운동선수들을 위한 훈련 일지 서비스

정민혁, 정지현

경희대학교 컴퓨터공학과

jeongmh09@khu.ac.kr, jihyoun0602@khu.ac.kr

Training diary services for athletes

Minhyeok Jeong, Jihyoun Jung

Department of Computer Science and Engineering, KyungHee University

요 약

운동 선수들에게 자신의 몸 상태를 관리하는 것은 꾸준한 훈련과 연습만큼 중요하다. 자신의 컨디션이나 부상 정도에 충분한 주의를 기울이지 않다가 심한 부상으로 이어지면서 선수 생활을 마감하는 경우도 흔하게 찾아볼 수 있다. 따라서 본 문서에서는 선수들의 훈련 내용, 몸 상태, 부상 정도 등을 통합적으로 기록하고 조회하고, 나아가 분석을 통한 조언을 받을 수 있는 기능을 제공하는 어플리케이션을 설계하고 구현한다.

1. 서론

1.1 연구 배경

1.1.1. 선수의 능동적 트레이닝/컨디션 관리 부재

선수들은 높은 강도의 훈련을 실시하지만, 강도 높은 훈련량에 따른 효과적인 트레이닝 및 컨디션 관리는 부재하다. 운동선수는 스스로 트레이닝 내용을 편리하게 관리할 수 있어야 하며, 실력향상을 위한 지속적 컨디션 관리는 필수이다.

운동은 신체적 역량과 기술의 대결이기 때문에, 훈련을 몸으로 익히는 것만이 전부라고 생각하지만 사실은 그렇지 않다. 코치/감독의 트레이닝과 전술에 대한 인지가 선수 본인에게 적절히 이루어지고, 이를 스스로 생각하며 훈련할 때 비로소 코칭과 전술은 '선수'의 것이 된다.

하지만 아마추어 선수들의 경우, 인지적 트레이닝을 전혀 해 본 경험이 없기 때문에 이를 어떻게 기록하고 관리해야 하는지 알지 못한다.

그러나 수기로 작성하는 훈련일지는 작성 시 긴 시간이 소요되고, 체계 없이 작성된 신체 및 심리 데이터는 직접적으로 훈련에 반영되기 어렵다.

1.1.2. 데이터 기반의 훈련관리

운동선수의 부상은 누적된 피로와 과부하로 인해 발생하는 것이 대부분이다.
한국스포츠개발원의 연구에 따르면, 훈련 중 발생하는 부상은 85.4% 이며, 가장 큰 원인은 '오버트레이닝' 이다.

그러므로 지도자는 선수의 운동 능력을 정확히 파악하고, 선수 개개인의 신체 컨디션에 맞는 훈련을 진행해야 한다.

하지만 선수단 대비 코치진의 인원은 현저히 적기 때문에 실질적으로 지속적인 선수단 관리, 선수 운동 및 심리 데이터 관리는 불가능하다. 때문에 일괄적으로 높은 수준의 훈련을 지속할 수밖에 없으며, 이는 훈련 효율 극소화를 야기하며 선수의 부상으로 이어진다.

이는 선수의 지속적인 운동 및 심리 데이터 관리를 통해 해결할 수 있다. 선수의 주관적인 느낌은 스포츠 과학 연구에 따른 선수 케어에 굉장히 중요한 지표이다. 때문에 선수들의 트레이닝, 컨디션 데이터를 수집, 파악, 통계 관리하여 데이터를 기반으로 선수 개개인의 상태에 알맞은 훈련을 가이드 하고자 한다.

1.2. 연구 목표

1.2.1. 운동선수들의 컨디셔닝 데이터 관리를 통한 부상방지

본 서비스는 선수들이 감각적으로만 느끼는 부상의 정도에 대해 체계적으로 기록을 남길 수 있게 하며, 이를 기간별/부위별로 분석해 선수 및 코치진이 부상정보에 대해 올바르게 인식할 수 있도록 한다. 이에 대한 정보를 가지고 훈련 및 경기계획을 수립하며 선수들의 부상을 방지한다.

이에 선수들에게 트레이닝 및 컨디션 관리에 최적화된 데이터를 직관적인 UI 를 통해 간편한 방법으로 수집한다.

1.2.2. 인지적 트레이닝 정보 활용을 통한 아마추어 운동선수의 잠재력 및 훈련효과 극대화

트레이닝에 대한 질적인 정보를 받아 선수들이 자신의 훈련에 대한 내용과 코칭데이터를 직접 기입함으로써 성찰적이고 반성적인 사고를 통해 훈련효과를 극대화하는 것을 목표로 한다.

1.2.3. 중장기적 목표설정을 통한 루틴 형성 및 멘탈 트레이닝

선수의 훌륭한 운동수행을 위해서는 심리훈련이 꼭 필요하다. 그리고 이 심리훈련의 기본은 바로 '목표설정' 이다. 먼저 운동선수 생활의 최종 목표와 이 목표를 이루기 위한 세부목표를 설정하고, 이를 달성할 수 있는 나만의 루틴을 설정하도록 한다. 그리고 이 루틴은 매일 훈련일지를 작성하며 달성여부를 체크하게 된다. 이 과정을 통해 매일 목표를 되새기도록 하고, 목표를 쪼개서 수립하고 달성하는 성취감을 통해 소기의 멘탈트레이닝 성과를 거둘 수 있다.

2. 관련연구

2.1 개발환경

[프론트엔드]

2.1.1 JavaScript

JavaScript 는 웹을 위한 인터프리터 언어이자 객체 기반의 스크립트 프로그래밍 언어이다. 웹 브라우저에서 동작하는 프로그래밍 언어이며, 다른 응용 프로그램의 내장 객체에도 접근 가능하다. 컴파일 과정이 필요 없기 때문에 빠른 시간 안에 스크립트 코드를 작성할 수 있다는 장점이 있다. 그 밖에도 웹에 특화된 기술 이기 때문에 운영체제나 플랫폼에 상관없이 작동되고 높은 확장성을 지닌다. 클라이언트에 특화된 기능들, 예컨대 쿠키, 세션, 스토리지 등을 이용할 수 있고 Axios 기술을 통해 서버와의 통신 기능을 수행한다.

본 서비스에서는 ES6 이상의 문법을 사용하며 크로스 브라우징 이슈 대응을 위해 BABEL 을 사용한다.

2.1.2 React Native

React Native 는 페이스북이 개발한 오픈 소스 모바일 애플리케이션 프레임워크이다. React Native 를 통해 Native 플랫폼 API 를 사용할 수 있다. React native 는 특별한 컴포넌트들의 집합으로서 이러한 컴포넌트들은 컴파일을 통해 Native 위젯으로 변환된다. UI 는 컴파일을 통해 native views 로 변환되지만, 로직은 자바스크립트로 실행된다는 특징이 있다. 따라서 자바스크립트 로직을 그대로 사용할 수 있다는 장점이 있다. 그 밖에도 React 를 알면 ios, android 동시 개발이 가능하고 소스코드 수정 시 변경된 내용을 바로 확인할 수 있다는 장점이 있다.

본 서비스는 짧은 시간 안에 ios , android 모두를 개발해야 하고 서비스의 특성상 native 적 성능을 요하지 않는 부분이 있기 때문에 React native 를 이용하여 모바일 앱을 제작한다.

2.1.3 Redux

Redux 란 JavaScript App 을 위한 상태 관리 라이브러리이다. React 로 프로젝트를 진행하게 되면, Component 는 local state 를 가지게 되고, 어플리케이션은 global state 를 가지게 된다. 각

컴포넌트들끼리 props 를 넘기는 등의 소통이 필요로 해지는데 어플리케이션이 복잡해질수록 이러한 통신 또한 복잡해진다. 이 때 Redux 를 사용하게 되면 상태의 중앙화가 가능해진다. 어플리케이션의 상태를 store 를 통해 관리하며 쉽게 저장하고 불러올 수 있다. Redux 는 상태를 변경하는 도중 side effect 가 일어나지 않도록 코딩하기를 권장한다. 이와 함께 상태를 변경하려는 시도를 복제 저장, 전송할 수 있도록 자바스크립트 객체 형태로 구성하기를 강제하기 때문에 상태를 예측 가능하게 한다는 장점이 있다.
특히 본 서비스는 redux 의 코드양에서 오는 피로감을 줄이기 위해 redux-toolkit 인 createSlice 를 사용하고자 한다.

[백엔드]

2.1.4 Node.js

Node.js 는 Chrome V8 JavaScript 엔진으로 빌드 된 JavaScript 런타임 환경이다. 프런트엔드에서 사용되는 JavaScript 를 백엔드에서 사용할 수 있게 해주고, 웹 브라우저 밖에서도 코드를 실행시킬 6 수 있게 해준다. 비동기 I/O 를 지원하기 때문에 대량의 가벼운 I/O 트랜잭션이 발생하는 웹 서버의 경우에도 우수한 처리 속도를 내며, 많은 인기를 끌고 있다. 또한 npm 이라는 패키지 매니저를 통해 수많은 패키지를 활용하여 개발을 훨씬 쉽게 할 수 있다.

2.1.5 Django

Django 는 파이썬 언어로 프로그래밍하는 오픈소스 웹 프레임워크이다. MTV (Model, Template, View) 디자인 패턴에 따라 만들어지는데, Model 은 보통의 경우 데이터베이스의 테이블을, Template 은 사용자에게 실제로 보여지는 문서나 웹 페이지, View 는 HTTP 응답을 통해 어떤 데이터를 클라이언트에게 보여줄지 결정하는 요소이다. Django 는 많은 기능을 정형화된 형태로 제공하기 때문에 빠른 개발속도가 특히 장점으로 부각된다. 또한 자체 제공하는 ORM (Object Relational Mapping) 방식을 사용할 수 있으므로 데이터베이스를 잘 몰라도 데이터베이스 설계가 가능하도록 구현되어 있다.

2.1.6. Flask

Flask 는 파이썬 언어로 프로그래밍하는 BSD 라이선스의 마이크로 웹 프레임워크이다. 핵심 기능을 간단하게 유지하고 확장성 넓은 특징을 제공하기 위해 데이터베이스나 유효성 확인 등의 서드 파티 라이브러리를 포함하지 않기 때문에 마이크로 웹 프레임워크라고 불린다. 대신 수많은 확장 기능을 제공하고 문법이 단순하기 때문에 가벼운 프레임워크로서, 개발자가 원하는 목적에 맞게 커스텀하기도 용이하다.

2.1.7 FastAPI

FastAPI 는 파이썬 언어의 3.6 버전이상의 API 를 빌드하기 위한 파이썬 기반의 웹 프레임워크이다. 파이썬이 인터프리터 언어임에도 불구하고 내부적으로 Starlette 을 사용하기 때문에 Node.js.나 Go 만큼 빠른 처리 속도를 낸다는 특징이 있다. 또한 Flask 와 코드 작성 형식이 유사하기 때문에 기존 Flask 유저는 더욱 쉽게 배울 수 있다. 우수한 성능과 배우기 쉽다는 장점덕에 높은 생산성을 자랑하기 때문에 많은 기업에서 FastAPI 를 채택하고 있다.

FastAPI 의 장점으로 Swagger UI 의 형태로 자동으로 API 문서를 작성해준다. 이로인해 개발에 많은 시간을 아낄 수 있게 된다.

