**2017학년도 1학기**

**캡스톤 디자인 교과목**

**캡스톤 디자인 결과보고서**

|  |  |  |
| --- | --- | --- |
| **소 속 (전공)** | 컴퓨터 공학과 | |
| **팀 명** | 실버키퍼 | |
| **지 도 교 수** | 양효식, 임태진 (인) | |
| **팀 장** | 학번: 13011061 | 이름: 김현진 |
| **팀 원** | 학번: 112221 | 이름: 백태영 |
| 학번: 13011058 | 이름: 차민광 |
| 학번: 14011094 | 이름: 최유니 |
| 학번: | 이름: |
| 학번: | 이름: |
| **제 출 일 자** | 2017 . 6 . 9 | |

**세종대학교 공학교육센터**

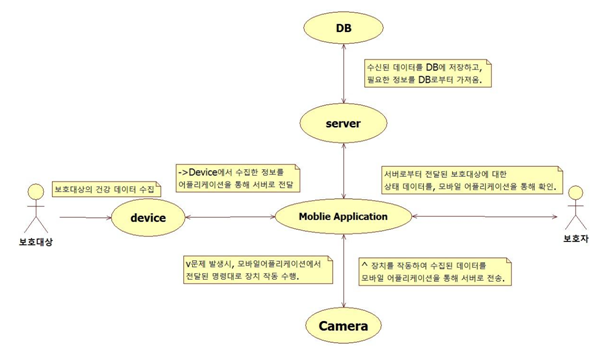
**최종보고서**

**1. 개발 목표**

개발 예정 프로그램인 ‘실버키퍼’는 독거노인의 고독사 방지와 건강관리를 위한 모니터링 솔루션이다. 웨어러블 디바이스를 통해 독거노인의 건강상태를 보다 정확하고 즉각적으로 확인할 수 있으며, 독거노인의 생체 정보를 통해 이상이 감지될 경우, 별도의 카메라 장치를 활용해 독거노인을 모니터링할 수 있다. 카메라 장치는 항상 작동하는 것이 아니라 건강상태에 이상이 있다는 신호가 들어온 경우에 보조적으로 작동함으로써 보호대상인 독거노인이 사생활을 침해받는다는 부담에서 벗어날 수 있도록 한다. 기본적으로 위급한 상황에 처한 독거노인의 정보를 가족, 지인 등 도움을 줄 수 있는 사람에게 신속하게 전달해 도움을 받을 수 있도록 하는 것을 목표로 한다. 딱히 별도의 기기를 작동하지 않아도 기능하도록 하며, 독거노인의 건강 상태에 대한 정보를 확인하도록 하는 것을 목표로 하고 있다.

**2. 설계 사양서 (20 페이지 이내)**

1. **요구사항 분석서**
2. **전체적인 개발방법**



크게 하드웨어, 네트워크, 어플리케이션 파트로 분할된다.

-하드웨어 : 보호대상의 건강 정보 데이터를 수집하고, 이를 어플리케이션으로 전송하는 기능을 담당한다. 보호 대상의 데이터를 수집하기 위해 Wearable Device를 사용했다. 처음에는 직접 아두이노를 이용해 Wearable Device를 제작할 생각이었으나, 후에 시중에 판매중인 Mi Band를 사용하는 것으로 바꿨다. 카메라가 내장된 라즈베리파이 장치를 사용하여 문제 발생시 스트리밍 영상을 서버로 전송했다. 하드웨어 부분은 크게 둘로 나뉜다. 우선 보호대상이 신체에 착용할 Wearable Device로, 처음에는 아두이노로 직접 제작하기로 했으나 Mi Band를 사용하는 것으로 바꾸었다. 이 웨어러블 장치는 보호대상의 심박수를 수집하고, 수집된 정보를 어플리케이션으로 전송한다. 어플리케이션과는 블루투스 송수신을 한다. 다음은 보호대상의 자택에 설치할 카메라, 즉 라즈베리파이다. 위급상황이 발생했을 때 라즈베리파이에 연결한 카메라를 작동시켜 스트리밍 영상을 서버로 전송하는 과정이었다. 그러나 구현 과정에서 영상을 서버로 전송하는 것이 아니라 어플리케이션으로 직접 전송하는 것으로 수정되었다. 또한 초기에는 C언어로 개발하기로 되어있었으나, 파이썬으로 개발하게 되었다.

-네트워크 : 시스템에 사용되는 데이터를 송수신하고 저장한다. 시스템에 사용될 서버를 구현하고, 데이터를 저장할 때 사용하는 데이터베이스를 작성했다. 디바이스를 통해 수집된 정보나 사용자에게 전달될 정보 등, 시스템 전반에서 사용되는 모든 데이터들을 관리한다. 이 파트는 JAVA언어로 구현되었다.

-어플리케이션 : 디바이스에서 수집된 데이터를 수신하여 저장하거나, DB로부터 찾아온 데이터를 사용자에게 전달하는 역할을 담당하는 파트다. 이 어플리케이션은 보호대상의 상태 정보를 디바이스로부터 블루투스 기능을 이용하여 데이터를 수집하는 역할을 한다. 어플리케이션을 통해 사용자의 귀가 여부를 확인하여 서버로 전송하기로 되었었으나, 개발 과정에서 시간부족으로 인해 귀가 여부 사항은 구현하지 못했다. 안드로이드를 이용해 구현했으며, 그와 같은 연유로 네트워크 파트와 마찬가지로 JAVA언어로 구현했다. Mi band와 연동되는 기존의 어플리케이션의 기능을 수정•보완하는 형식으로 구현했다. 보호대상의 생체정보를 열람할 수 있으며, 보호자가 정보를 바로 확인할 수 있도록 푸쉬알림 기능을 추가했다. 원래는 보호대상의 wifi목록 스캔을 통해 귀가여부를 확인하고 그에 대한 정보를 서버로 전송하기로 되었었으나, 구현 과정에서 시간이 부족했기에 이에 대한 내용은 구현하지 못했다.

1. **통신 포맷**

통신은 크게 데이터 통신과 동영상 스트리밍으로 나뉜다. 데이터 통신은 모바일 안드로이드와 통신을 해야 하기 때문에 안드로이드 통신에서 종종 쓰이는 http통신을 사용할 것이다. 또한 동영상 스트리밍에서는 실시간으로 동영상 전송이 가능한 RTSP통신을 사용할 예정이었으나, Flask를 이용하여 영상 스트리밍이 가능한 웹 서버를 구축하여 http통신을 사용하였다.

데이터베이스는 오라클을 이용해 구현했다. 또한 서버는 톰캣을 이용해 WAS를 구현했으며, 어플리케이션의 푸쉬알림을 위해 FCM을 사용했다.

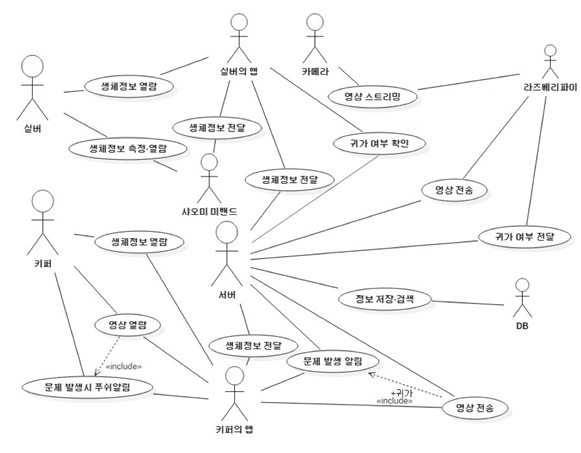
1. **실제 구현**

본래는 보호대상자의 앱에서 Wifi 스캔을 한 후, 그에 따른 결과를 서버로 전송해 귀가여부를 확인하는 기능이 있을 예정이었다. 그러나 시간 관계상 구현하지 못했다. 그렇기 때문에 요구사항 분석서와 제안서에 있던 [귀가여부 확인] 부분은 제대로 개발되지 않았다.

1. **유스케이스 다이어그램**
2. **액터 설명**

|  |  |  |
| --- | --- | --- |
| Actor | 구분 | 설명 |
| 실버 | 시스템 사용자 | 미밴드에 심박수, 걸음수를 제공하는 보호대상자 |
| 키퍼 | 시스템 사용자 | 실버의 상태정보를 앱을 통해 확인하는 보호자 |
| 샤오미 미밴드 | 웨어러블 디바이스 | 실버의 심박수, 걸음수와 같은 생체데이터를 수집해 실버의 앱으로 전달하는 기기 |
| 카메라 | 하드웨어 모듈 | 위급상황시 실버에 대한 영상촬영 |
| 라즈베리파이 | 통신용 하드웨어 장치 | 서버와 연결되어, 실버의 귀가여부를 알림. 실버의 심박수가 평균보다 떨어지거나 올라갈 경우, 카메라로 촬영한 영상을 서버로 전송 |
| 서버 | 시스템 | 정보를 처리하는 중앙관리시스템. 앱과 정보를 주고받아 DB에 저장, 라즈베리파이를 통해 영상을 앱으로 전송하는 기능을 담당. |
| DB | 시스템 | 실버와 키퍼에 대한 정보를 저장. |
| 앱 | 안드로이드 모바일 어플리케이션 | 앱을 통해 실버의 심박수와 걸음수를 알 수 있다.  위급상황시 키퍼용 앱을 통해 영상을 확인할 수 있다. 추가기능의 유무에 따라 키퍼용과 실버용으로 나뉨. |

1. **전체 시스템 유스케이스**



1. 사건흐름

-기본흐름

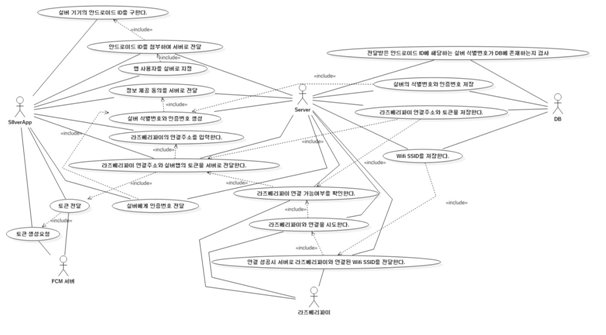
1. 실버의 생체정보를 샤오미 미밴드를 통해 측정한다.
2. 실버의 앱이 샤오미 미밴드에서 측정된 생체정보를 서버로 전송한다.
3. 서버는 전송 받은 생체정보를 DB에 저장한다.
4. 실버와 키퍼는 DB에 저장된 생체정보를 서버를 통해 앱으로 확인한다.

-대안흐름

1. 서버가 실버에게 문제가 발생한 것을 인식한다.
2. 서버가 푸쉬알림을 통해 키퍼의 앱으로 문제발생을 알린다.
3. 키퍼가 앱을 통해 문제 상황을 인식한다.
4. 실버가 외출했을 경우에 별도의 방법으로 실버의 상태를 확인하고 자택에 있을 경우에 키퍼가 영상을 통해 실버의 상태를 확인하게 하려 했으나, 구현하지 못했다.
5. 사전/사후조건

-사전조건

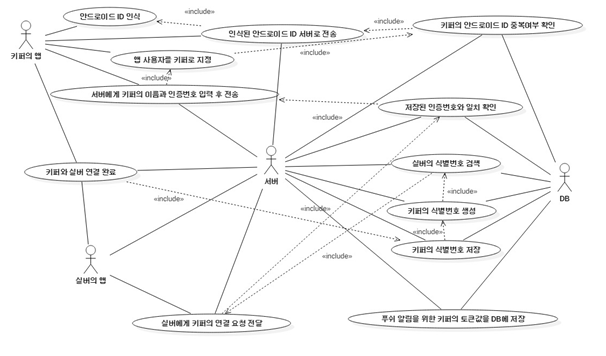
1. 실버와 키퍼 모두 스마트폰을 보유하여야 하며, 해당 시스템의 어플리케이션이 스마트폰에 설치되어있어야 한다.
2. 실버는 샤오미 미밴드를 착용하고 있어야 한다.
3. 실버의 자택은 Wifi 통신이 가능해야 한다.
4. **실버의 초기 인증과정 유스케이스**



