)
 */
class ConnectThread extends Thread {
    String urlStr;

    public ConnectThread(String inStr) {
        urlStr = inStr;
    }

    public ConnectThread() {
        urlStr = "http://www.kma.go.kr/XML/weather/sfc_web_map.xml";
    }

    public void run() {
        int temp=0;

        try {

            URL urlForHttp = new URL(urlStr);
            XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
            XmlPullParser parser = factory.newPullParser();
            parser.setInput(urlForHttp.openStream(), "utf-8");

            boolean bSet=false;
            int eventType = parser.getEventType();

            while(eventType != XmlPullParser.END_DOCUMENT) {
                switch(eventType) {
                    case XmlPullParser.START_DOCUMENT:
                        break;
                    case XmlPullParser.END_DOCUMENT:
                        break;
                    case XmlPullParser.START_TAG:
                        String tag = parser.getName();
                        if (tag.equals("local")) {
```

```

weatherData weather = new weatherData();
weather.ItemContents = "";
weather.Id = parser.getAttributeValue(null, "stn_id");

String state = parser.getAttributeValue(null, "desc");
weather.description = state;
String temperaturess = "섭씨";
weather.temperature = parser.getAttributeValue(null, "ta");
temperaturess += weather.temperature;
temperaturess += "도";
weather.ItemContents = weather.Id + "," + state + "," + temperaturess +
"";

dataList.add(weather);
bSet = true;
}
break;
case XmlPullParser.END_TAG:
break;
case XmlPullParser.TEXT:
if(bSet) {

bSet=false;

}
break;
}
eventType = parser.next();
temp++;
}

s_handler.post(new Runnable() {

    TextView tempText = (TextView) findViewById(R.id.tempText);
    SharedPreferences pref;
    SharedPreferences.Editor editor;
    String c_local;
    String c_temp;
    String c_desc;

    public void run() {
        ProgressDialog mProgressDialog;
        mProgressDialog = ProgressDialog.show (SettingActivity.this,"", "잠시만
기다려 주세요.",true, true);

```

```

int len = dataList.size();
for(int i = 0; i<len ; i++){

    pref = getSharedPreferences("prefs", 0);

    c_local = pref.getString("local", "108");

    if (dataList.get(i).Id.equals(c_local)) {

        c_temp = dataList.get(i).temperature;
        c_desc = dataList.get(i).description;
        tempText.setText(c_temp);
        tempText.invalidate();

        editor = pref.edit();           //에디터
        editor.putString("desc", c_desc);
        editor.putString("temp", c_temp); //저장
        editor.commit();

    }

    dataList.clear();
    if (mProgressDialog!=null&&mProgressDialog.isShowing()){
        mProgressDialog.dismiss();
    }

}

});
} catch(Exception ex) {
    ex.printStackTrace();
}

}

}

```

※ 이 외의 모듈에 대한 소스코드는 생략하고, 소스파일로 첨부합니다.