본 서비스에서는 FastAPI 를 사용하여 웹 애플리케이션 서버를 구축한다.
JetBrains 사에서 2020 년에 200 개 이상의 국가에서 28,000 명 이상의 Python 사용자를 대상을 진행한 설문조사를 바탕으로 공개한 자료에 따르면, 웹 개발을 위해 Python 이 사용되는 비율이 50%에 육박한다 (사진 1). 그만큼 Python 또한 웹 개발에 점점 최적화되고 있음을 의미한다. 또한 그림 2 를 통해 FastAPI 가 세번째로 많이 사용되는 웹 프레임워크임을 알 수 있다.
JetBrains 사에서 밝힌 바에 따르면, FastAPI 는 2020 년도 설문조사에 처음 포함되었다. 그럼에도 불구하고 FastAPI 의 선호도가 3 순위에 오를만큼 성능과 편리함이 입증되었음을 나타낸다.

사진 1. 파이썬의 주요 사용 용도

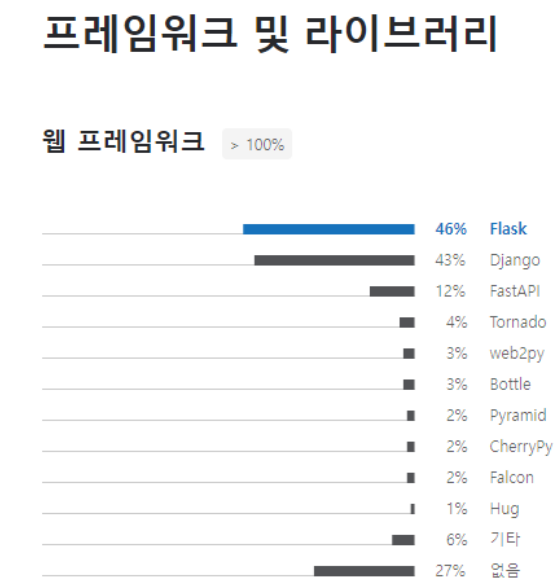
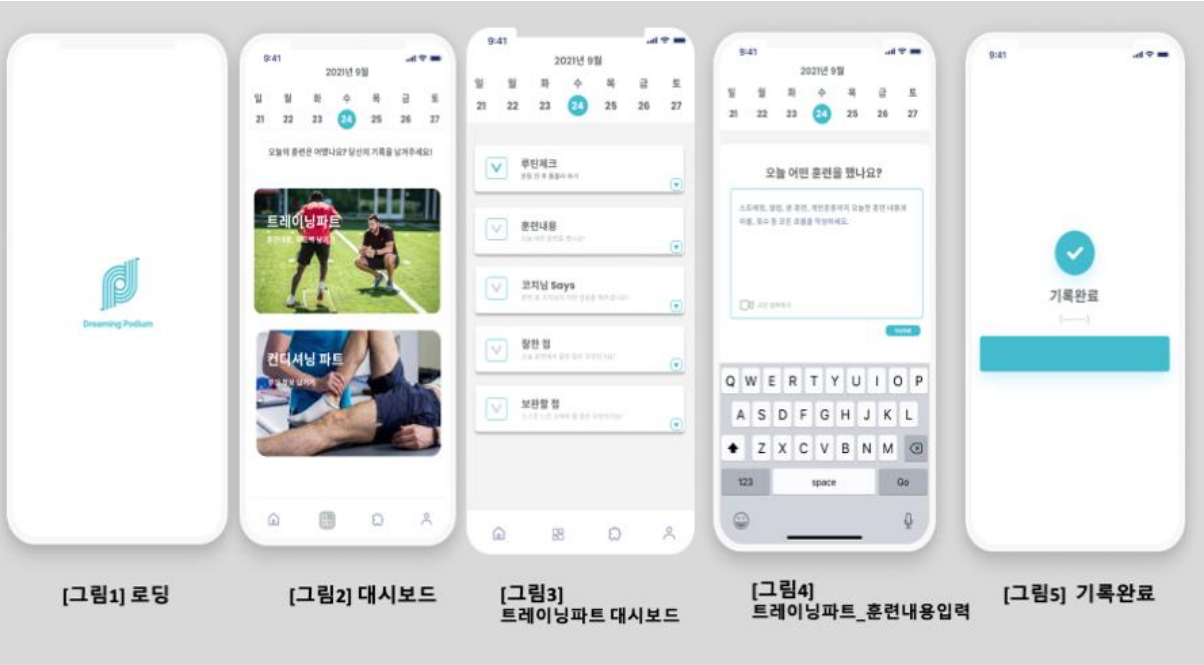


사진 2. 파이썬 웹 프레임워크 선호도

3. 프로젝트 내용

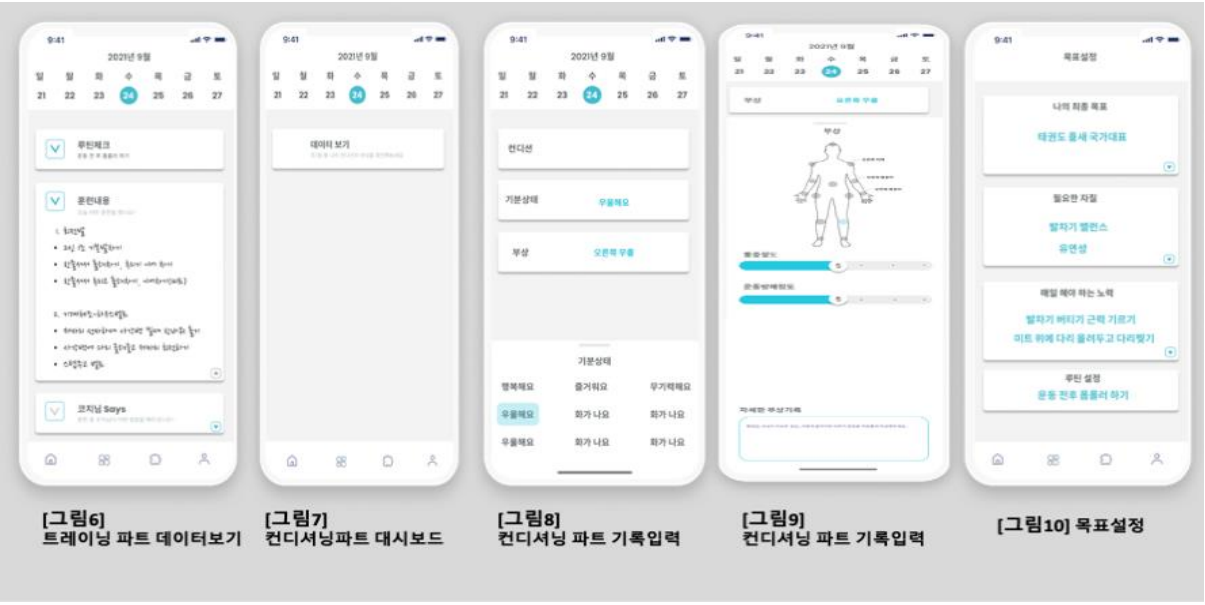
3.1 시나리오

아래는 구현된 UI 의 모습이다.



[그림1] 로딩 [그림2] 대시보드 [그림3] 트레이닝파트 대시보드 [그림4] 트레이닝파트_훈련내용입력 [그림5] 기록완료

[그림 2]는 어플리케이션의 메인 화면이다. 메인 에서는 날짜를 선택할 수 있는 상단바와 트레이닝파트, 컨디셔닝파트를 출력한다. 상단바의 날짜를 선택하면 해당 날짜에 작성한 훈련 일지가 보인다. 트레이닝파트는 양적인 데이터를 입력하는 메뉴로서, 클릭 시 트레이닝파트의 대시보드를 출력한다. **[그림 3]**은 트레이닝파트의 대시보드이다. 하위 메뉴로는 루틴체크, 훈련내용, 코치님 says, 잘한 점, 보완할 점 이 있다. 각 메뉴는 같은 컴포넌트로 구성되어 동일한 UI 로 출력된다. 하위 메뉴를 클릭할 경우 **[그림 4]** 내용 입력 창이 출력 되며 사용자가 입력을 마친 뒤 제출 버튼을 클 **[그림 5]** 기록 완료 창이 출력된다.



[그림6] 트레이닝 파트 데이터보기 [그림7] 컨디셔닝파트 대시보드 [그림8] 컨디셔닝 파트 기록입력 [그림9] 컨디셔닝 파트 기록입력 [그림10] 목표설정

[그림 6] 은 트레이닝파트의 대시보드에서 하위 메뉴를 펼쳤을 시의 UI 이다. 각 메뉴 탭은 클릭 시 아래로 슬라이드 애니메이션 효과를 내며 기록한 내용이 보인다. **[그림 7]** 은 컨디셔닝파트의 대시보드이다. 데이터 보기를 클릭하면 **[그림 8]** 내용 입력 창이 출력된다. 해당 창은 컨디션, 기분상태, 부상 등의 양적 정보를 입력할 수 있다. 컨디션, 기분 상태 선택 UI 는 메뉴 탭으로 구성된다. **[그림 9]는** 부상 정보에 해당하는 입력창이다. 신체 부위를 명시하고 터치 영역을 제공하여 신체 부위를 이미지와 매핑 시킨다. 부상 정도는 슬라이더를 제공하여 입력 받는다. **[그림 10]**은 목표 설정 파트이다. 사용자가 자신의 목표, 자질, 노력, 루틴 등을 설정할 수 있는 입력창이 출력된다.

3.2. 요구사항

3.2.1 UI 요구사항

React Native 기반으로 만들어진 앱이기 때문에 크로스 플랫폼으로 서비스를 제작하고자 한다. 특히 ios/android 의 네이티브 적 속성을 모두 고려하여 UI 를 제작해야 한다. 다양한 기기들에서도 가변적으로 크기를 맞출 수 있도록 해야한다. 본 서비스는 미적인 부분뿐만 아니라 UX 증대를 위해 설계되어야 한다. 홈 화면과 각 탭들의 화면을 제외하고는 유사한 느낌의 UI 로 통일되어 사용자가 매일 글을 쓰는 데에 있어서 익숙함을 줄 수 있도록 한다. 또한 서비스 특성상 사용자가 글을 쓰는 항목이 많이 때문에 피로도를 줄여주는 UI 가 요구되며 직관적인 UI 가 요구된다.

3.2.2 회원가입 및 로그인에 대한 요구 사항

카카오톡과 애플 계정 연동 API 를 활용한 소셜 로그인 기능을 도입한다. 많은 웹사이트에서 회원가입을 요구하게 되면서 새로운 계정 생성에 피로감을 느끼는 사람들이 생기게 되었다. 카카오톡은 남녀노소를 불문하고 사용하고 있으며, 카카오 계정을 연동하여 로그인하는 소셜 로그인 기능은 새로운 계정 생성에 대한 부담을 해소시켜 준다. 또한 회원가입시 요구되는 개인 정보들은 민감한 요소인데, 소셜 로그인을 통하면 이미 어느정도 보안이 검증된 이름있는 기업의 서비스를 거치게 되므로, 직접 회원가입 기능으로 유저 정보를 받고 관리하는 것보다 보안적으로 우수하다. 애플 계정 연동 기능은 ios 앱에서 타 소셜 로그인 기능을 사용하기 위해 필수적으로 포함해야 하는 기능이다. 따라서 애플 계정 연동기능 또한 구현한다.

3.2.3 훈련 내용 기록에 대한 요구 사항

- 훈련 내용을 유형별로 버튼을 통해 기록할 수 있도록 구현한다. 존재하는 유형 이외에 추가로 기록하고 싶은 내용을 따로 기록할 인터페이스를 구현한다.

- 훈련 항목 외에 훈련 중 내가 잘한 점, 못한 점을 기록할 수 있는 인터페이스를 구현한다. 이 항목에는 사진이나 영상을 첨부할 수 있게 구현하여 실제 다이어리를 쓰는 것처럼 훈련에 대한 스스로의 고찰을 적을 수 있게 한다.