1. 사건 흐름

-기본 흐름

1. 실버의 앱은 실버 기기의 안드로이드 ID를 구한다.
2. 실버의 앱은 기기의 안드로이드 ID를 서버로 전송한다.
3. 전달받은 안드로이드 ID에 해당하는 실버 식별번호가 있는지 DB에서 검사한다.
4. 해당하는 실버 식별번호가 없다면, 실버 앱은 역할 선택을 수행한다.
5. 앱 설치시 사용자를 실버로 지정하면, 역할지정에 대한 정보를 서버로 전달한다.
6. 실버의 앱은 서버에 정보 제공 동의를 전달한다.
7. 서버는 실버의 앱으로부터 동의를 전달받으면, 같이 전달받은 안드로이드 ID를 변형하여 실버의 식별번호를 생성하고,함께 인증번호를 생성한다.
8. 앱은 FCM 서버에 토큰 생성을 요청한다.
9. FCM 서버는 앱으로 생성된 토큰을 전달한다.
10. 실버의 앱은 라즈베리파이의 연결주소를 입력한다.
11. 실버의 앱은 라즈베리파이 연결주소와 토큰을 서버로 전달한다.
12. 서버는 라즈베리파이 연결주소를 통해 연결 가능 여부를 확인한다.
13. 연결이 성공하면, 라즈베리파이는 서버로 연결된 Wifi SSID를 전달한다.
14. 서버는 생성된 식별번호와 인증번호, 라즈베리파이의 연결주소, 실버앱의 토큰, 연결된 Wifi SSID를 DB에 저장한다.
15. 서버는 실버의 앱에 인증번호를 전달한다
16. **키퍼의 초기 인증과정 유스케이스**



1. 사건 흐름

-기본 흐름

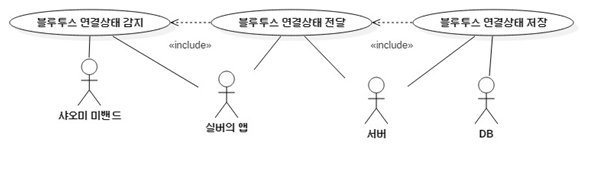
1. 실버기기의 안드로이드 ID를 인식한다.
2. 인식된 ID를 서버로 전송한다.
3. 인식된 ID를 바탕으로 DB에서 중복여부를 확인한다.
4. 앱을 실행한 후에 앱 사용자를 키퍼로 지정한다.
5. 키퍼는 실버의 앱으로 인증번호를 확인, 앱에 이름과 함께 입력 후 전송한다.
6. DB에 저장된 인증번호와 일치여부를 확인한다.
7. 실버에게 키퍼의 연결 요청을 전달한다.
8. 실버가 연결을 승인하면 실버의 식별번호를 검색한다.
9. 검색된 실버의 식별번호에 연결되는 키퍼의 식별번호를 생성한다.
10. DB에 실버의 식별번호와 같은 필드에 키퍼의 식별번호를 저장한다.
11. 실버와 키퍼의 연결이 완료된다.
12. 푸쉬 알림을 위한 키퍼의 토큰값을 DB에 저장한다

-대안 흐름

1. 인증번호를 잘못 입력할 경우 키퍼에게 인증 실패에 대한 메시지를 전달하고 실버/키퍼 선택화면으로 돌아간다.
2. 실버가 이름을 확인 후에 연결을 거절하면 키퍼에게 연결 거절에 대한 메시지를 전달하고 인증번호와 이름 입력화면으로 돌아간다.
3. 사전/사후 조건

-사전 조건

1. 실버의 초기 인증과정과 키퍼의 초기 인증과정은 함께 병행되어야 한다.
2. 인증을 승인하지 않는 과정은 모두 대안흐름으로 본다.
3. 샤오미 미밴드의 이상상태 확인 유스케이스



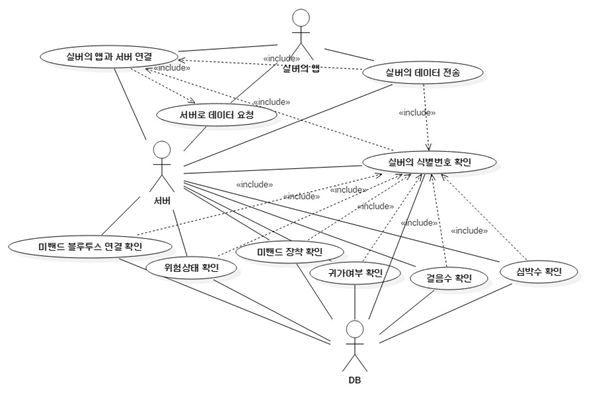
1. 사건흐름

-기본 흐름

1. 실버의 앱은 샤오미 미밴드의 블루투스 연결상태를 감지한다.
2. 실버의 앱은 서버로 샤오미 미밴드의 블루투스 연결상태를 전달한다.
3. 서버는 미밴드의 블루투스 연결상태를 DB로 저장한다.
4. 사전/사후조건

-사전조건

1. 샤오미 미밴드를 실버가 착용하고 있어야 한다.
2. **실버의 정보확인 유스케이스**



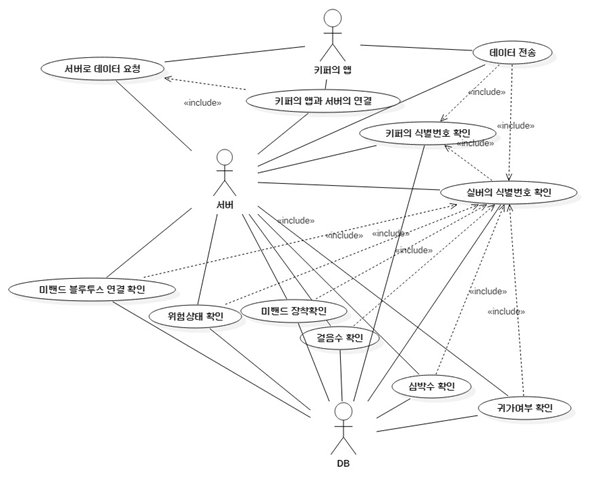
1. 사건 흐름

-기본 흐름

1. 실버의 앱이 서버에 데이터를 요청한다.
2. 실버의 앱과 서버가 연결된다.
3. 서버는 실버의 식별번호를 확인한다.
4. DB에 저장된 실버의 심박수, 걸음수, 위험상태를 서버가 검색하고 가지고 온다.
5. 미밴드의 블루투스 연결상태와 장착상태를 서버로 가지고 온다.
6. 실버의 심박수, 걸음수, 위험상태, 장착상태가 실버의 앱에 전송된다.
7. 사전/사후 조건

-사전 조건

1. 실버의 앱으로 보여질 정보의 저장을 위해 [DB 데이터 저장]이 선행되어야 한다.
2. 미밴드의 블루투스 연결상태 확인을 위해 [샤오미 미밴드의 이상상태 확인]이 선행되어야 한다.
3. 실버는 주기적으로 샤오미 미밴드를 충전해주어야 한다.
4. DB에서 서버와 연결된 앱의 토큰을 확인해야 한다.
5. **키퍼의 정보확인 유스케이스**



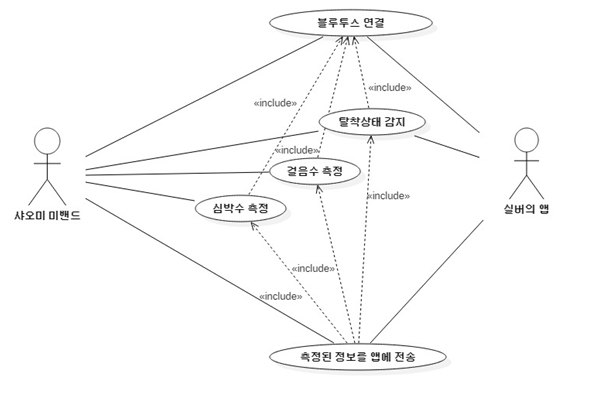
1. 사건흐름

-기본 흐름

1. 키퍼의 앱이 서버로 데이터를 요청한다.
2. 키퍼의 앱과 서버가 연결된다.
3. 키퍼의 식별번호를 확인한 후 그에 대응하는 실버의 식별번호를 확인한다.
4. DB는 수집되어 있는 실버의 심박수, 걸음수, 위험상태를 서버에 전달한다.
5. 미밴드의 블루투스 연결상태와 장착상태를 서버로 가지고 온다.
6. 실버의 심박수, 걸음수, 위험상태, 장착상태가 실버의 앱에 전송된다.
7. 사전/사후 조건

-사전 조건

1. 실버의 앱으로 보여질 정보의 저장을 위해 [DB 데이터 저장]이 선행되어야 한다.
2. 미밴드의 블루투스 연결상태 확인을 위해 [샤오미 미밴드의 이상상태 확인]이 선행되어야 한다.
3. DB에서 서버와 연결된 앱의 토큰을 확인해야 한다.
4. **샤오미 미밴드의 생체정보 수집 유스케이스**



1. 사건흐름

-기본 흐름

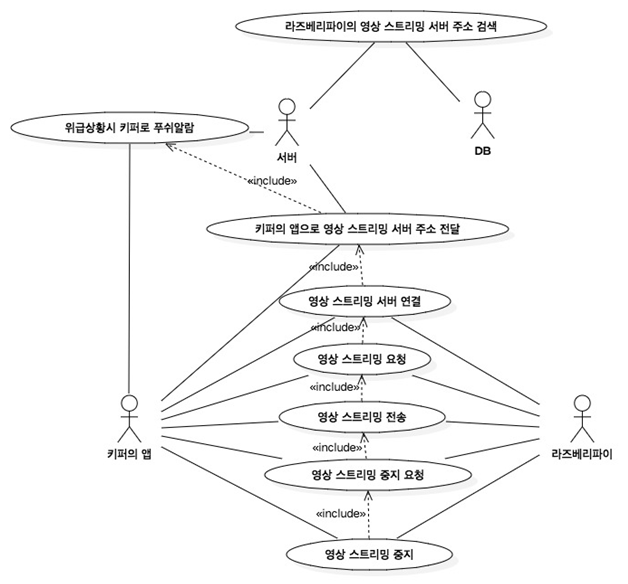
1. 샤오미 미밴드와 실버의 앱이 설치된 스마트폰을 블루투스로 연결 한다.
2. 샤오미 미밴드는 실버의 손목을 통해 실버의 10초에 한번씩 심박수를 측정한다.
3. 심박수 측정과 함께 탈착상태를 감지한다.
4. 샤오미 미밴드가 걸음수를 측정한다.
5. 샤오미 미밴드를 통해 측정된 생체정보는 샤오미 미밴드와 블루투스 연결되어 있는 실버의 앱으로 전달된다
6. 사전/사후 조건

-사전 조건

1. 샤오미 미밴드와 실버의 앱은 블루투스를 통해 연결되어 있다.
2. 초기 설정을 위해 Mi 피트 앱 설치 후에 펌웨어 업그레이드를 진행한다

-사후 조건

1. 생체정보 저장을 위해 DB 데이터 저장 유스케이스가 후행되어야 한다.
2. **라즈베리파이의 영상 스트리밍 유스케이스**



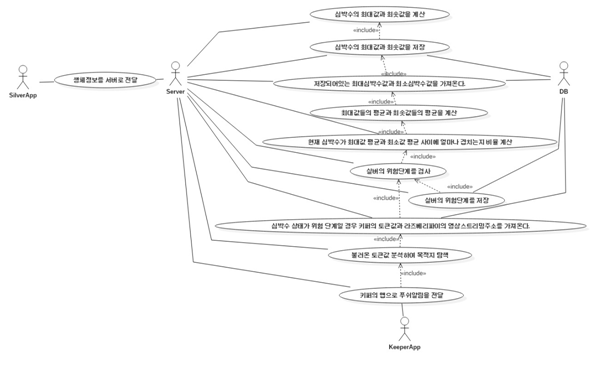
1. 사건흐름

-기본흐름

1. 서버는 DB에 저장된 라즈베리파이의 영상 스트리밍 서버 주소를 검색한다.
2. 위급상황 발생시 키퍼로 영상 스트리밍 서버주소와 함께 푸쉬알림을 보낸다.
3. 키퍼는 푸쉬알람을 통해 영상 스트리밍 서버에 연결을 요청한다.
4. 라즈베리파이는 키퍼에게 영상 스트리밍을 전송한다.
5. 키퍼는 영상 스트리밍을 받고 있는 중에 스트리밍 중지 요청을 할 수 있다.
6. 라즈베리파이는 키퍼로부터 스트리밍 중지요청을 받아 스트리밍을 중지한다.
7. 사전/사후 조건

-사전 조건

1. 카메라는 라즈베리파이에 장착된 상태로 실버의 집안에 설치되어있다.
2. 문제발생에 대해 [문제 발생 처리]가 선행되어야 한다.
3. 영상 스트리밍은 키퍼의 앱에서 중지 버튼을 누르지 않더라도 5분이 지나면 자동으로 꺼진다.
4. **문제발생 처리 유스케이스**



1. 사건흐름

-기본흐름

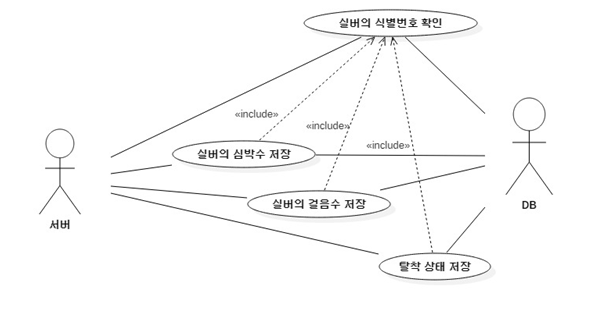
1. 서버는 새로고침 버튼을 누를 때마다 수집된 심박수값들 중 최솟값과 최댓값을 계산하여 DB에 저장한다.
2. 최솟값과 최대값들의 평균을 구하고, 저장된 평균의 기준수치와 일치하는 비율을 계산한다. 70%~100%인 경우에는 안전, 50~69%인 경우 경고, 49% 이하인 경우 위험으로 판단한다.
3. 서버에서 계산한 결과를 토대로, 실버의 상태를 검사한다.
4. 서버는 실버의 상태를 DB에 저장한다.
5. 상태가 ‘위험’에 해당할 경우 문제가 발생했음을 인식한다.
6. 서버는 키퍼의 토큰값과, 라즈베리파이의 MAC 주소를 DB로부터 가져온다.
7. 서버는 키퍼의 토큰값을 통해 푸시알림을 보낼 목적지를 인식한다.
8. 서버는 키퍼의 앱으로 푸쉬알림을 보내 스트리밍 여부를 묻는다.
9. 사전/사후조건

-사전 조건

1. 데이터 분석을 위해 [샤오미 미밴드의 생체정보 수집]이 선행되어야 한다.

-사후 조건

1. 영상 스트리밍을 제공하기 위해 [라즈베리파이의 영상 스트리밍]이 후행된다.
2. **DB 데이터 저장 유스케이스**



1. 사건흐름

-기본 흐름

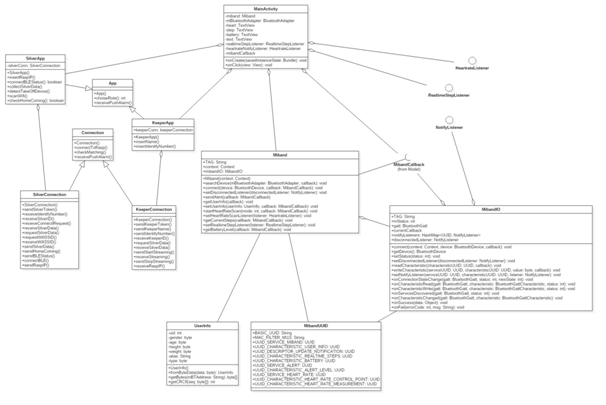
1. 정보를 저장하기 위해 서버는 DB에 저장된 실버의 식별번호를 확인한다.
2. 서버는 실버의 심박수를 DB에 저장한다.
3. 서버는 실버의 걸음수를 DB에 저장한다.
4. 서버는 실버의 탈착 상태를 DB에 저장한다.
5. 사전/사후 조건

-사전 조건

1. 서버가 저장할 심박수, 걸음수 측정을 위해 [샤오미 미밴드의 생체정보 수집]이 선행되어야 한다.
2. **클래스 다이어그램 및 데이터 맵 - 클래스 다이어그램과 데이터 맵의 정리**
3. **클래스 다이어그램**
   1. 클래스 - 앱
      1. 연산

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스 | 연산 | 파라메터 | 반환값 | 설명 |
| App | choiseRole | x | boolean | 인증과정에서 역할을선택한다. |
| App | receivePushAlarm | x | void | 서버에서 보낸 푸쉬알림을 받는다. |
| App | requestToken | x | String | FCM서버로부터 토큰값을 요청한다. |
| App | checkRegist | x | int | 초기인증이 완료됨을 서버로 전달한다. |
| App | getAndroidID |  | String | 해당 기기의 안드로이드 ID를 받아온다. |
| SilverApp | insertRaspIP | raspIP: byte[] | byte[] | 라즈베리파이의 영상 서버주소를 입력한다. |
| SilverApp | connectBLEStatus | x | boolean | 블루투스 연결상태를 감지한다. |
| SilverApp | collectSilverData | x | void | 미밴드에서 실버의 앱으로 데이터를 수집한다. |
| SilverApp | detectTakeOffDevice | x | boolean | 미밴드의 탈착상태를 감지한다. |
| SilverApp | scanWifi | x | boolean | Wifi목록을 스캔한다. |
| SilverApp | checkHomeComing | x | boolean | 귀가여부를 판단한다. |
| SilverApp | checkRegist | x | int | 초기인증이 완료됨을 서버로 전달한다. |
| KeeperApp | insertName | keeperName: String | String | 키퍼의 이름을 입력한다. |
| KeeperApp | insertIdentifyNumber | identifyNumber: String | String | 키퍼의 인증번호를 입력한다. |
| Connection | connectToServer | x | boolean | 앱과 서버를 연결한다. |
| Connection | connectToRasp | x | boolean | 앱과 영상 서버를 연결한다. |
| Connection | checkMatching | x | boolean | 서버에서 키퍼의 앱과 실버의 앱의 연결 완료를 확인한다 |
| Connection | receivePushAlarm | x | void | 서버에서 보낸 푸쉬알림을 받는다. |
| Connection | checkRegist | x | int | 초기인증이 완료됨을 서버로 전달한다. |
| SilverConnection | sendSilverToken | silverToken: String | int | 실버의 토큰을 서버로 전달한다. |
| SilverConnection | receiveIdentifyNumber | identifyNumber: int | int | 서버로부터 인증번호를 전달받는다. |
| SilverConnection | receiveSilverID | silverID: int | int | 서버로부터 실버의 식별번호를 전달받는다 |
| SilverConnection | receiveConnectRequest | connect: boolean | void | 서버에서 키퍼의 연결요청을 전달한다. |
| SilverConnection | receiveSilverData | x | int | 서버에서 실버의 데이터를 전달받는다. |
| SilverConnection | requestSilverData | x | String | 서버로 실버의 데이터를 요청한다. |
| SilverConnection | sendSilverData | silverVO:SilverVO | int | 서버로 실버의 데이터를 전달한다. |
| SilverConnection | requestWifiSSID | wifiSSID: string | void | 서버로 Wifi의 SSID를 요청한다. |
| SilverConnection | receiveWifiSSID | x | String | 서버로부터 Wifi의 SSID를 전달받는다. |
| SilverConnection | sendHomeComing | x | boolean | 서버로 실버의 귀가여부를 전달한다. |
| SilverConnection | sendBLEStatus | x | boolean | 블루투스 연결상태를 전달한다. |
| SilverConnection | connectBLE | x | int | 블루투스를 연결한다. |
| SilverConnection | sendRaspIP | x | int | 서버로 라즈베리파이의 IP주소를 전달한다. |
| KeeperConnection | sendKeeperToken | x | int | 서버로 키퍼의 토큰값을 전달한다. |
| KeeperConnection | sendKeeperName | keeperName:String,  number:int | int | 키퍼의 이름을 서버로 전달한다. |
| KeeperConnection | sendIdentifyNumber | x | int | 키퍼의 인증번호를 서버로 전달한다. |
| KeeperConnection | receiveKeeperID | keeperID:String | void | 서버로부터 키퍼의 식별번호를 전달받는다. |
| KeeperConnection | requestSilverData | silverData: silverVO | x | 서버로 실버의 데이터를 요청한다. |
| KeeperConnection | receiveSilverData | x | Int | 서버로부터 실버의 데이터를 전달받는다. |
| KeeperConnection | sendStartStreaming | x | void | 라즈베리파이로 영상 스트리밍의 시작을 요청한다. |
| KeeperConnection | receiveStreaming | x | void | 라즈베리파이에서 영상스트리밍을 전송받는다. |
| KeeperConnection | sendStopStreaming | x | void | 라즈베리파이로 영상스트리밍의 중지요청을 보낸다. |
| KeeperConnection | receiveRaspIP |  | void | 서버로부터 라즈베리파이의 ip주소를 전달받는다. |

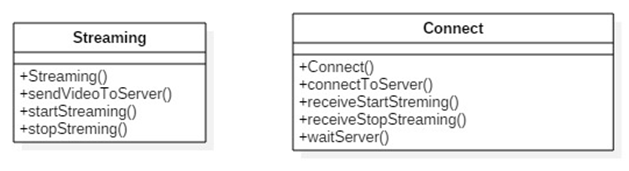
* + 1. 다이어그램



* 1. 클래스 - 라즈베리파이
     1. 연산

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스 | 연산 | 파라메터 | 반환값 | 설명 |
| Streaming | sendVideoToApp | x | void | 스트리밍한 영상을 앱으로 전송한다.(RTSP) |
| Streaming | startStreaming | x | void | 영상스트리밍을 시작한다. |
| Streaming | stopStreaming | x | void | 영상스트리밍을 중지한다. |
| Connect | connectToServer | x | boolean | 서버와 연결한다(RTSP) |
| Connect | connectToApp | x | boolean | 앱과 연결한다(RTSP) |
| Connect | receiveStartStreaming | x | boolean | 영상스트리밍 시작요청을 받는다. |
| Connect | receiveStopStreaming | x | boolean | 영상스트리밍 중지요청을 받는다 |
| Connect | waitClient | x | boolean | 클라이언트가 연결될 때까지 기다린다. |