- 훈련 중 받은 피드백을 기록할 수 있게 구현한다.

- 나에 대한 응원 메시지를 100 자 이내로 작성할 수 있는 인터페이스를 구현한다.

3.2.4 신체/심리 상태 기록에 대한 요구 사항

- 유형화된 신체 컨디션 정보를 버튼으로 보여주고, 사용자가 해당하는 신체 컨디션 정보를 선택할 수 있게 구현한다.

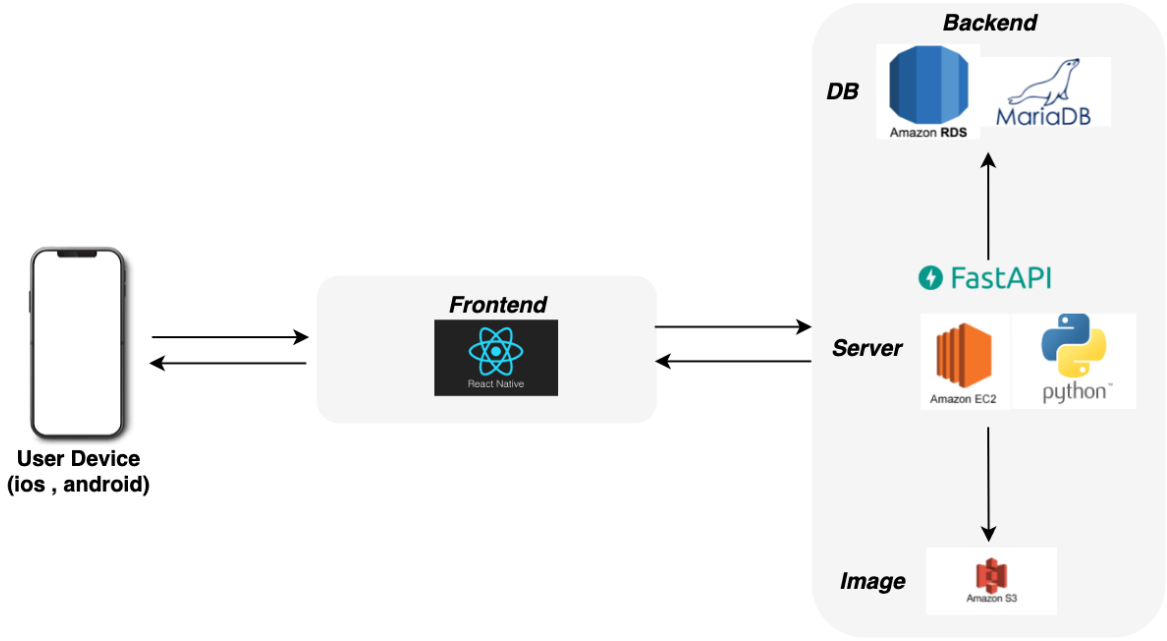
- 유형화된 심리 컨디션 정보를 버튼으로 보여주고, 사용자가 해당하는 심리 컨디션 정보를 선택할 수 있게 구현한다.

- 유형화된 부상 정보들을 버튼으로 보여주고, 사용자가 해당하는 부상 정보들을 선택할 수 있게 구현한다.

- 신체 주요 관절 및 근육을 선택해 사용자가 자신의 부상 부위의 통증 정도와 운동 방해 정도를 입력할 수 있게 구현한다. 복수 선택 가능하게 구현한다.

3.3. 시스템 설계

3.3.1. 시스템 구성도

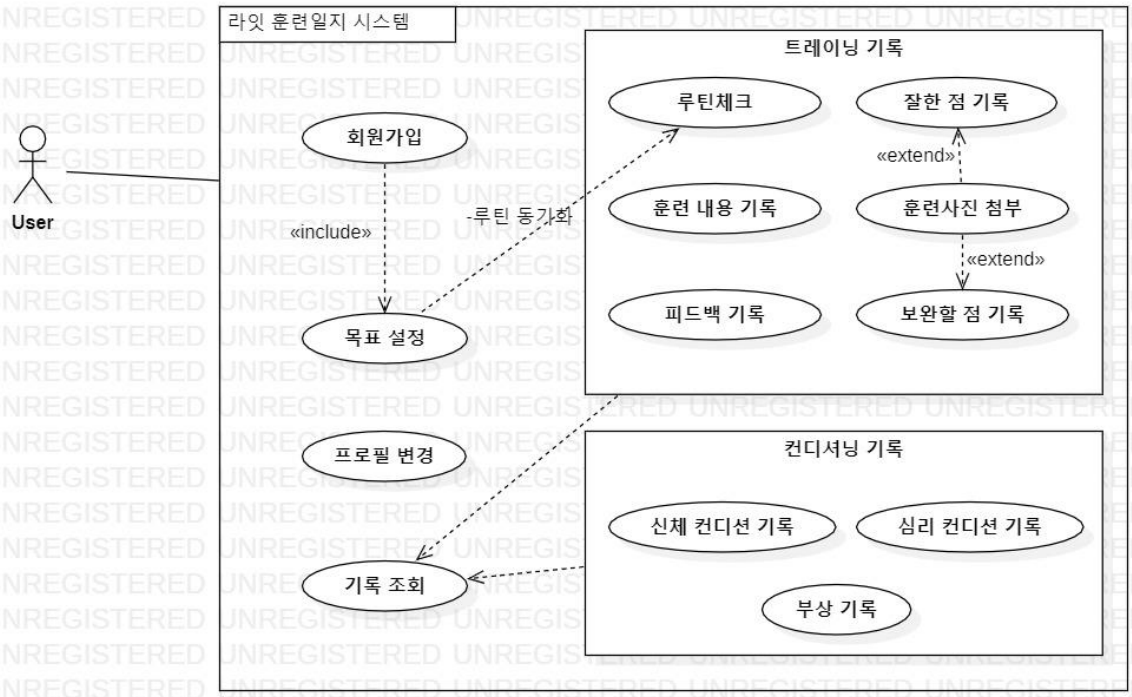


[그림 x] 시스템 구성도

시스템 구성도는 그림 X 와 같다. Frontend와 backend가 분리된 클라이언트-서버 아키텍처 디자인 패턴이다. Frontend에서 데이터를 Backend 서버에 요청하면 서버에서 요청한 데이터를 전달해 주어 frontend에서 데이터에 직접 접근하지 않아 데이터 자체와 데이터 경로를 안전하게 보관하고 제어할 수 있다. Frontend는 ----. Backend는 AWS EC2 서버 위에서 Python 언어의 FastAPI 웹 프레임워크를 사용하여 만들어진 프로그램이 프런트 엔드에 데이터 접근에 대한 API를 제공한다. 생성된 각종 데이터는 AWS RDS 서비스를 사용하여 구축된 MariaDB에 저장이 되고, 이미지 파일들은 AWS S3 버킷에 업로드 되어 저장된다.