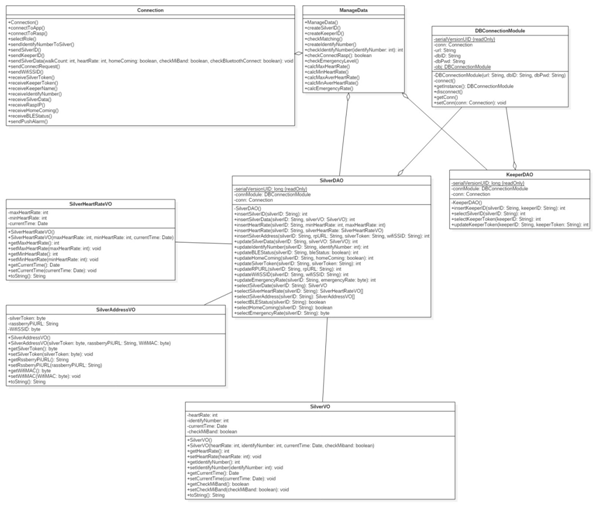
* + 1. 다이어그램



* 1. 클래스 - 서버
     1. 연산

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스 | 연산 | 파라메터 | 반환값 | 설명 |
| ManageData | createSilverID | X | string | 실버의 식별번호를 생성한다. |
| ManageData | createKeeperID | X | string | 키퍼의 식별번호를 생성한다. |
| ManageData | checkMatching | X | String | 키퍼의 앱과 실버의 앱의 연결이 완료되었는지 확인한다. |
| ManageData | checkSilverID | silverID:String |  | 실버 식별번호가 존재하는지 검사한다. |
| ManageData | checkKeeperID | keeperID:String |  | 키퍼 식별번호가 존재하는지 검사한다. |
| ManageData | saveToken | token:String | int | 토큰값을 저장한다. |
| ManageData | createIdentifyNumber | X | int | 실버의 인증번호를 생성한다. |
| ManageData | checkIdentifyNumber | identifyNumber: int | int | 키퍼의 앱으로 보내진 인증번호가 저장된 인증번호와 일치하는지 확인한다. |
| ManageData | checkConnectRasp | X | void | 라즈베리파이와의 연결여부를 확인한다. |
| ManageData | checkEmergencyLevel | emergencyRate: int | int | 심박수의 위험단계를 확인한다. |
| ManageData | calcMaxHeartRate | heartRate: int | int | 심박수의 최대값을 계산한다. |
| ManageData | calcMinHeartRate | heartRate: int | int | 심박수의 최소값을 계산한다. |
| ManageData | calcMaxAverHeartRate | maxHeartRate: int | int | 심박수의 최대값들의 평균을 계산한다. |
| ManageData | calcMinAverHeartRate | minHeartRate: int | int | 심박수의 최소값들의 평균을 계산한다. |
| ManageData | calcEmergencyRate | heartRate: int | int | 현재 심박수들이 최대최소의 평균범위 내에 얼마나 분포하고 있는지 계산한다. |
| Connection | connectToApp | identifyNumber: int | void | 앱과 서버를 연결한다. 필요할 때에 식별번호로 역할을 구분한다. |
| Connection | connectToRasp | X | void | 라즈베리파이와 서버를 연결한다. |
| Connection | selectRole | X | boolean | 앱 사용자를 실버 혹은 키퍼로 지정한다. |
| Connection | sendIdentifyNumberToSilver | identifyNumber: int | void | 실버에게 인증번호를 전달한다. |
| Connection | sendSilverID | silverID: string | void | 실버에게 실버의 식별번호를 전달한다. |
| Connection | sendKeeperID | keeperID: string | void | 키퍼에게 키퍼의 식별번호를 전달한다. |
| Connection | sendSilverData | heartRate: int, walkCount: int, homeComing: boolean, checkEmergency: int, checkMiBand: boolean, checkBluetoothConnect: boolean | void | 서버에서 앱으로 실버의 데이터를 전달한다. |
| Connection | sendConnectRequest | identifyNumber: int, keeperName: string | void | 실버에게 키퍼가 보낸 이름과 인증번호와 함께 연결 요청을 전달한다. |
| Connection | sendWifiSSID | wifiSSID : String | void | 실버의 앱으로 Wifi SSID를 전달한다. |
| Connection | receiveSilverToken |  | void | 실버에게서 실버의 토큰값을 전달받는다. |
| Connection | receiveKeeperToken |  | void | 키퍼에게서 키퍼의 토큰값을 전달받는다. |
| Connection | receiveKeeperName |  | void | 키퍼에게 키퍼의 이름을 전달받는다. |
| Connection | receiveIdentifyNumber |  | void | 키퍼에게 인증번호를 전달받는다. |
| Connection | receiveAndroidID |  |  | 실버/키퍼로부터 해당 기기의 안드로이드 ID를 전달받는다. |
| Connection | receiveRole |  |  | 역할 선택 내용을 전달받는다. |
| Connection | receiveRegist |  |  | 인증이 완료됨을 전달받는다. |
| Connection | receiveSilverData |  | void | 실버의 앱으로부터 데이터를 받아온다. |
| Connection | receiveRaspIP | raspIP: byte[] | void | 라즈베리파이의 영상 주소를 전달받는다. |
| Connection | receiveHomeComing | homeComing: boolean | void | 귀가여부를 전달받는다. |
| Connection | receiveBLEStatus | checkBluetoothConnect: boolean | void | 실버의 앱에서 블루투스 연결상태를 전달받는다. |
| Connection | sendPushAlarm |  |  | 앱으로 푸시알림을 전달한다. |
| SilverDAO | insertSilverID | silverID: string | int | 실버의 식별번호를 저장한다. |
| SilverDAO | insertSilverData | silverID: String, silverVO: silverVO | int | 실버의 데이터를 저장한다. |
| SilverDAO | insertHeartRate | silverID: String, minHeartRate: int, maxHeartRate: int, silverHeartRate: SilverHeartRateVO | int | 실버의 심박수를 저장한다. |
| SilverDAO | insertSilverAddress | silverID: String, rpURL: String, silverToken: String, wifiSSID: String | int | 라즈베리파이 URL, 토큰값, wifi의 SSID를 저장한다. |
| SilverDAO | updateSilverData | silverID: String, silverVO: SilverVO | int | 실버의 데이터를 갱신한다. |
| SilverDAO | updateIdentifyNumber | silverID: String, identifyNumber: int | int | 인증번호를 갱신한다. |
| SilverDAO | updateBLEStatus | silverID: String, bleStatus: boolean | int | 미밴드의 블루투스 연결상태를 갱신한다. |
| SilverDAO | updateHomeComing | silverID: String, homeComing: boolean | int | 실버의 귀가상태를 갱신한다. |
| SilverDAO | updateSilverToken | silverID: String, silverToken: String | int | 실버에게 부여된 토큰값을 갱신한다. |
| SilverDAO | updateRPURL | silverID: String, rpURL: String | int | 라즈베리파이의 서버 주소를 갱신한다. |
| SilverDAO | updateWifiSSID | silverID: String, wifiSSID: String | int | 공유기의 SSID정보를 갱신한다. |
| SilverDAO | updateEmergencyRate | silverID: String, emergencyRate: byte[] | int | 위험상태를 갱신한다. |
| SilverDAO | selectSilverData | silverID: String | SilverVO | 실버의 데이터를 검색한다. |
| SilverDAO | selectSilverHeartRate | silverID: String | SilverHeartRateVO[] | 실버의 심박수를 검색한다. |
| SilverDAO | selectSilverAddress | silverID: String | SilverAddressVO[] | 실버의 토큰값, 라즈베리파이의 서버 주소, 공유기의 MAC주소를 검색한다. |
| SilverDAO | selectBLEStatus | silverID: String | boolean | 미밴드의 블루투스 연결 여부를 검색한다. |
| SilverDAO | selectHomeComing | silverID: String | boolean | 실버의 귀가여부를 검색한다. |
| SilverDAO | selectEmergencyRate | silverID: String | byte[] | 위험상태를 검색한다. |
| SilverDAO | checkIdentifyNumber | identifyNumber:int | String | 입력받은 인증번호에 해당하는 실버 정보가 있는지 확인한다. |
| SilverVO | getHeartRate | X | int | 실버의 심박수를 받는다. |
| SilverVO | setHeartRate | heartRate: int | void | 실버의 심박수를 설정한다. |
| SilverVO | getIdentifyNumber | X | int | 인증번호를 받는다. |
| SilverVO | setIdentifyNumber | identifyNumber: int | void | 인증번호를 설정한다. |
| SilverVO | getCurrentTime | X | Date | 현재시간을 받는다. |
| SilverVO | setCurrentTime | currentTime: Date | void | 현재시간을 설정한다. |
| SilverVO | getCheckMiBand | X | boolean | 샤오미 미밴드의 장착여부를 받는다. |
| SilverVO | setCheckMiBand | checkMiBand: boolean | void | 샤오미 미밴드의 장착여부를 설정한다. |
| SilverVO | toString | X | string | 전체 데이터를 반환한다. |
| SilverHeartRateVO | getMaxHeartRate | X | int | 심박수의 최대값을 받는다. |
| SilverHeartRateVO | setMaxHeartRate | maxHeartRate: int | void | 심박수의 최대값을 설정한다. |
| SilverHeartRateVO | getMinHeartRate | X | int | 심박수의 최소값을 받는다. |
| SilverHeartRateVO | setMinHeartRate | minHeartRate: int | void | 심박수의 최소값을 설정한다. |
| SilverHeartRateVO | getCurrentTime | X | Date | 현재시간을 받는다. |
| SilverHeartRateVO | setCurrentTime | currentTime: Date | void | 현재시간을 설정한다. |
| SilverHeartRateVO | toString | X | string | 전체 데이터를 반환한다. |
| silverAddressVO | getSilverToken | X | byte[] | 실버의 토큰값을 받는다. |
| silverAddressVO | setSilverToken | silverToken: byte[] | void | 실버의 토큰값을 설정한다. |
| silverAddressVO | getRassberryPiURL | X | String | 라즈베리파이의 서버 주소를 받는다. |
| silverAddressVO | setRassberryPiURL | rassberryPiURL: String | void | 라즈베리파이의 서버 주소를 설정한다. |
| silverAddressVO | getWifiSSID | X | byte[] | 공유기의 SSID를 받는다. |
| silverAddressVO | setWifiSSID | wifiSSID: byte[] | void | 공유기의 SSID를 설정한다. |
| silverAddressVO | toString | X | String | 전체 데이터를 반환한다. |
| KeeperDAO | insertKeeperID | silverID: String, keeperID: String | int | 키퍼의 식별번호를 저장한다 |
| KeeperDAO | selectSilverID | silverID: String | int | 실버의 식별번호를 검색한다. |
| KeeperDAO | selectKeeperToken | keeperID: String | int | 키퍼의 토큰값을 검색한다. |
| KeeperDAO | updateKeeperToken | keeperID: String, keeperToken: String | int | 키퍼의 토큰값을 갱신한다. |
| DBConnectionModule | getInstance | X | DBConnectionModule | 해당 객체에 보안을 건다. |
| DBConnectionModule | disconnect | X | void | DB와의 연결을 끊는다. |
| DBConnectionModule | getConn | X | void | DB와의 연결을 받는다. |
| DBConnectionModule | setConn | conn: Connection | void | DB와의 연결을 설정한다. |

* + 1. 다이어그램



1. **데이터맵**

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 실버의 초기 인증과정 | |
| Data Name | 자료형 | 크기 | 설명 |
| silverToken | byte[] | 6 | 실버의 기기의 토큰값. |
| silverID | String | 15 | 실버의 식별번호 |
| identifyNumber | int | 4 | 실버와 키퍼의 연결을 위한 인증번호 |
| rassberriPiURL | String | 20 | 라즈베리파이 연결주소. |
| rassberryPiMAC | byte[] | 6 | 라즈베리파이 MAC주소. |
| wifiSSID | String | 20 | 와이파이 SSID값. |

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 키퍼의 초기 인증과정 | |
| Data Name | 자료형 | 크기 | 설명 |
| silverID | string | 15 | 실버의 식별번호 |
| keeperID | string | 15 | 키퍼의 식별번호 |
| keeperToken | byte[] | 6 | 키퍼의 토큰값. |
| identifyNumber | int | 4 | 실버와 키퍼의 연결을 위한 인증번호 |
| keeperName | string | 15 | 키퍼의 이름 |
| connectSilverKeeper | bool | 1 | 실버와 키퍼의 연결 여부 |

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 샤오미 미밴드의 이상상태 확인 | |
| Data Name | 자료형 | 크기 | 설명 |
| checkBluetoothConnect | bool | 1 | 미밴드의 블루투스 연결 여부 확인 |
| checkMiBand | bool | 1 | 미밴드의 탈착여부 확인 |

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 실버의 정보 확인 | |
| Data Name | 자료형 | 크기 | 설명 |
| heartRate | int | 4 | 실버의 심박수 |
| walkCount | int | 4 | 실버의 걸음수 |
| homeComing | bool | 1 | 실버의 귀가여부 확인 |
| silverID | string | 15 | 실버의 식별번호 |
| rassberryPiMACAddress | byte[] | 6 | 라즈베리파이의 MAC주소 |
| wifiSSIDAddress | string | 15 | 라즈베리파이와 연결된 공유기의 SSID |
| emergencyLevel | int | 4 | 실버의 위험단계 |

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 키퍼의 정보 확인 | |
| Data Name | 자료형 | 크기 | 설명 |
| heartRate | int | 4 | 실버의 심박수 |
| walkCount | int | 4 | 실버의 걸음수 |
| homeComing | bool | 1 | 실버의 귀가여부 |
| keeperID | string | 15 | 키퍼의 식별번호 |
| silverID | string | 15 | 실버의 식별번호 |
| checkMiBand | bool | 1 | 미밴드의 탈착여부 확인 |
| checkBluetoothConnect | bool | 1 | 미밴드의 블루투스 연결 여부 확인 |
| emergencyLevel | int | 4 | 실버의 위험단계 |

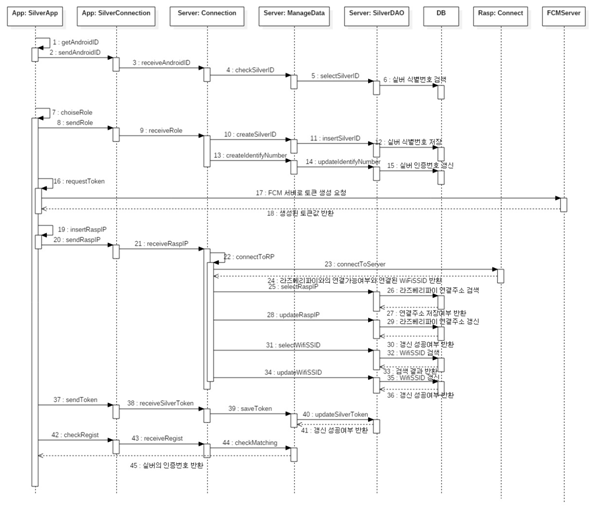
|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 샤오미 미밴드의 생체정보 수집 | |
| Data Name | 자료형 | 크기 | 설명 |
| checkBluetoothConnect | bool | 1 | 미밴드의 블루투스 연결 여부 확인 |
| silverID | string |  | 실버의 식별번호 |
| heartRate | int | 4 | 실버의 심박수 |
| walkCount | int | 4 | 실버의 걸음수 |
| checkMiBand | boolean | 1 | 실버의 샤오미 미밴드 탈착 여부 |

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 라즈베리파이의 영상 스트리밍 | |
| Data Name | 자료형 | 크기 | 설명 |
| wifiMACAddress | byte[] | 6 | 라즈베리파이와 연결된 공유기의 MAC주소 |
| rassberryPiMACAddress | byte[] | 6 | 라즈베리파이의 MAC주소 |
| rassberryPiIP | struct in\_addr | 32bit | 라즈베리파이의 IP주소 |