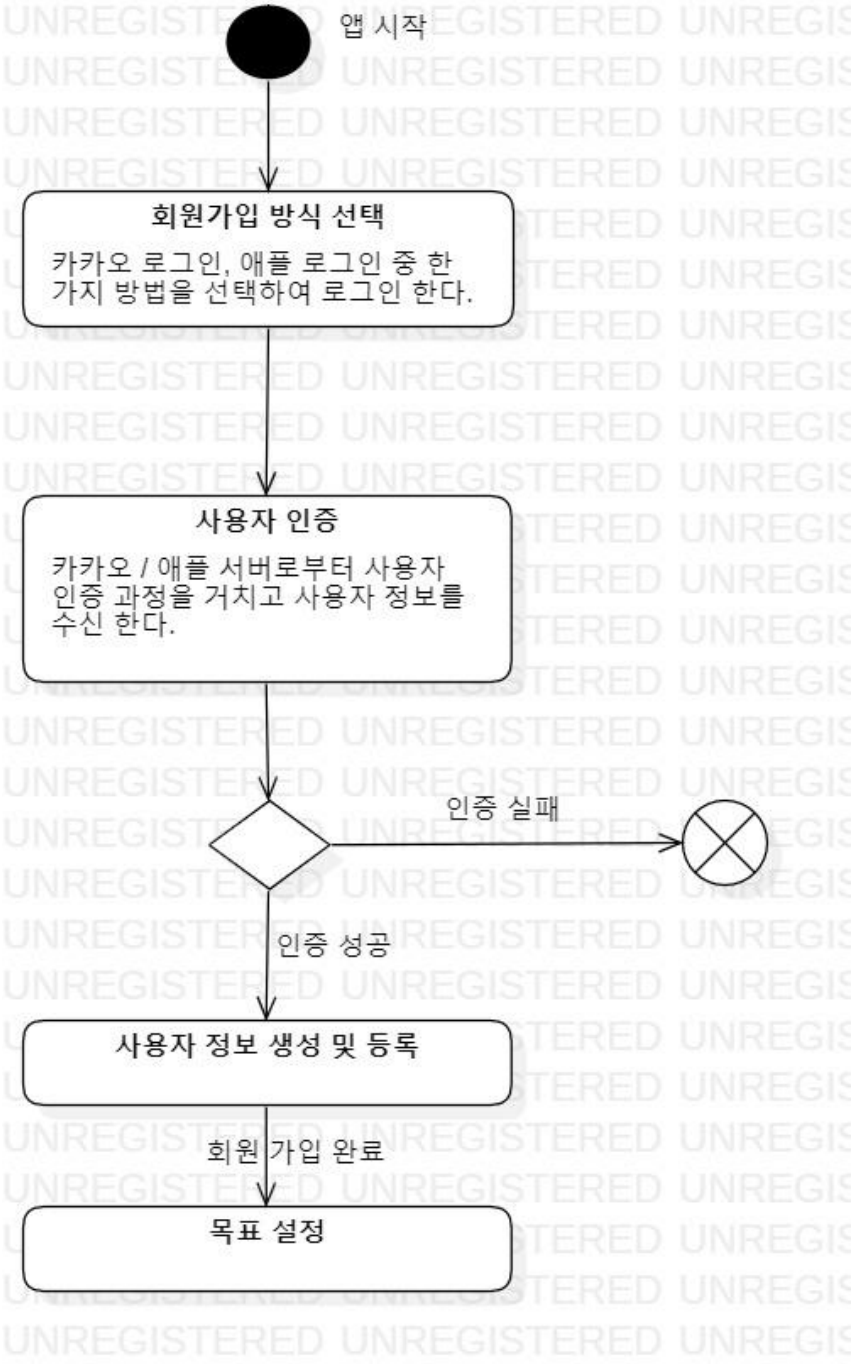
3.3.2 UML System Diagram

3.3.2.1 Use Case Diagram

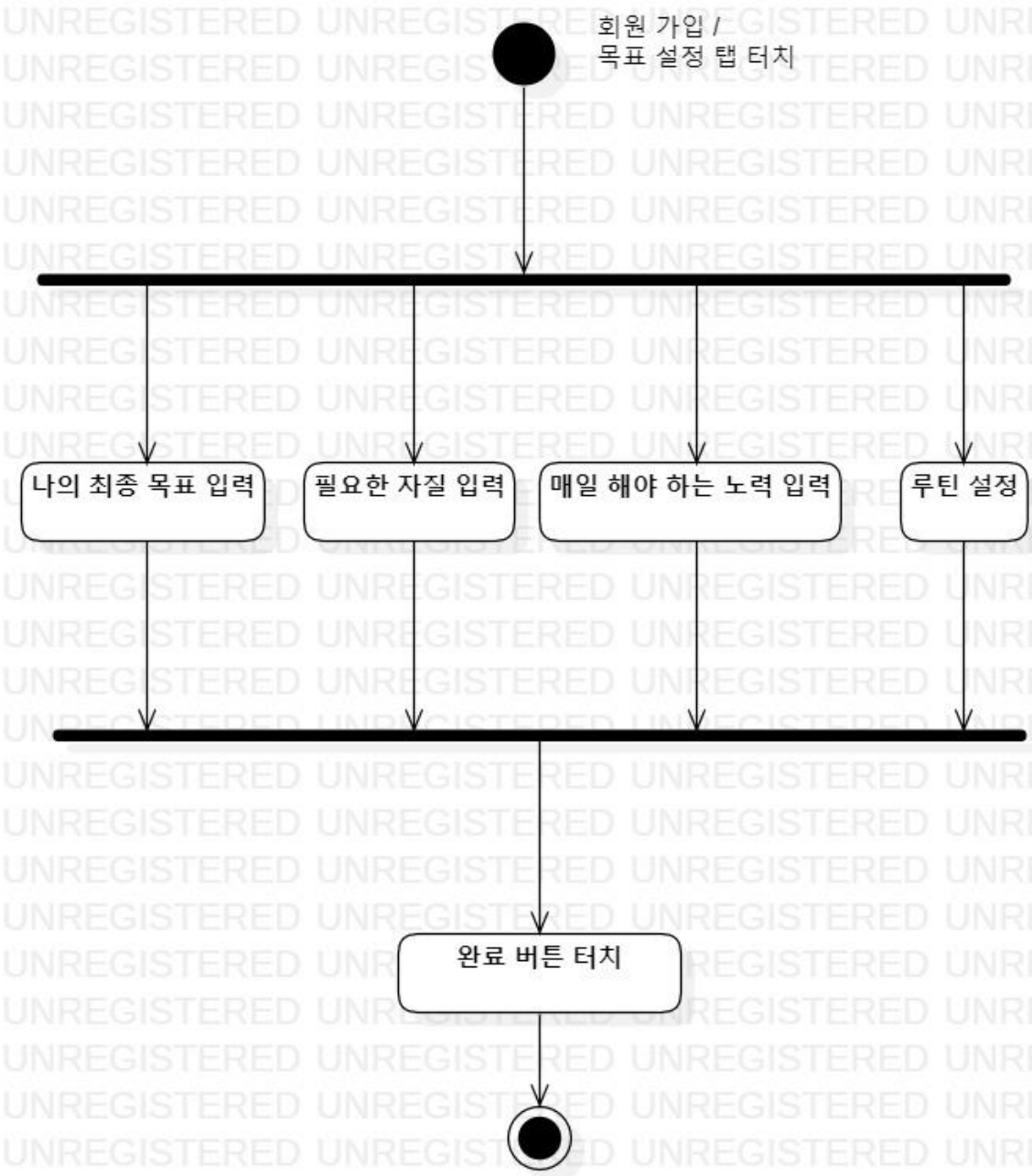


사용자는 앱 실행 이후 초기 실행 시, 회원가입과 이후 목표설정의 과정을 거친다. 회원가입은 안드로이드 기기의 경우 카카오 계정을 통해 회원가입을 진행할 수 있고, 애플 기기의 경우는 카카오 계정 또는 애플 계정을 통해 회원가입을 진행할 수 있다. 이미 회원이거나 회원가입을 마친 경우, 메인 화면에서 트레이닝 파트와 컨디셔닝 파트를 선택하여 기록을 조회하거나 기록을 새로 작성할 수 있고, 또는 자신의 프로필이나 설정했던 목표를 수정할 수 있다. 목표 설정에서 설정한 루틴 항목은 트레이닝 기록 파트에 연동되어 매일 기록 작성시에 체크할 수 있게 나타난다. 트레이닝 기록과 컨디셔닝 기록은 화면 상단에 있는 날짜 스크롤 UI를 통해 다른 날짜의 기록을 조회하거나 수정할 수 있다.

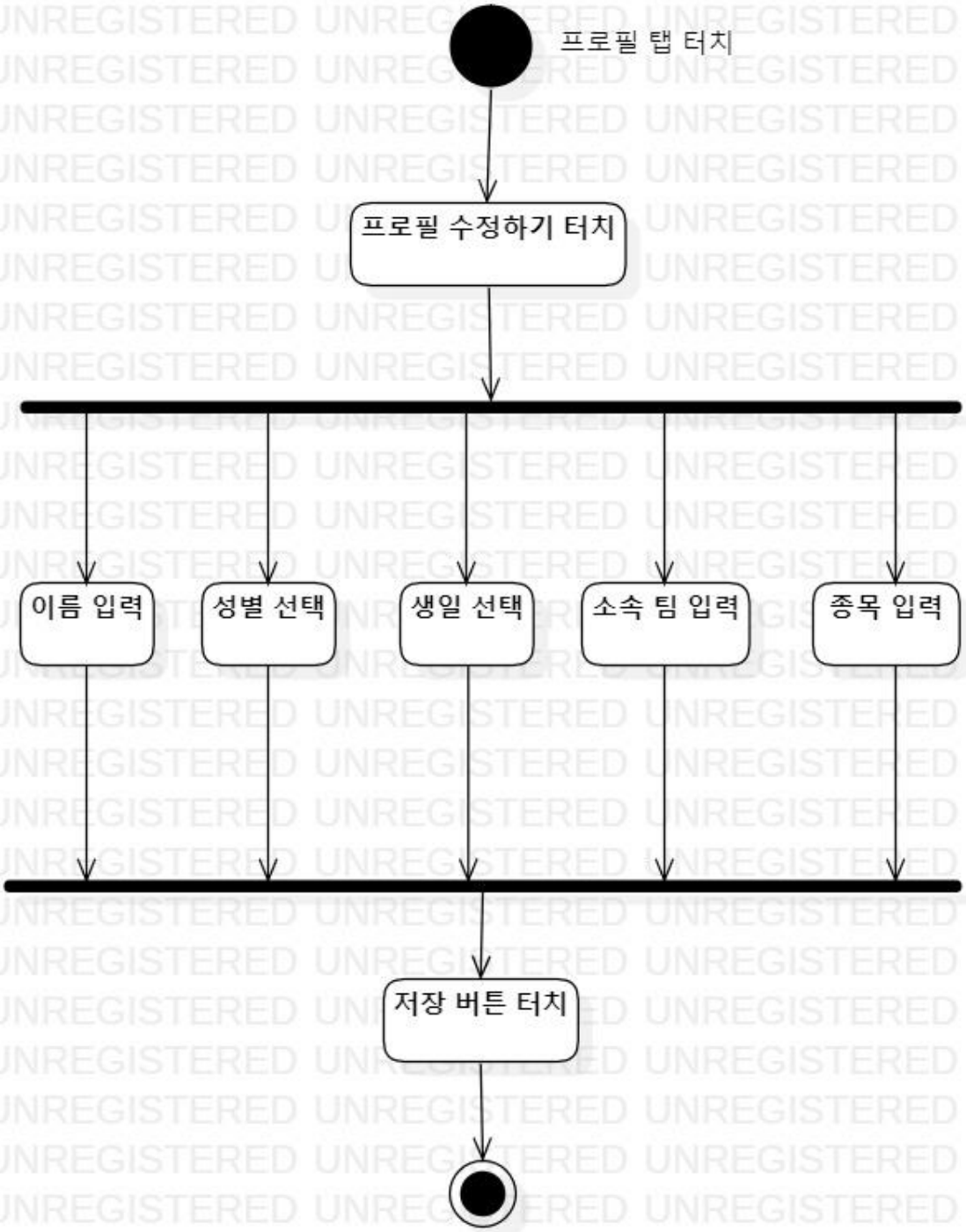
3.3.2.2 Activity Diagram



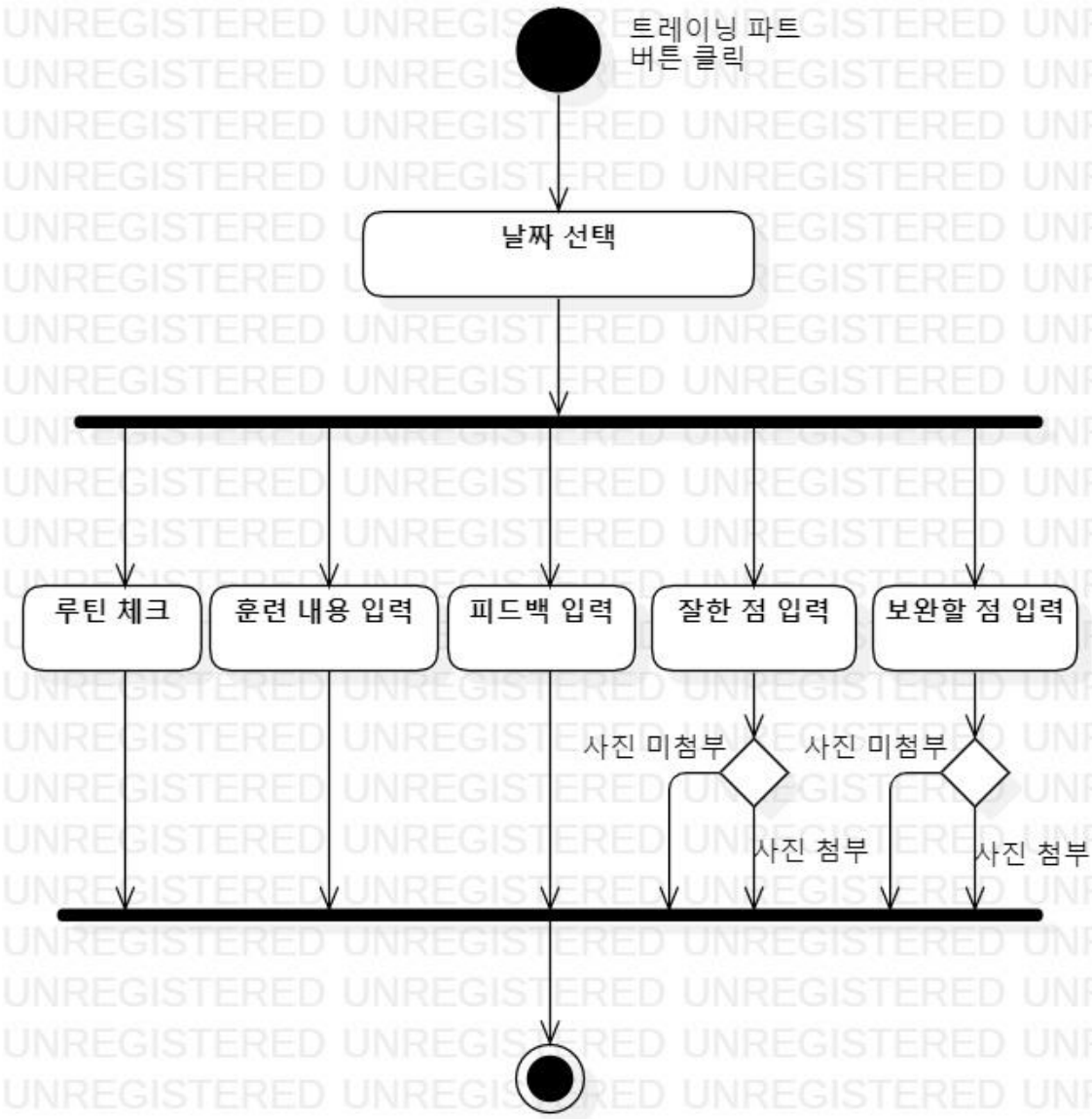
최초 앱 실행 시에 회원가입(로그인) 방식을 선택하고 로그인하는 과정을 나타낸 액티비티 다이어그램이다. 모종의 이유로 인증에 실패하는 경우는 로그인 화면에서 로그인을 재시도해야 한다.