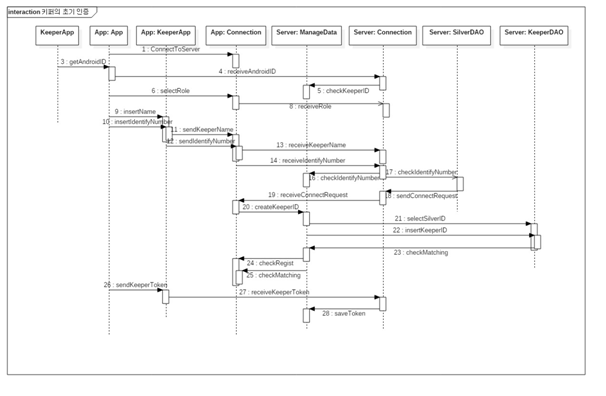
|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | 문제 발생 처리 | |
| Data Name | 자료형 | 크기 | 설명 |
| AverHeartRate | int | 4 | 실버의 평균 심박수 |
| heartRate | int | 4 | 실버의 심박수. |
| streamRequest | boolean | 1 | 키퍼의 영상 열람 확인 여부 |
| maxHeartRate | int | 4 | 실버의 최대 심박수 |
| minHeartRate | int | 4 | 실버의 최소 심박수. |
| maxAvgHeartRate | int | 4 | 실버의 최대 심박수의 평균. |
| minAvgHeartRate | int | 4 | 실버의 최소 심박수의 평균. |
| silverID | string | 15 | 실버의 식별번호 |
| keeperID | string | 15 | 키퍼의 식별번호 |
| emergencyRate | int | 4 | 실버의 위험도 비율. |
| emergencyLevel | byte | 3 | 실버의 위험단계. |
| rassberriPiURL | String |  | 라즈베리파이의 연결주소 |
| keeperToken | byte[] | 6 | 키퍼의 토큰값 |

|  |  |  |  |
| --- | --- | --- | --- |
| use case 명 | | DB 데이터 저장 | |
| Data Name | 자료형 | 크기 | 설명 |
| silverID | string | 15 | 실버의 식별번호 |
| heartRate | int | 4 | 실버의 심박수 |
| walkCount | int | 4 | 실버의 걸음수 |
| checkMiBand | boolean | 1 | 실버의 샤오미 미밴드 탈착 여부 |

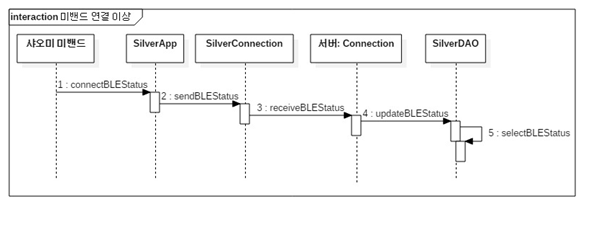
1. **시퀀스 다이어그램 - 제시한 시퀀스 다이어그램의 정리**
2. **실버의 초기 인증과정**



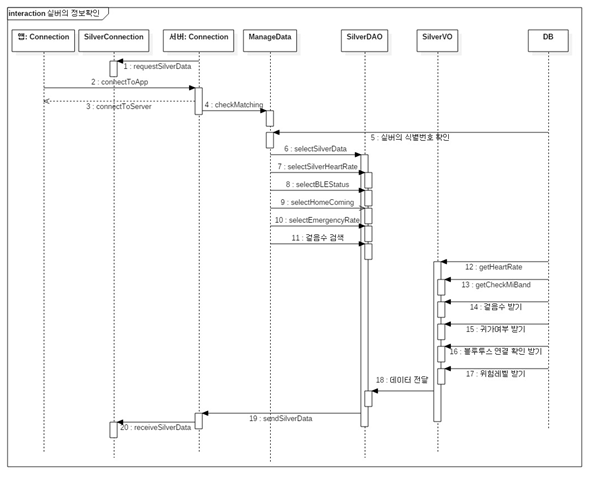
1. SilverApp이 안드로이드ID 습득을 위해 getAndroidID 메소드를 호출한다.
2. 서버로 실버의 안드로이드ID를 전달하기 위해 sendAndroidID메소드를 호출한다.
3. 서버는 안드로이드ID를 전달받기 위해 receiveAndroidID 메소드를 호출한다.
4. 전달받은 안드로이드ID에 해당하는 실버 식별번호가 있는지 검사하기 위해 checkSilverID 메소드를 호출한다.
5. 실버 ID 검사를 위해 SilverDAO 클래스의 selectSilverID 메소드를 호출한다.
6. DB에서 실버 식별번호를 검사한다.
7. 해당하는 실버가 존재하지 않으면, 역할선택을 위해 choiseRole 메소드를 호출한다.
8. 역할을 실버로 선택하면, 서버로 역할 전달을 위해 sendRole 메소드를 호출한다.
9. 서버의 Connection은 역할전달을 받기 위해 receiveRole메소드를 호출한다.
10. 실버의 식별번호를 생성하기 위해 createSilverID메소드를 호출한다.
11. 생성한 실버 식별번호를 저장하기 위해 SilverDAO의 insertSilverID 메소드를 호출한다.
12. 실버 식별번호를 저장한다.
13. 실버 인증번호를 생성하기 위해 createIdentifyNumber메소드를 호출한다.
14. 실버 인증번호를 갱신하기 위해 updateIdentfiyNumber 메소드를 호출한다.
15. 실버 인증번호를 갱신한다.
16. 후에 푸쉬알림을 위해 실버 기기의 토큰값을 받는 requestToken메소드를 호출한다.
17. FCM 서버로 토큰 생성을 요청한다.
18. 생성된 토큰을 받는다.
19. SilverApp은 사용자가 라즈베리파이의 연결주소를 입력하는 insertRaspIP 메소드를 호출한다.
20. 입력한 라즈베리파이 연결주소를 서버로 전달하려 sendRaspIP 메소드를 호출한다.
21. 서버는 라즈베리파이 연결주소를 받는 receiveRaspIP 메소드를 호출한다.
22. 라즈베리파이 연결주소가 유효한지 확인하기 위해 connectToRP 메소드를 호출한다.
23. 라즈베리파이는 서버와의 연결을 위해 connectToServer 메소드를 호출한다.
24. 라즈베리파이와의 연결 가능여부와, 라즈베리파이와 연결된 WifiSSID값을 반환한다.
25. 라즈베리파이 연결주소가 중복되는지 확인하기 위해 selectRaspIP메소드를 호출한다.
26. DB에서 해당 라즈베리파이 연결주소를 검색한다.
27. 해당 라즈베리파이 연결주소의 저장여부를 반환한다.
28. 라즈베리파이 연결주소값이 없다면, 갱신을 위해 updateRaspIP 메소드를 호출한다.
29. DB에서 해당 라즈베리파이의 연결주소값을 갱신한다.
30. 갱신 성공 여부를 반환한다.
31. WifiSSID가 중복되는지를 확인하기 위해 selectRaspIP메소드를 호출한다.
32. DB에서 해당 WifiSSID를 검색한다.
33. 해당 WifiSSID의 저장여부를 반환한다.
34. 해당 WifiSSID값이 없다면, 갱신을 위해 updateSSID 메소드를 호출한다.
35. DB에서 해당 WifiSSID값을 갱신한다.
36. 갱신 성공 여부를 반환한다.
37. 수집한 토큰값을 서버로 전달하기 위해 sendToken메소드를 호출한다.
38. 서버의 Connection은 토큰값을 받기 위해 receiveToken메소드를 호출한다.
39. 서버의 ManageData는 토큰값을 저장하기 위해 saveToken 메소드를 호출한다.
40. 해당 토큰값을 갱신하기 위해 SilverDAO의 updateSilverToken메소드를 호출한다.
41. 갱신 성공여부를 반환한다.
42. 실버 인증이 완료되었는지 확인하기 위해 checkRegist 메소드를 호출한다.
43. 실버 인증 완료여부를 서버로 전달하기 위해 서버의 receiveRegist 메소드를 호출한다.
44. 인증 완료여부를 확인, 인증번호를 전달하기 위해 checkRegist 메소드를 호출한다.
45. 실버의 앱으로 인증번호가 전달된다.
46. **키퍼의 초기 인증과정**



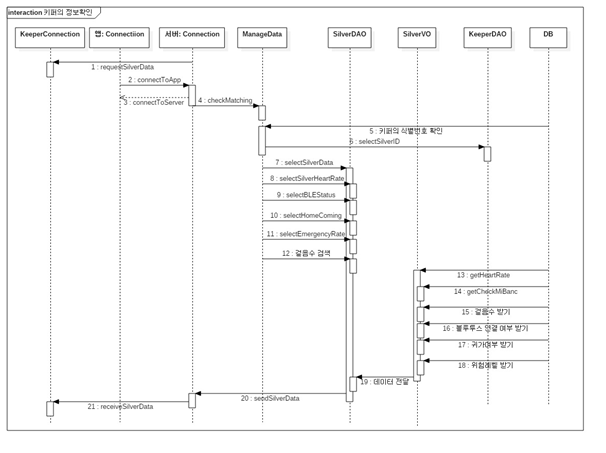
1. ConnectToServer 메소드로 키퍼앱과 서버를 연결한다.
2. 키퍼의 스마트폰에 대한 안드로이드ID를 인식하기 위해 getAndroidID를 호출한다.
3. receiveAndroidID로 인식된 안드로이드ID를 서버로 전송한다.
4. DB에 저장된 안드로이드ID와 인식된 안드로이드ID의 중복 여부를 확인한다.
5. selectRole로 역할을 키퍼로 선택한다.
6. 서버에서 역할에 대한 정보를 전송받는다.
7. insertName 메소드를 이용해 앱에서 키퍼의 이름을 입력한다.
8. 앱에서 인증번호를 입력하기 위해 insertIdentifyNumber를 이용한다.
9. 키퍼의 이름을 서버로 전송한다.
10. 입력된 식별번호를 서버로 전송한다.
11. 서버에서 키퍼의 이름을 전송받는다.
12. 서버에서 인증번호를 전송받는다.
13. 인증번호가 저장된 인증번호와 일치하는지 checkIdentifyNumber로 확인한다.
14. 입력받은 인증번호에 해당하는 실버 정보가 있는지 checkIdentifyNumber로 확인한다.
15. 실버에게 키퍼가 보낸 이름과 인증번호와 함께 연결 요청을 전달한다.
16. receiveConnectRequest로 실버앱이 서버로부터 키퍼의 연결요청을 전달받는다.
17. 키퍼의 식별번호를 생성한다.
18. selectSilverID 메소드로 연결되는 실버의 식별번호를 검색한다.
19. insertKeeperID 메소드로 키퍼의 식별번호를 DB에 저장하다.
20. checkMatching로 서버에게 키퍼의 앱과 실버의 앱의 연결이 완료되었다고 알린다..
21. 초기인증이 완료됨을 checkRegist 메소드를 이용해 서버로 전달한다.
22. 서버에서 키퍼의 앱과 실버의 앱의 연결 완료를 확인한다.
23. 앱에서 푸쉬알림을 위한 토큰을 서버로 보낸다.
24. 앱에서 보내진 토큰을 서버에서 받는다.
25. 수신된 토큰을 DB에 저장한다.
26. **샤오미 미밴드의 이상 상태 확인**



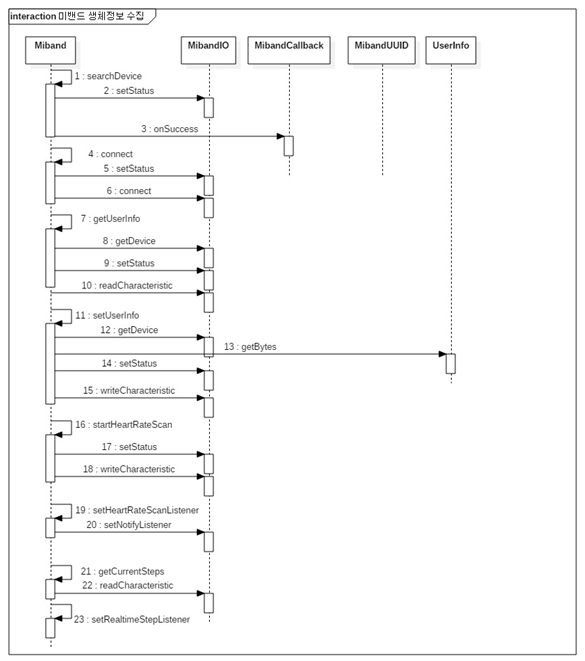
1. 샤오미 미밴드에서 블루투스의 연결 상태를 감지, 실버의 앱으로 상태를 보낸다.
2. 실버의 앱에서 SilverConnection으로 블루투스 연결 상태를 보낸다.
3. 서버에서 샤오미 미밴드의 블루투스 연결 상태를 받아들인다.
4. 서버에서는 새로 받아들인 블루투스 연결 상태로 갱신한다.
5. 서버의 SilverDAO에서 블루투스의 연결 상태를 검색한다.
6. **실버의 정보확인**