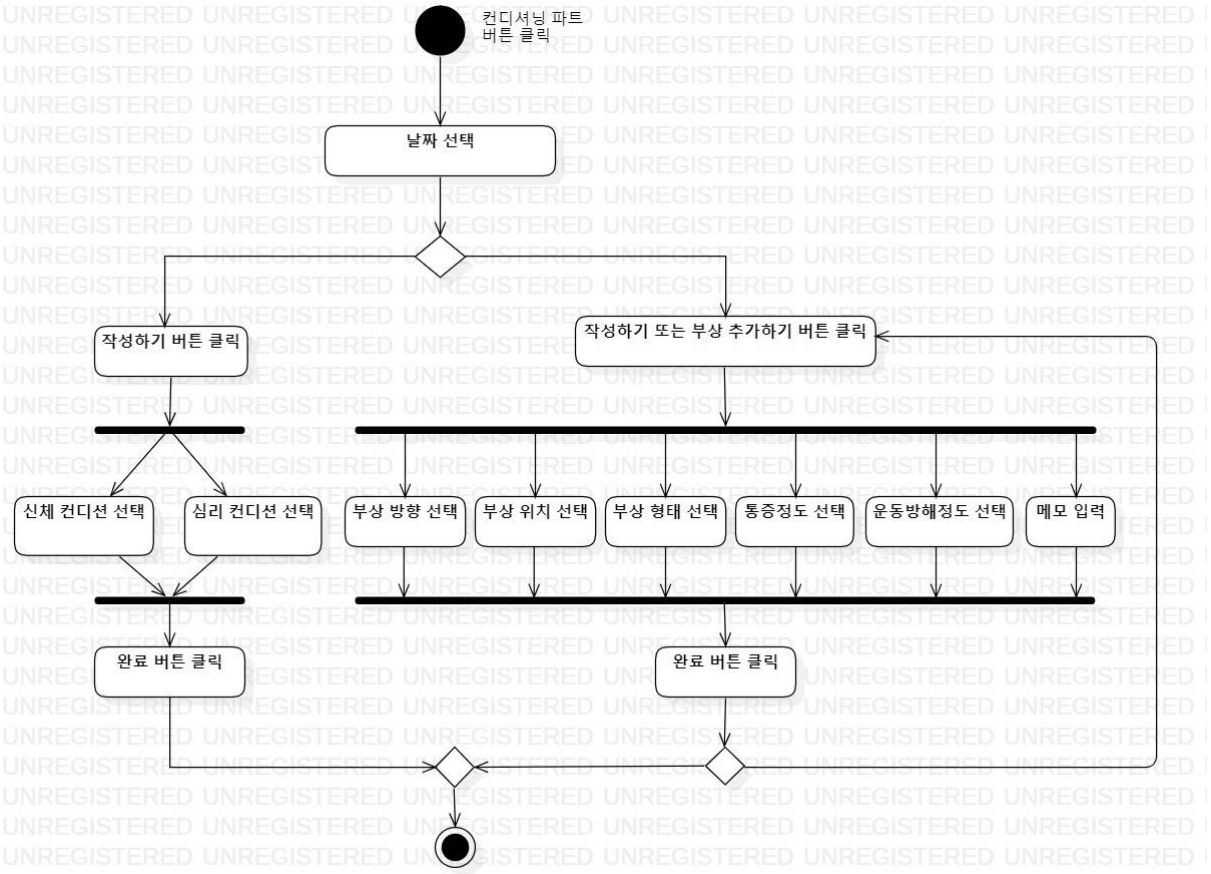
회원 가입을 완료한 경우 목표 설정 페이지가 화면에 출력되어 사용자로부터 입력을 받는다. 이 때 입력할 정보들은 공란으로 둘 수 있으며, 추후에 메인 화면 하단의 목표설정 탭을 눌러 목표 설정 페이지로 이동하여 추가하거나 수정할 수 있다.



사용자 프로필 정보는 화면 하단의 프로필 탭을 누르면 나오는 프로필 페이지에서 '프로필 수정하기' 항목을 선택하여 상시 수정할 수 있다.

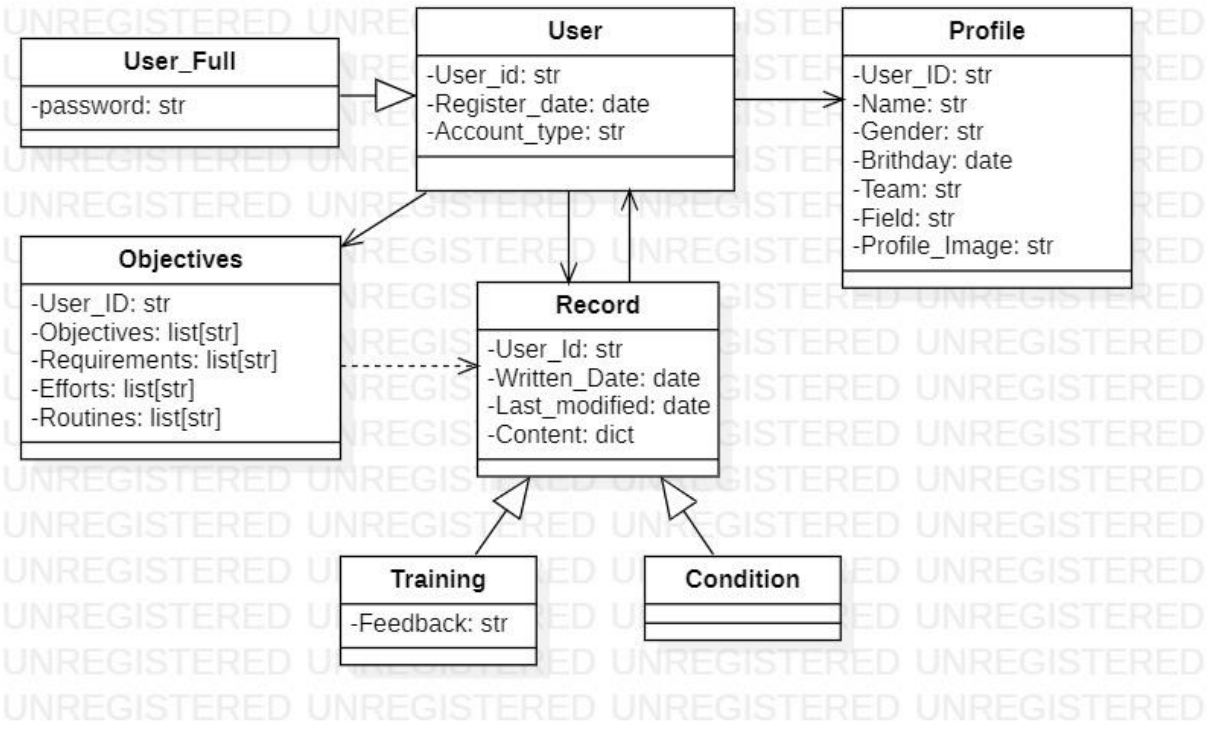


본 서비스의 메인 기능 중 하나인 트레이닝 일지 기록 파트이다. 루틴 체크는 목표설정에서 입력했던 루틴들을 지켰는지 체크하도록 연동되어 나타나는 항목이다. 잘한 점과 보완할 점 항목에서는 핸드폰 갤러리에 저장된 사진을 업로드 할 수 있다.



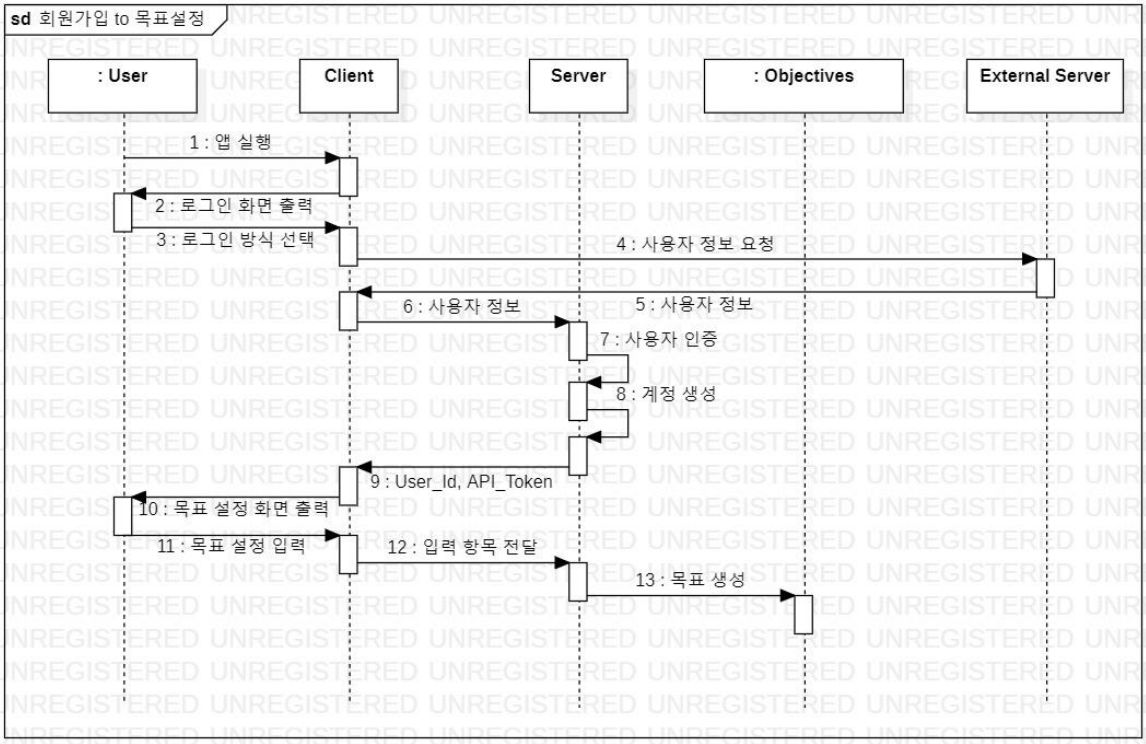
컨디셔닝 파트는 부상 항목을 여러 개 생성하고 기록할 수 있어서 반복구조를 띈다. 신체 컨디션과 심리 컨디션의 모든 입력 옵션들은 버튼으로 되어있고 여러 개 선택이 가능하다. 부상 리포트는 부상의 방향, 위치 형태를 목록에서 선택하고 통증의 정도와 운동방해 정도를 스크롤로 수치 기록을 할 수 있게 구성 되어있고 각 부상에 대한 메모를 남길 수 있다.

3.3.2.3. Class Diagram

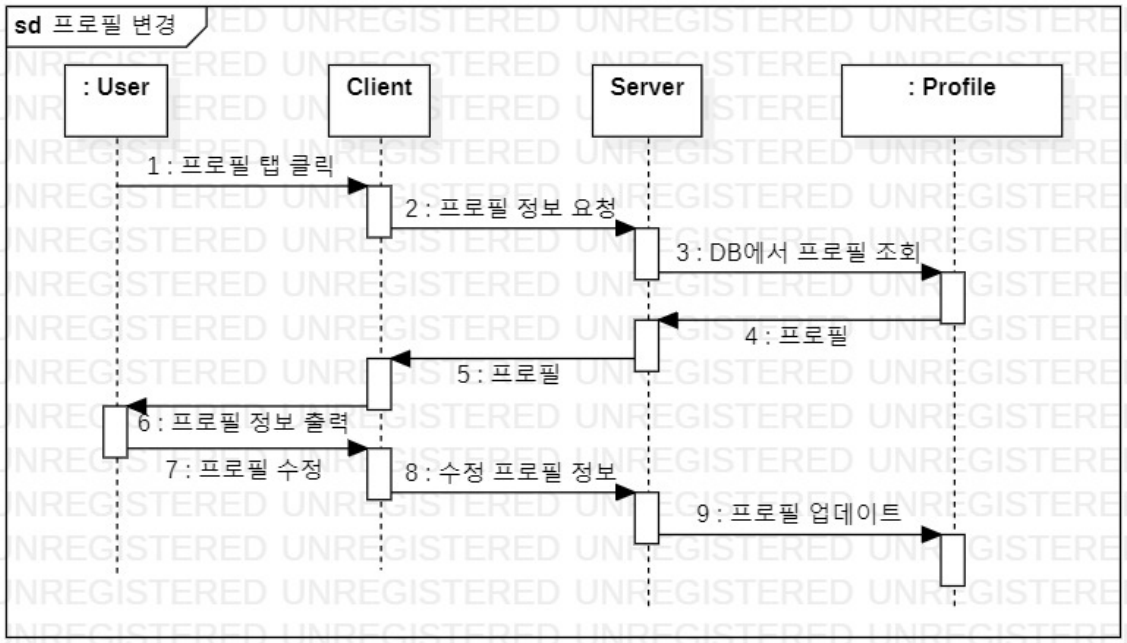


클래스는 User, Objectives, Profile, Record와 User의 child class인 User_Full과 Record의 child class인 Training과 Condition이 있다. User_Full은 User 인스턴스를 반환할 경우 보안을 위해 password(API_Token)를 제외한 정보들 만을 반환하기 위해 password를 가지고 있는 child class를 생성하였다. DB에 새로 유저를 생성할 때는 User_Full 인스턴스를 사용하여 저장되고, 일반적인 경우에는 User 클래스의 인스턴스를 이용하여 데이터의 전달이나 갱신이 이루어진다. Record는 트리이닝 기록의 Training과 컨디셔닝 기록의 Condition의 공통부분을 일반화한 부모 클래스이다.

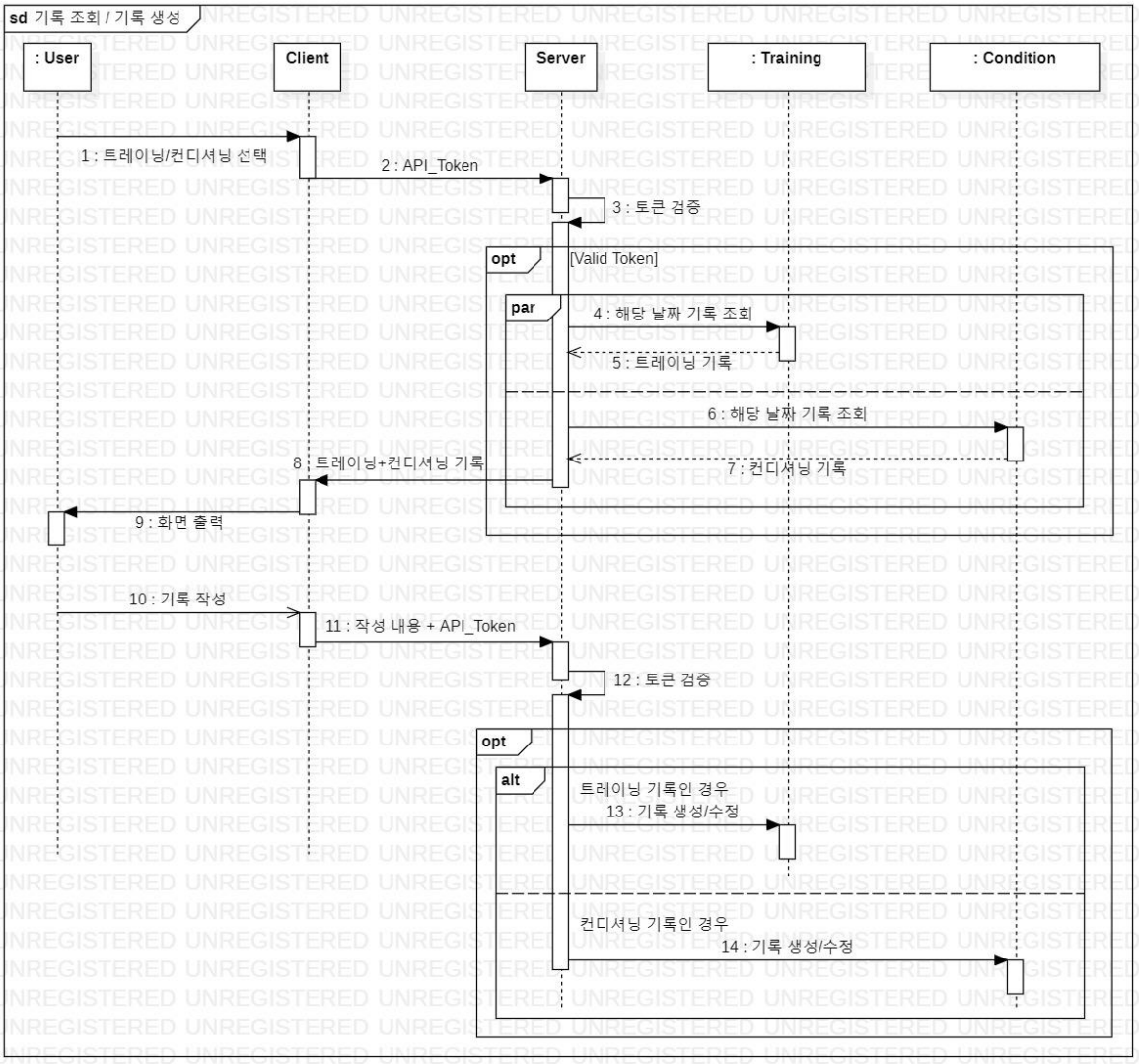
3.3.2.4 Sequence Diagram



회원가입부터 목표 설정까지 과정을 시퀀스 다이어그램으로 나타낸 것이다. External Server는 회원가입 유형에 따라 카카오 서버 혹은 애플 서버를 가리킨다.

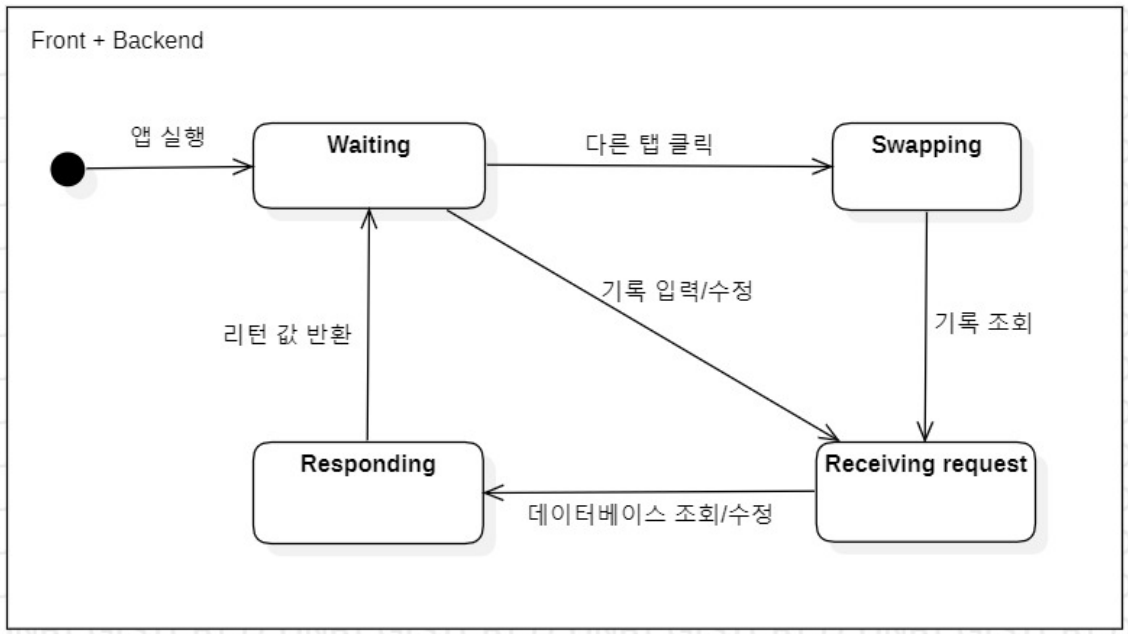


프로필 변경을 시퀀스 다이어그램으로 나타낸 것이다. Profile은 DB의 Profile 테이블을 나타낸다.



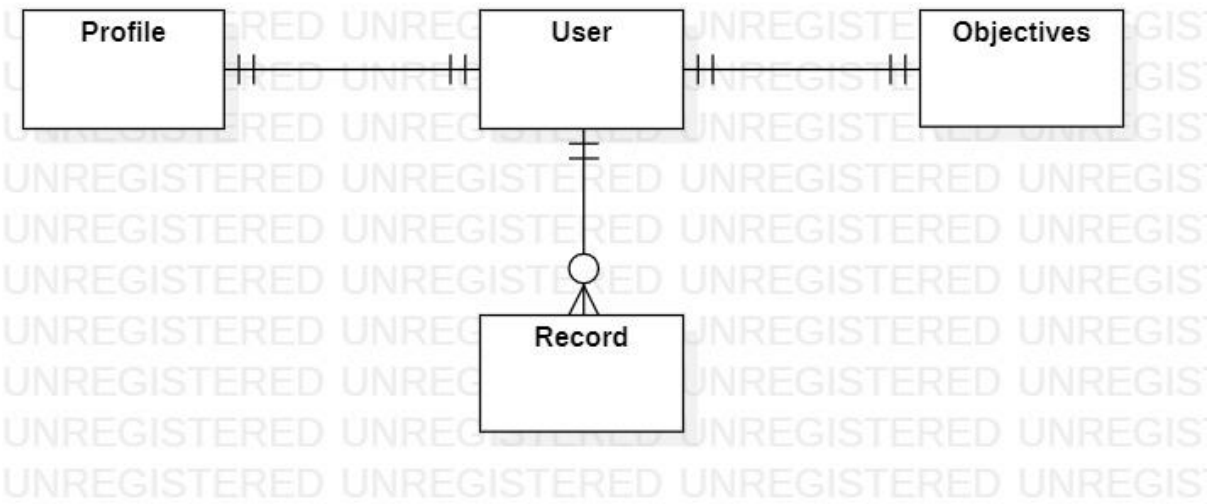
기록 조회 및 기록 생성 과정을 시퀀스 다이어그램으로 나타낸 것이다. 기록을 조회하거나 새로 기록을 작성할 때 클라이언트는 서버에 회원가입 시 발급받은 토큰을 함께 보낸다. 이 토큰을 서버에서 검증함으로써 User Validation을 한다. Invalid한 Token이 왔을 경우 해당 서버는 해당 요청을 수행하지 않는다.