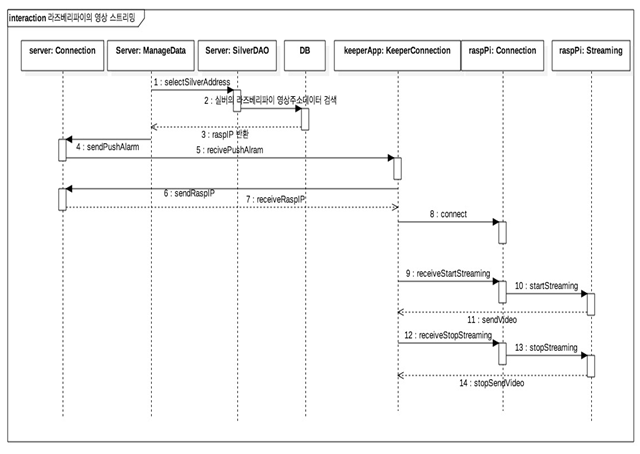
1. 실버의 앱에서 서버로 실버의 데이터를 요청한다.
2. 실버의 앱에서 서버의 Connection 클래스로 앱과 서버를 연결한다. 필요할 때에는 식별번호로 실버인지 키퍼인지 구분할 수 있다.
3. 서버에서 실버의 앱으로 서버와 앱을 연결한다.
4. 실버의 앱과 서버의 연결이 완료되었는지 확인하기 위해 checkMatching을 호출한다.
5. DB에서 실버의 식별번호를 확인한다.
6. 서버에서 실버의 데이터를 검색하기 위해 selectSilverData 메소드를 호출한다.
7. 서버에서 실버의 심박수를 검색한다.
8. 서버에서 블루투스의 연결상태를 검색한다.
9. 서버에서 실버의 귀가여부 상태를 검색한다.
10. 현재 심박수의 위험상태가 얼마나 되는지를 검색한다.
11. 실버의 걸음수를 서버에서 검색한다.
12. DB에서 서버로 실버의 심박수를 받아온다.
13. DB에서 서버로 미밴드의 장착여부를 받아온다.
14. DB에서 서버로 실버의 걸음수를 받아온다.
15. 실버의 귀가여부를 서버가 DB에서 받아온다.
16. 샤오미 미밴드의 블루투스 연결 상태를 DB에서 서버로 받아온다.
17. DB에서 실버의 심박수의 위험레벨이 어떻게 되는지 받아온다.
18. silverVO 클래스에서 silverDAO 클래스로 실버의 데이터를 전달한다.
19. silverDAO클래스에서 서버의 connection 클래스로 실버의 데이터를 보낸다.
20. 실버의 앱으로 실버의 데이터를 보내기 위해 receiveSilverData 메소드를 호출한다.
21. **키퍼의 정보확인**



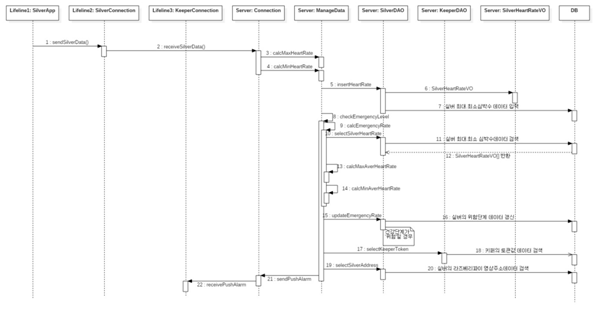
1. 키퍼의 앱에서 서버로 실버의 데이터를 요청한다.
2. 키퍼의 앱에서 서버의 Connection 클래스로 앱과 서버를 연결한다. 필요할 때에는 식별번호로 실버인지 키퍼인지 구분할 수 있다.
3. 서버에서 키퍼의 앱으로 서버와 앱을 연결한다.
4. 키퍼의 앱과 서버의 연결이 완료되었는지 확인하기 위해 checkMatching을 호출한다.
5. DB에서 키퍼의 식별번호를 확인한다.
6. 키퍼의 식별번호와 연결되어있는 실버의 식별번호를 KeeperDAO클래스에서 검색한다.
7. 서버에서 실버의 데이터를 검색하기 위해 selectSilverData 메소드를 호출한다.
8. 서버에서 실버의 심박수를 검색한다.
9. 서버에서 블루투스의 연결상태를 검색한다.
10. 서버에서 실버의 귀가여부 상태를 검색한다.
11. 현재 심박수의 위험상태가 얼마나 되는지를 검색한다.
12. 실버의 걸음수를 서버에서 검색한다.
13. DB에서 서버로 실버의 심박수를 받아온다.
14. DB에서 서버로 미밴드의 장착여부를 받아온다.
15. DB에서 서버로 실버의 걸음수를 받아온다.
16. 실버의 귀가여부를 서버가 DB에서 받아온다.
17. 샤오미 미밴드의 블루투스 연결 상태를 DB에서 서버로 받아온다.
18. DB에서 실버의 심박수의 위험레벨이 어떻게 되는지 받아온다.
19. silverVO 클래스에서 silverDAO 클래스로 실버의 데이터를 전달한다.
20. silverDAO클래스에서 서버의 connection 클래스로 실버의 데이터를 보낸다.
21. 키퍼의 앱으로 실버의 데이터를 보내기 위해 receiveSilverData 메소드를 호출한다.
22. **미밴드 정보수집**



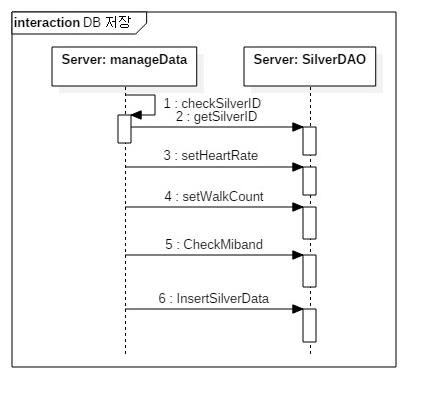
1. 앱이 작동하면, Mi Fit 앱에서 먼저 연결된 디바이스가 있는지를 검색하기 위해 searchDevice 메소드를 호출한다.
2. 미밴드를 ‘STATUS\_SEARCH\_DEVICE’로 설정하기 위해 setStatus를 호출한다.
3. 검색이 성공하였을 경우 MibandCallback 인터페이스의 onSuccess 메소드를 수행한다.
4. 디바이스 검색이 성공하여 디바이스와의 연결을 위해 connect 메소드를 호출한다.
5. 미밴드를 ‘STATUS\_CONNECT’로 설정하기 위해 setStatus를 호출한다.
6. 실제 연결을 위해 MibandIO 클래스의 connect 메소드를 호출한다.
7. 연결된 디바이스를 장착한 유저의 정보를 전달받기 위해 getUserInfo메소드를 호출한다.
8. 현재 연결된 디바이스 호출을 위해 getDevice 메소드를 수행한다.
9. 미밴드를 ‘STATUS\_GET\_USERINFO’로 설정하기 위해 setStatus를 호출한다.
10. 디바이스로부터 정보 수신을 위해 readCharacteristic 메소드를 호출한다.
11. 연결된 디바이스를 장착한 유저의 정보를 갱신하기 위해 setUserInfo메소드를 호출한다.
12. 현재 연결된 디바이스 호출을 위해 getDevice 메소드를 수행한다.
13. 현재 연결된 사용자정보를 얻어오기 위해 getBytes메소드를 호출한다.
14. 미밴드를 ‘STATUS\_SET\_USERINFO’로 설정하기 위해 setStatus를 호출한다.
15. 디바이스로 정보 송신을 위해 writeCharacteristic 메소드를 호출한다.
16. 심박수 측정 시작을 위해 startHeartRateScan 메소드를 수행한다.
17. 미밴드를 ‘STATUS\_START\_HEARTRATE\_SCAN’으로 설정하기 위해 setStatus를 호출한다.
18. 디바이스로 정보 송신을 위해 writeCharacteristic 메소드를 호출한다.
19. 실시간으로 심박수값을 받아오기 위해 setHeartRateScanListener 메소드를 수행한다.
20. 심박수 알림을 위해 setNotifyListener 메소드를 호출하여, 심박수를 알리는 리스너로 오버라이드를 수행한다.
21. 현재 걸음수를 받아오기 위해 getCurrentSteps 메소드를 호출한다.
22. 디바이스로부터 걸음수 정보 수신을 위해 readCharacteristic 메소드를 호출한다.
23. 실시간으로 걸음수값을 받아오기 위해 setRealtimeStepListener 메소드를 수행한다.
24. 걸음수 알림을 위해 setNotifyListener 메소드를 호출하여, 걸음수를 알리는 리스너로 오버라이드를 수행한다.
25. **라즈베리파이의 영상스트리밍**



1. 문제가 발생했을 경우 silverDAO를 통해 silver의 Address를 찾는다.
2. DB에 저장되어 있는 실버의 라즈베리파이의 영상주소 데이터를 검색한다.
3. 실버의 라즈베피아이의 영상주소 데이터를 반환 받는다.
4. Server는 문제가 발생한 silver의 keeper에게 pushAlarm을 보낸다.
5. Keeper는 Server로 부터 pushAlarm을 받는다.
6. Server는 Keeper에게 silver의 집에 설치된 라즈베리파이의 서버 IP 주소를 보낸다.
7. Keeper는 Server로부터 라즈베리파이의 IP를 받는다.
8. Keeper는 Server로부터 받은 라즈베리파이의 IP를 가지고 영상 서버에 접속한다.
9. Keeper는 라즈베리파이에게 영상 스트리밍을 요청한다.
10. 라즈베리파이는 카메라를 통해 영상 스트리밍을 시작한다.
11. Keeper는 카메라의 영상 스트리밍을 라즈베리파이를 통해 전송 받는다.
12. Keeper는 라즈베리파이에게 영상 스트리밍 중지를 요청한다.
13. 라즈베리파이는 카메라를 통해 영상 스트리밍을 중지한다.
14. 카메라의 영상 스트리밍을 라즈베리파이를 통해 Keeper에게 전송하는 것을 중단한다.
15. **문제 발생 처리**



1. SilverApp은 서버로 데이터를 전송하기 위해 sendSilverData 메소드를 호출.
2. sendSilverData는 Server의 Connection으로 데이터를 전송. Server의 Connection은 전달받은 데이터를 DB로 저장하는 메소드를 호출한다.
3. 심박수의 최대값을 계산하기 위해 calcMaxHeartRate 메소드를 호출한다.
4. 심박수의 최솟값을 계산하기 위해 calcMinHeartRate 메소드를 호출한다.
5. 계산된 최대심박수와 최소심박수를 저장하기 위해, insertHeartRate 메소드를 호출한다.
6. VO 클래스 형태로 데이터를 입력하기 위해 클래스 생성자를 실행한다.
7. 심박수 데이터를 DB에 저장한다.
8. 심박수의 위험단계를 계산하기 위해 checkEmergencyLevel 메소드를 호출한다.
9. 위험치를 도출하기 위해 checkEmergencyRate 메소드를 호출한다.
10. 계산하기 위한 데이터를 가져오기 위해, selectSilverHeartRate 메소드를 호출한다.
11. DB로부터 저장된 실버의 최대 심박수 데이터들을 가져온다.
12. DB로부터 저장된 실버의 최소 심박수 데이터들을 가져온다.
13. 최고심박수의 평균 수치 계산을 위해 calcMaxAvgHeartRate 메소드를 호출한다.
14. 최저심박수의 평균 수치 계산을 위해 calcMinAvgHeartRate 메소드를 호출한다. 계산된 수치들을 이용해 건강상태를 판단한다.
15. 판단된 건강상태를 저장하기 위해, updateEmergencyRate 메소드를 호출한다.
16. 실버의 위험단계를 저장한다.
17. checkEmergencyLevel 메소드에 의해 위험 상태로 판단될 시, 위험 상태를 알리기 위해 우선 키퍼의 토큰값을 가져오는 selectKeeperToken메소드를 호출한다.
18. DB로부터 키퍼의 토큰값을 가져온다.
19. 라즈베리파이의 연결주소를 가져오기 위해 selectSilverAddress 메소드를 호출한다.
20. DB로부터 라즈베리파이의 연결주소값을 가져온다.
21. 받아온 키퍼 토큰값을 통해, 키퍼의 앱으로 푸시알림을 보내는 sendPushAlarm 메소드를 호출한다.
22. 푸시알림이 전달되고, receivePushAlarm 메소드가 호출된다.
23. **DB 데이터 저장**



1. Server의 manageData는 SilverID를 확인한다.
2. Server의 manageData는 Server의 SilverDAO로부터 SilverID를 가져온다.
3. Server의 manageData는 Server로부터 실버의 심박수 데이터를 가져온다.
4. Server의 manageData는 Server로부터 실버의 걸음수 데이터를 가져온다.
5. Server의 manageData는 Server로부터 미밴드 탈착여부 데이터를 가져온다.
6. Server의 manageData는 Server의 SilverDAO를 통해 데이터를 DB에 저장한다.