3.3.2.5 State Diagram



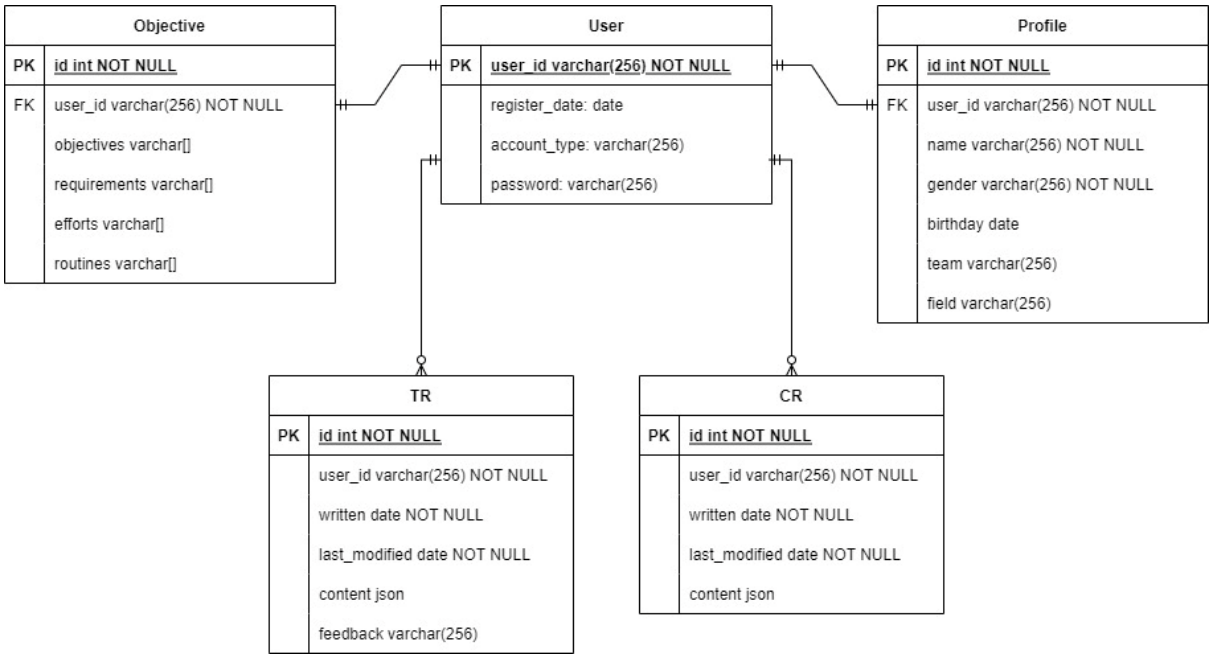
프런트 와 백 엔드를 묶어서 앱 서비스 자체의 State를 단순하게 표현한 State Diagram은 다음과 같다. 앱을 실행하면 유저의 행동을 기다리는 Waiting 상태로 시작한다. 다른 탭을 클릭하면 Swapping 상태로 되고, 이동한 화면의 기록을 조회하기 위해 Receiving Request 상태가 된다. 이 상태에서 데이터 베이스를 조회/수정하면 responding 상태가 되고, 요청에 대한 리턴 값을 반환하고 나면 다시 Waiting 상태가 된다.

3.3.2.6 ER Diagram



ER Diagram을 단순하게 표현하면 다음과 같다.

3.3.2.6.1 Table information



테이블의 정보를 담아서 ER Diagram을 세부적으로 나타낸 다이어그램이다. 유저당 하나씩 Objective와 Profile 레코드를 생성하게 되므로 1대 1 관계이고, 유저 한 명이 여러 트레이닝, 컨디셔닝 기록을 남길 수 있으므로 1대 다 관계이다.

3.4 구현

3.4.1. Frontend:

3.4.1.1 구성

[Github 주소]: https://github.com/PodiumDreaming/dreamingPodium-front

3.4.2.1. 구성

소스 코드 패키지 구성은 다음과 같다.

Index.js: 프로그램을 실행시키는 메인 소스코드이다. 템플릿 및 자바스크립트의 컴포넌트를 조합하여 렌더링하고 실제 표시 한다.

/android : 안드로이드 네이티브 코드 파일이다. 네이티브 설정 , 권한 및 소셜 로그인 설정 등을 수정한다.

/ios. : IOS 네이티브 코드 파일이다. 네이티브 설정, 권한 및 소셜로 그인 설정 등을 수정한다.

/App : 메인 파일이다.

/assets: 이미지, SVG, 폰트를 보관하는 파일이다.

/Components : 각 페이지에 쓰이는 컴포넌트를 보관한다 . 크게 컨디셔닝, 트레이닝, 공통 컴포넌트로 분리하고 공용 컴포넌트는 언제든 재사용 할 수 있도록 확장성 있게 제작한다.

/conditioning : 컨디셔닝 파트에 쓰이는 컴포넌트를 보관하는 파일이다. 컨디셔닝 파트에 직접적으로 쓰이는 컴포넌트의 파일명은 Condition 이라는 prefix를 사용하고 컨디셔닝 파트의 컴포넌트 혹은 스크린의 부분 재료로 쓰이는 파일명은 App 이라는 prefix를 사용하여 구분한다.

/training : 컨디셔닝 파트에 쓰이는 컴포넌트를 보관하는 파일이다. 트레이닝 파트의 컴포넌트 혹은 스크린의 부분 재료로 쓰이는 파일명은 App 이라는 prefix를 사용하여 구분한다.

AppCalendar.js , AppModal.js , AppPicker. Js ... 해당 파일들은 모든 스크린에서 사용되는 공용 컴포넌트이다. 특히 외부 라이브러리를 사용한 경우 편리한 사용성과 이해를 위해 컴포넌트로 한 번 감싸는 과정을 거쳤다.

/Config : 스타일속성, 질환리스트 등 정적인 정보를 모아놓은 파일이다.

- globalStyles.js : 색상, 폰트 크기, 폰트 , 화면 크기 , 이미지 등을 선언하여 모든 페이지에서 일관성 있게 사용 할 수 있도록 하였다.
- TermsScreen.js : 약관 html 코드를 보관한다.

/Navigation: 스크린 전환을 위한 내비게이션 , 스택을 정리한 파일이다

- **BottomTab.js** : 하단 바텀 네비게이션에 스크린을 정의한다. 홈화면, 목표설정, 프로필 스크린이 그 것이다.
- **DepthStack.js** : 기본 스택에서 접근할 스크린을 정의한다. 글쓰기 화면, 프로필 수정화면 등이 그 것이다.
- **DreamStack.js , HomeStack.js, ProfileStack.js** : 바텀탭에 들어갈 스택이다. 각 스택들은 필요한 스크린을 쌓아 이동할 수 있다.

/reducer: : Redux를 위한 파일이다. Createasyncthunk를 사용하여 긴 코드를 줄일 수 있었다. 해당 상태값들을 이용하여 백엔드와 많은 상호작용 없이도 프론트내에서 처리할 수 있도록 하였다. 이 때, 서비스가 과도하게 무거워지는 것을 방지하기 위해 최적화 작업에 신경썼다.

- userSlice.js : 유저 정보에 해당하는 상태와 액션등을 정리하고 비동기 처리를 통해 초기값을 지정한다.
- modalSlice.js : 모든 페이지의 모달을 조정한다. 모달 visible 여부와 모달 내의 화면을 직접 주입할 수 있다.
- postingSlice.js : 기록에 관한 모든 상태값을 관리한다. 초기값을 서버측 Get 요청 구조와 동일하게 잡고 서비스가 렌더될 시 해당 값들을 한 번만 불러온다.

/screens : 스크린을 모아둔 파일이다. 각 스크린에서 필요한 컴포넌트들을 조합하여 사용한다. 스크린들은 스택에 조합되어 사용된다.

/Auth

LoginScreen.js : 로그인 스크린이다.

loginApple.js, loginKakao.js : 소셜로그인을 위한 프로세스를 정의해둔 파일이다. 추

/Depth: 컨디셔닝, 프로필 수정 , 트레이닝 , 글쓰기 스크린을 모아둔 파일이다.

/Home : 홈화면, 프로필화면 ,목표설정 화면등 바텀탭에 해당하는 스크린을 모아둔 파일이다.

/utils: : Axios 인스턴스를 생성하고 모든 통신을 지정해 둔 파일이다. 미리 통신을 지정해 둠으로써 디버깅과 수정에 용의하게 하고자 하였고 각 페이지에서 쉽게 사용할 수 있도록 하였다.

- **api.js** : axios 인스턴스를 생성한 파일이다. Base Url은 환경변수로 지정하여 보안을 관리한다. 요청시 에러 핸들링과 헤더를 지정하여 공통적으로 사용할 수 있게 하였다.
- **auth.js** : 로그인을 위한 통신을 정의한 파일이다. 각 통신들은 스크린 페이지에서

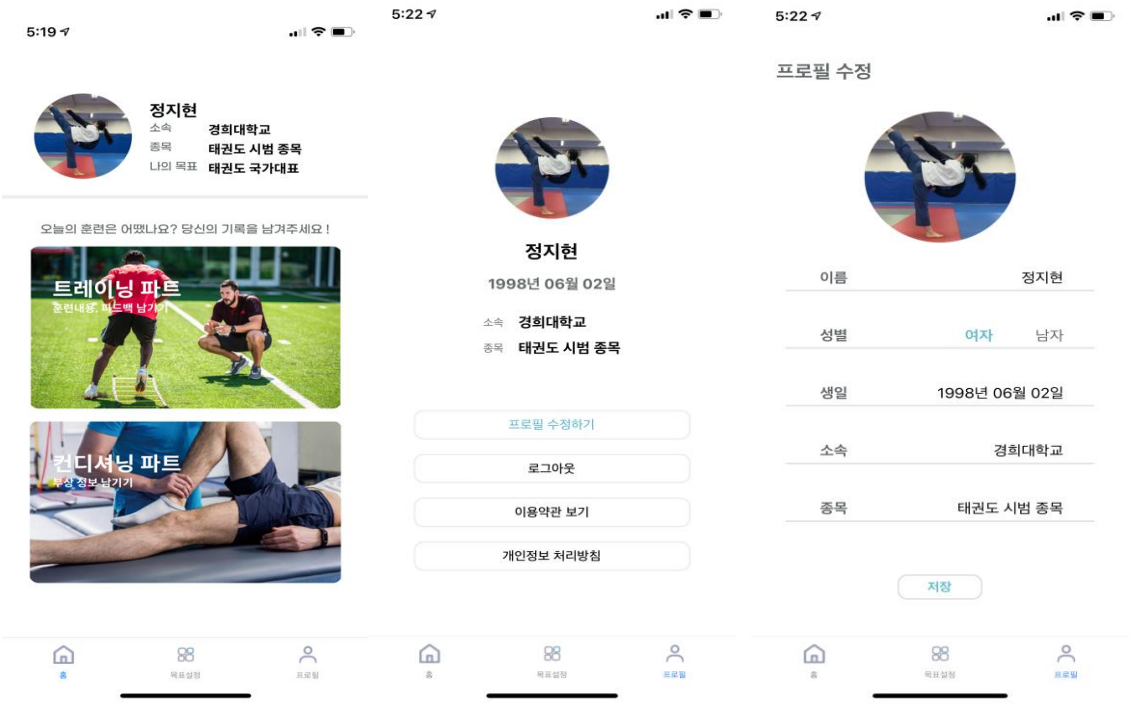
함수형태로 사용할 수 있다.

- **note.js** : 기록을 위한 통신을 정의한 파일이다.
- **profile.js :** 프로필을 위한 통신을 정의한 파일이다.

index.js : 렌더링 할 네비게이터와 스크린등을 묶어놓은 APP 컴포넌트가 위치한 파일이다. 실제 화면에 표시되는 내용 등은 여기에서 정의된다.

store.js : redux sotre 전체 전역 변수를 모아서 관리한다.