**3. 테스트 명세서 (2페이지 내외)**

|  |  |
| --- | --- |
| 테스트 대상 | 보호대상자의 앱과 보호자의 앱, 데이터베이스 |
| 입력 명세 | 샤오미 미밴드를 통해 수집되는 보호대상자의 심박수, 걸음수가 입력된다.  데이터베이스에 저장되는 보호대상자의 식별번호, 인증번호, 입력된 시간, 미밴드 연결여부, 보호자의 식별번호, 연결된 라즈베리파이의 주소가 입력된다. |
| 출력 명세 | 보호대상자의 심박수의 최대값과 최소값을 구한다. 또한 이 최대값과 최소값을 통해 보호대상자의 현재 위험상태를 파악한다.  위험상태를 파악하고 위험한 상태라 인지하면 보호자의 앱으로 푸쉬알림을 보낸다. 또한 푸쉬알림을 받으면 보호자의 의사에 따라 연결된 카메라를 통해 실시간으로 영상을 확인한다. |
| 테스트 환경 | 보호대상자의 초기인증과정에 기록된 IP주소를 가진 라즈베리파이가 보호대상자의 자택 안에 설치되어있다. 라즈베리파이에는 카메라가 설치되어 있다.  앱, 서버, 영상 서버가 구동되고 있는 상태다.  보호대상자가 착용하고 있는 미밴드는 보호대상자의 앱과 블루투스 연결되어야 한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| silverID | WalkCount | HeartRate | CurrentTime | ConnMiBand |
| SV\_72330cdb0c716b1 | 7032 | 63 | 2017-06-02 15:00 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 82 | 2017-06-02 15:00 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 79 | 2017-06-02 15:00 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 0 | 2017-06-02 15:00 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 52 | 2017-06-02 15:01 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 73 | 2017-06-02 15:01 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 73 | 2017-06-02 15:01 | 1 |
| SV\_72330cdb0c716b1 | 7032 | 58 | 2017-06-02 15:01 | 1 |
| SV\_72330cdb0c716b1 | 7045 | 105 | 2017-06-02 15:01 | 1 |
| SV\_72330cdb0c716b1 | 7045 | 53 | 2017-06-02 15:02 | 1 |
| SV\_72330cdb0c716b1 | 7045 | 61 | 2017-06-02 15:02 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 90 | 2017-06-02 15:02 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 100 | 2017-06-02 15:02 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 68 | 2017-06-02 15:03 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 43 | 2017-06-02 15:03 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 79 | 2017-06-02 15:03 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 78 | 2017-06-02 15:03 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 74 | 2017-06-02 15:03 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 49 | 2017-06-02 15:04 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 71 | 2017-06-02 15:04 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 72 | 2017-06-02 15:04 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 78 | 2017-06-02 15:04 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 86 | 2017-06-02 15:04 | 1 |
| SV\_72330cdb0c716b1 | 7059 | 52 | 2017-06-02 15:05 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| MAXHEARTRATE | MINHEARTRATE | CURRENTTIME | SILVERID |
| 79 | 0 | 2017-06-02 15:28:44:475 | SV\_72330cdb0c716b1 |
| 81 | 0 | 2017-06-02 15:28:52:757 | SV\_72330cdb0c716b1 |
| 87 | 0 | 2017-06-02 15:29:56:883 | SV\_72330cdb0c716b1 |
| 88 | 58 | 2017-06-02 15:30:11:923 | SV\_72330cdb0c716b1 |
| 88 | 57 | 2017-06-02 15:30:24:916 | SV\_72330cdb0c716b1 |
| 88 | 45 | 2017-06-02 15:31:13:562 | SV\_72330cdb0c716b1 |
| 95 | 45 | 2017-06-02 15:31:25:943 | SV\_72330cdb0c716b1 |
| 95 | 50 | 2017-06-02 15:33:29:620 | SV\_72330cdb0c716b1 |
| 96 | 50 | 2017-06-02 15:33:42:517 | SV\_72330cdb0c716b1 |
| 96 | 58 | 2017-06-02 15:34:21:892 | SV\_72330cdb0c716b1 |
| 98 | 58 | 2017-06-02 15:34:50:865 | SV\_72330cdb0c716b1 |
| 98 | 52 | 2017-06-02 15:35:03:298 | SV\_72330cdb0c716b1 |
| 140 | 52 | 2017-06-02 15:35:15:725 | SV\_72330cdb0c716b1 |
| 140 | 0 | 2017-06-02 15:36:04:346 | SV\_72330cdb0c716b1 |
| 111 | 48 | 2017-06-02 15:37:49:400 | SV\_72330cdb0c716b1 |
| 93 | 48 | 2017-06-02 15:38:19:961 | SV\_72330cdb0c716b1 |
| 93 | 60 | 2017-06-02 15:38:46:848 | SV\_72330cdb0c716b1 |
| 93 | 54 | 2017-06-02 15:38:59:272 | SV\_72330cdb0c716b1 |
| 98 | 54 | 2017-06-02 15:39:24:107 | SV\_72330cdb0c716b1 |

**4. 요구사항 대비 시스템 구현 내용**

|  |  |  |
| --- | --- | --- |
| 요구사항 | 구현 결과 |  |
| 스마트폰의 Wifi 스캔과 라즈베리파이의 Wifi 연결여부를 이용하여 귀가여부를 판단할 수 있다. | 시간 관계상 이에 관련된 사항은 전혀 만들지 못했다. | 0% 구현 |
| 3분간 앱이 수집한 심박수 데이터를 서버로 보내고, 서버는 이 수집된 데이터 안에서 최대, 최소 심박수를 찾아 따로 DB에 저장한다. | 앱에 구현되어 있는 새로고침 버튼을 누를 때마다 심박수 데이터가 서버로 보내지고 DB에 저장된다. | 약 80% 구현 |

**5. 개발 추진 내역**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 프로젝트 진행기간(2017.03.17~2017.06.02) | | | | | | | | | | | | | | | |
| 개발내용 | | 추진일정 | | | | | | | | | | | | 역할분담 | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 제안서 작성 및 발표 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영, 차민광, 김현진, 최유니 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | 클래스 구조, UI구상 및 작성 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영, 차민광, 김현진, 최유니 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | 애플리케이션-서버간  소켓 통신 구축 |  |  |  |  |  |  |  |  |  |  |  |  | 김현진 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | 서버 DB (sql 쿼리) 구축 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | 애플리케이션의 UI 구현 |  |  |  |  |  |  |  |  |  |  |  |  | 차민광 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | 앱과 미밴드 블루투스 연결과 데이터 수집, 전송 |  |  |  |  |  |  |  |  |  |  |  |  | 김현진 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | 서버에서 생체정보 받아와 앱에서 띄우기 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영, 김현진 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | 서버에서 인증번호, 식별번호 생성과 전달 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | 앱으로부터 생체정보 서버로 받아와 문제 발생 감지와 처리 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | 앱과 라즈베리파이를 이용한 귀가여부 확인 |  |  |  |  |  |  |  |  |  |  |  |  | 구현 안함 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | 서버로부터 푸쉬알람 받아 앱에서 알림창 띄우고 입력내용 서버로 전송 |  |  |  |  |  |  |  |  |  |  |  |  | 김현진 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | 앱 초기 인증과정(실버, 키퍼) |  |  |  |  |  |  |  |  |  |  |  |  | 백태영 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 13 | 앱에서 스트리밍 요청을 받아 라즈베리파이의 영상을 앱으로 전송 |  |  |  |  |  |  |  |  |  |  |  |  | 김현진 | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | 기능 통합 후 테스트 및 디버깅 |  |  |  |  |  |  |  |  |  |  |  |  | 백태영, 차민광, 김현진, 최유니 | |
|  |  |  |  |  |  |  |  |  |  |  |  |

**6. 개발 프로그램 (5페이지 이내)**

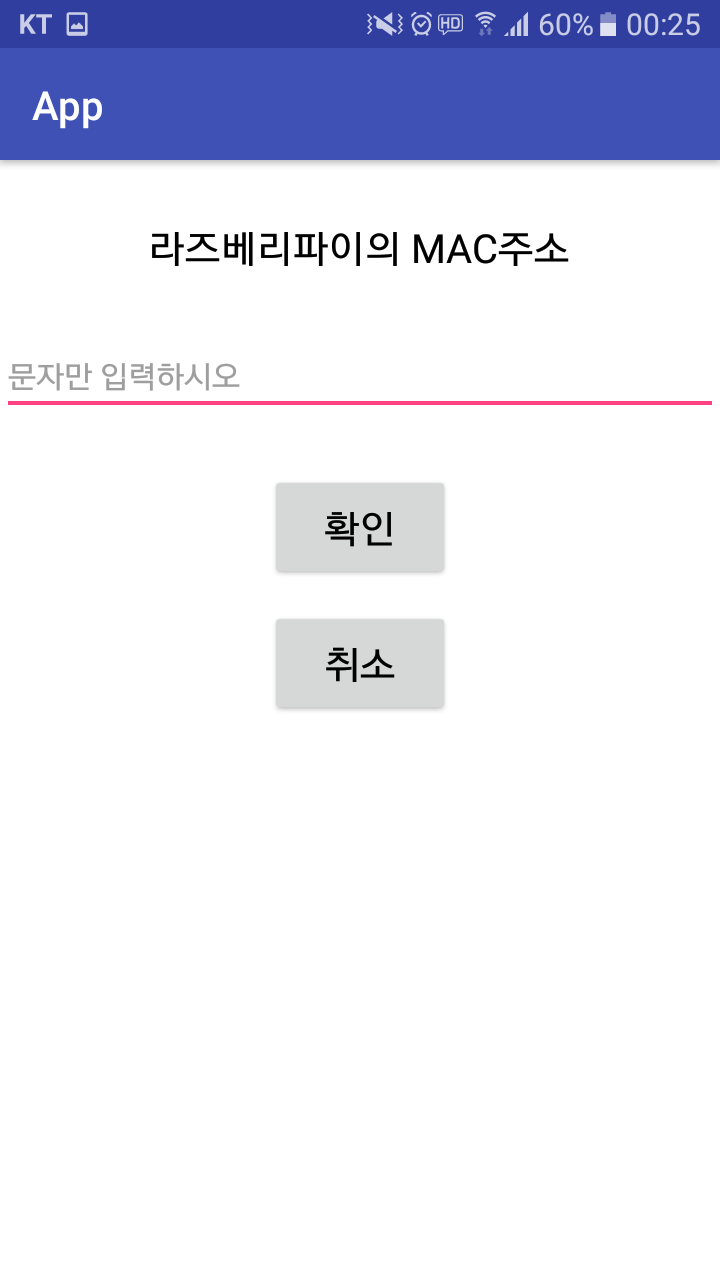
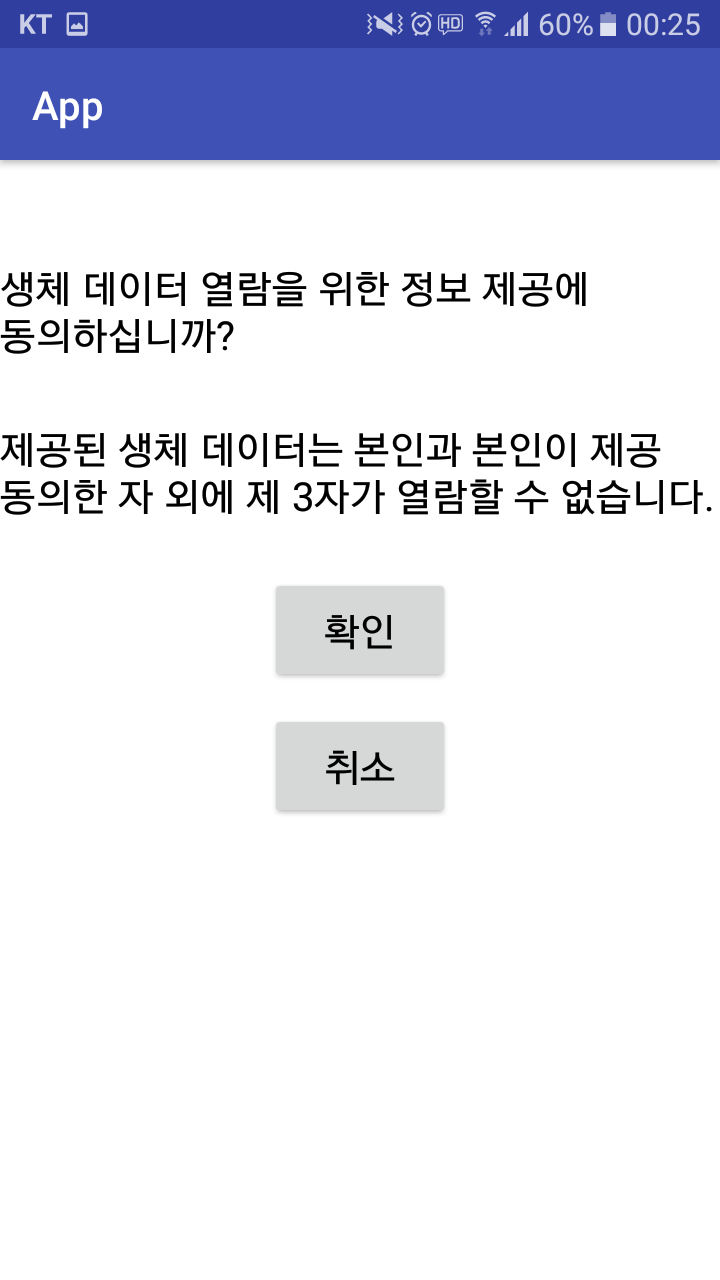
1. 앱

1. 앱을 처음으로 실행시키게 되면 역할을 선택하라는 페이지가 화면에 뜨게 된다.



2. 선택한 역할에 따라 초기 인증과정을 거친다.

1. 보호대상자(실버)



1. 보호자(키퍼)

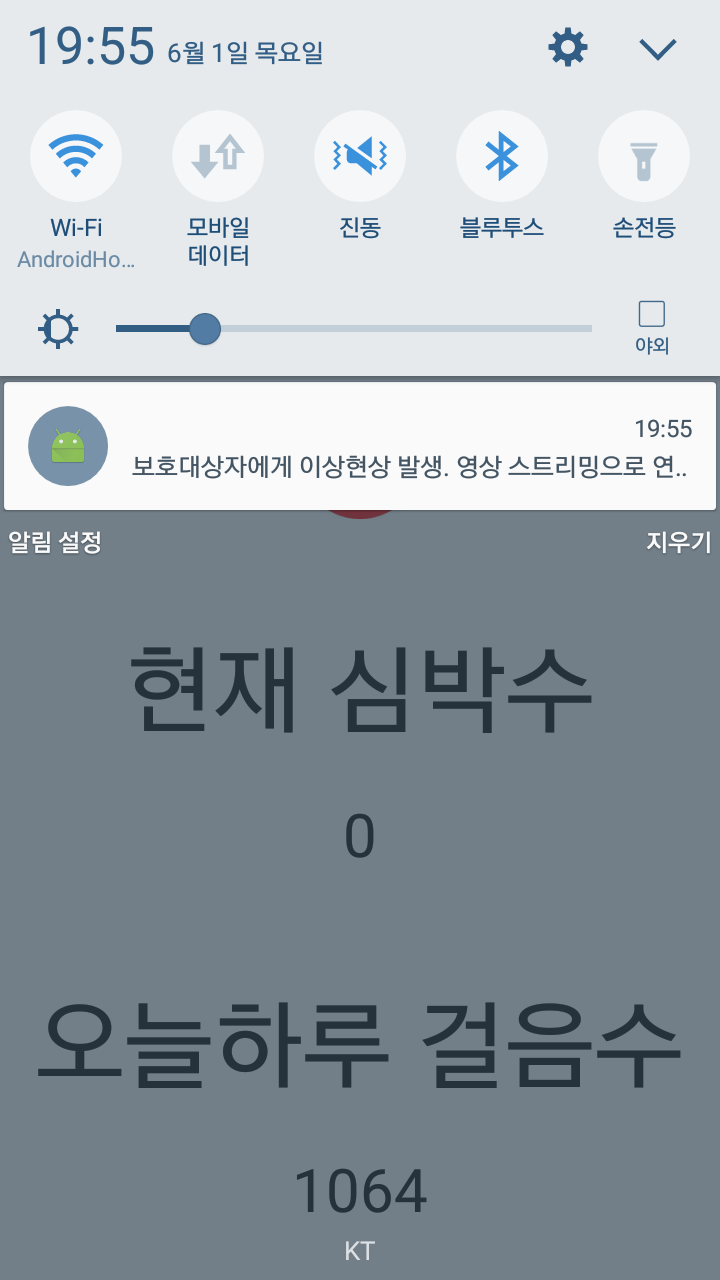


3. 실버의 정보 확인화면으로 실버의 심박수와 걸음수를 확인한다.

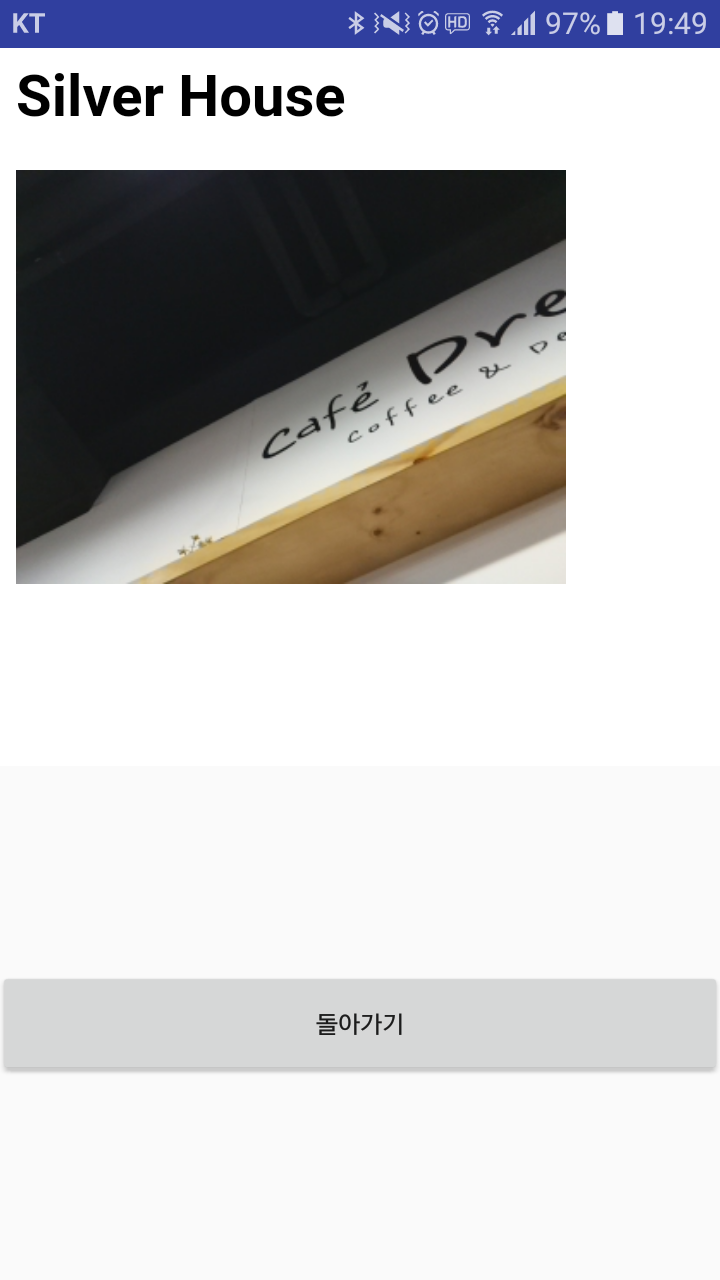


위에 있는 붉은 원은 위급상황을 나타낸다. 안전한 경우 녹색으로, 경고일 경우 노란색, 위험할 경우 붉은색으로 표시된다. 위의 상황은 심박수 측정이 제대로 되지 않아 0이 입력된 것으로, 위급상황이기 때문에 붉은색으로 표시되어있다.

4. 위급상황이 일어났을 때, 키퍼의 핸드폰으로 팝업창이 뜬다.



5. 팝업창을 누르면 키퍼의 앱에 촬영 영상이 떠오른다.



하단에 있는 [돌아가기] 버튼을 누르면 이전 화면으로 돌아간다. 영상은 실버의 자택에 설치된 카메라에 촬영되는 영상을 실시간으로 스트리밍해 받아온다.

**7. 설계 구성 요소**

|  |  |  |
| --- | --- | --- |
| **설계**  **구성**  **요소** | **목표**  **설정** | 보호자가 멀리 떨어져있거나 자주 전화를 하지 못하는 상황에서도 독거노인의 심박수를 앱을 통해 간단히 알 수 있도록 한다. 또한 독거노인에게 위급상황이 발생했을 때는 독거노인의 자택 내부를 살필 수 있도록 한다. |
| **합성** | 네트워크와 웹프로그래밍, 데이터베이스,자바 수업에서 배운 지식을 이용해 서버와 DB 구현, 따로 공부를 해 앱과 라즈베리파이의 영상 서버 구현. |
| **분석** | 개발자의 요구사항으로부터 이를 분석하여 요구사항 명세서를 작성한다. 그리고 전체 시스템을 분석해 방안을 모색한다. |
| **제작** | 유스케이스 다이어그램, 클래스 다이어그램, 시퀀스 다이어그램을 도출해 전체적인 시스템과 순서를 파악한 후 시스템을 구현한다. |
| **시험** | 구현한 시스템을 지속적으로 테스트. 계속적인 테스트로 정상작동 여부를 확인한다. |
| **평가** | 평가는 프로그램의 시연을 통해 이루어진다. |
| **제한**  **조건** | **산업**  **표준** |  |
| **경제성** | 웨어러블 디바이스인 미밴드와 라즈베리파이, 라즈베리파이 전용 카메라만 있으면 되기에 경제적으로 저렴하다. |
| **안정성** | 어떤 사용자가 미밴드를 착용하든지 보호대상자의 스마트폰에만 연결되어 있다면 측정이 가능하다. 그러나 미밴드가 제대로 인식이 되지 않거나 하는 경우 심박수가 측정이 되지 않는 경우가 있다. |
| **미학** | 눈에 잘 띄게 User Interface를 만들어놓았다. 인증과정에서 글뿐만 아니라 그에 맞는 이미지를 띄웠기 때문에 시력이 좋지 않은 보호대상자도 쉽게 파악할 수 있다. 또한 데이터 화면 역시 글씨가 크고 색깔이 선명해 눈에 잘 들어온다. |
| **사회**  **영향** | 보호대상자의 생체정보를 공개한다는 점에서 민감하기 때문에 사전에 보호대상자의 동의를 구한다. 고독사를 방지시켜 고독사의 비율이 줄어들 것이라 기대된다. |

**8. 향후 개선 계획**

우선 시간이 부족해서 채 구현하지 못한 귀가확인 여부에 대한 기능을 구현할 필요가 있다. 또한 미밴드에서 받아오는 데이터가 미밴드의 디스플레이에 떠오르는 데이터를 가져오는 것이 아니라 미밴드에서 직접적으로 받아오도록 개선해야 한다. 위험상황 알림을 좀 더 제대로 하기 위해 초기값을 넣어줘야 할 필요가 있다. 그리고, 구현된 앱을 제대로 실행하기 위해서는 미밴드의 공식 앱인 Mi Fit를 사용할 필요가 있다. 이 Mi Fit를 사용하지 않아도 심박수를 수집할 수 있도록 만들어야 할 필요가 있다. 무엇보다도 딱히 새로고침 버튼을 누르지 않더라도 일정 시간이 지나면 심박수를 수집해 자동으로 바꿔주는 기능을 추가할 것이다.

**9. 개발과정에서의 문제점**

미밴드의 데이터 수집과 관련한 개발 단계에서 난항을 겪었다. 미밴드가 보호대상자의 앱과 블루투스 연결이 되어있어야 데이터 수집이 가능한데 미밴드1은 정보가 많은 반면에 미밴드2는 상대적으로 최근에 출시되어 정보가 많지 않았다. 미밴드의 블루투스 UUID를 찾아내는 과정이 어려웠다. 블루투스 연결 후에도 미밴드에 저장되어 있는 데이터를 읽어오는 데에는 성공하지 못하였고, 미밴드 자체의 데이터가 수집되는 순간에만 앱에서 데이터를 읽어올 수 있었다. 앱에서는 [미밴드로부터 데이터 수집 - 서버로 데이터 전송 - 서버에서 데이터 수신]의 과정을 거쳐 화면에 데이터를 띄우게 되는데 이 과정을 각각의 쓰레드로 구현하여 순차적인 실행과 제어에 문제가 발생하기도 했다. 미밴드를 이용하여 생체 정보를 수집하기 때문에 기계 자체가 갖는 제대로된 데이터 수집에 대한 한계가 있다. 미밴드 착용자가 움직이고 있을 경우 정상적인 데이터 수집이 안될 수 있으며, 미밴드를 착용하지 않아 심박수가 0이 된 경우와 정보수집이 제대로 되지 않아 심박수가 0이 된 경우를 구분하기 어렵다.