3.4.1.2. 구현 내용



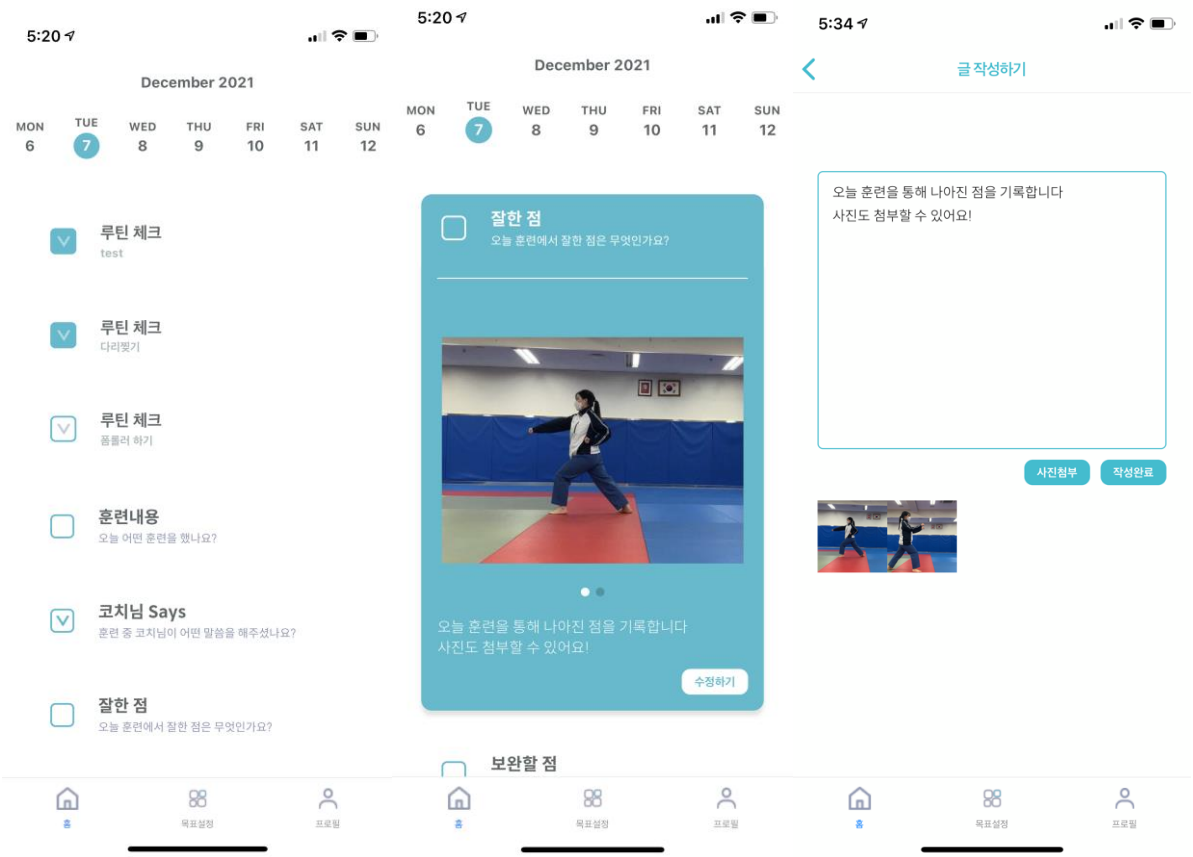
[대시보드 페이지, 프로필 페이지, 프로필 수정 페이지]

대시보드는 '트레이닝 파트'와 '컨디셔닝 파트' 로 진입할 수 있는 두 개의 탭이 출력된다.

대시보드 상단 프로필 구역과 프로필 탭의 정보를 동기화 시켜 사용자에게 보여준다. 이때 불필요한 데이터 fetch 횟수를 줄이고 사용감을 높이기 위해 프로필 데이터는 전역 state 로 관리하였다.

프로필 이미지는 'photo' 타입으로 제한하며 form data 형식으로 서버에 전달한다.

모든 버튼은 사용자 인지를 위해 터치 시 opacity 가 변경되어 feedback 을 줄 수 있게 구현하였다.



[트레이닝 파트, 글쓰기 페이지]

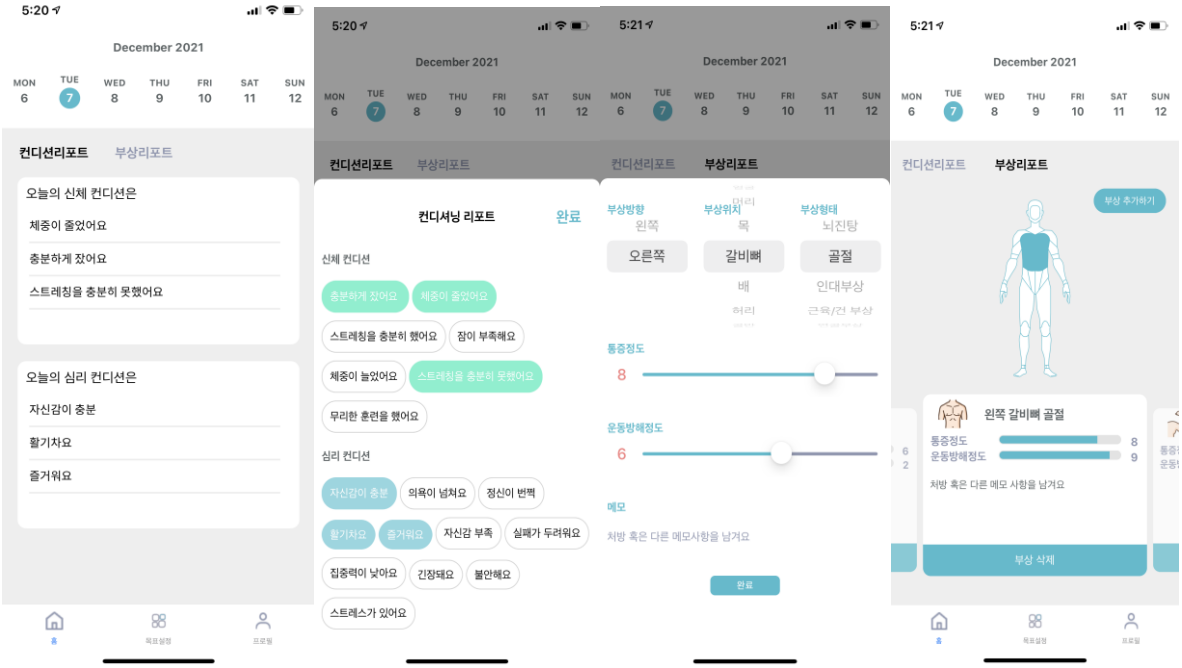
트레이닝 파트의 상단에는 날짜를 조정할 수 있는 캘린더를 배치하였다.

'루틴 체크' 부분에는 하단 탭에 있는 '목표설정' 페이지에서 사용자가 작성한 루틴이 보인다. 매일 유저가 지정한 루틴을 행했는지 체크할 수 있다.

'훈련 내용' ,'코치님 says', '잘한 점' , '못한 점' 은 작성되어 있지 않을 경우 글쓰기 페이지로 넘어간다. 글쓰기 페이지에서는 글과 여러 개의 사진을 첨부할 수 있다.

글을 작성하면 해당 주제를 터치할 시 아래로 펼쳐진다. 이 때, 사진이 여러 장 있을 경우 스와이퍼가 출력된다. 수정하기로 진입할 시 글과 사진 삭제가 가능하다.

유저가 한 주제에 해당하는 글을 작성할 때마다 서버로 전송된다. 하지만 따로 서버에 update된 내용을 받아오지 않고 redux (전역 상태 관리 미들웨어)로 프론트 내에서 처리한다. 이로써 사용자는 매끄러운 UI/UX를 경험할 수 있다.



[컨디셔닝 파트, 컨디셔닝 입력 Bottom madal]

컨디셔닝 파트는 선수들의 질적인 정보를 입력하는 공간이다. 리포트 형식이 떠오르는 UI를 차용했고 많은 정보를 빠르게 입력할 수 있는 UI를 연구하였다. Bottom modal 과 Select tab을 이용하여 depth 없이 정보를 입력할 수 있도록 구성하였다.

통증 정도와 운동 방해 정도는 슬라이더를 이용해 객관적 수치를 나타낼 수 있도록 하였다.

컨디셔닝 파트 또한 제출과 동시에 서버로 정보가 전달이 되지만 프론트 내의 전역 관리를 통해 별다른 요청 없이 사용자에게 입력한 정보를 보여준다.

3.4.2. Backend:

[Github 주소]: https://github.com/PodiumDreaming/Dreaming_Podium_Backend

3.4.2.1. 구성

소스 코드 패키지 구성은 다음과 같다.

main.py: 프로그램을 실행시키는 메인 소스코드이다.

app: 나머지 서브 패키지들을 묶은 패키지이다.

app/util.py: 시간 변환 같은 일부 나른 소스 코드에서 사용하는 함수들을 선언해 놓은 파일이다.

app/config/config.py: 데이터베이스 주소나 알고리즘 secret key등의 환경변수를 불러오는 파일이다.

app/database: 데이터베이스와 관련된 작업을 행하는 소스코드 파일들을 묶은 패키지이다.

- **Model.py:** Object Relational Mapping(ORM)을 통해 데이터베이스 테이블로 전환이 가능한 Pydantic model을 선언한 파일이다. Pydantic 라이브러리를 통해 입력 데이터의 검증을 편리하게 할 수 있다.
- **Tables.py:** SQL Alchempy ORM 라이브러리를 사용하여 생성할 테이블의 스키마를 정의한 파일이다.
- **conn.py:** AWS RDS와 S3로 연결하는 인스턴스를 생성하는 함수를 정의한 파일이다.
- **crud.py:** HTTP method의 데이터베이스로의 실질적인 Create, Read, Update, Delete 오퍼레이션을 구현해 놓은 파일이다.

app/router: HTTP method(GET, POST등)들을 핸들링 하는 함수들이 정의된 파일들이 있는 패키지이다.

- **Apple.py:** 애플 계정을 통한 로그인을 처리한다.
- **Images.py**: AWS S3버킷으로의 사진 업로드를 처리한다.
- **KaKao.py:** 카카오 계정을 통한 로그인을 처리한다.
- **Objective.py:** 목표 설정에 관한 함수들이 정의된 파일이다.
- **Profile.py:** 사용자 프로필에 관한 함수들이 정의된 파일이다.
- **Record.py:** 트레이닝, 컨디셔닝 기록에 관한 함수들이 정의된 파일이다.

3.4.2.2. 구현 환경

백 엔드는 Python 언어를 사용하여 FastAPI라는 웹 프레임워크를 사용하여 만들어졌다. 서버는 AWS EC2 Ubuntu 서버 위에서 구동되고 있으며, 데이터베이스는 AWS RDS 서비스를 통해 MaridaDB를 사용하고 있다. 이미지 파일 업로드를 위해서는 AWS S3 버킷을 이용하고 있다.

FastAPI는 작성한 코드를 자동으로 문서화해주는 편의기능을 제공한다. 현재 API에 대한 documentation은 http://3.35.43.76:8000/docs#/ 으로 접속하여 사용방법을 확인하거나 클라이언트 없이도 테스트해보는 것이 가능하다. 단, 소셜 로그인 같은 몇몇 API는 클라이언트에서 파라미터로 외부 서버에서 받은 데이터를 필요로 하므로 docs 페이지에서 테스트가 불가능한 경우가 있다. Github에 README.MD 파일을 통해 로컬에서 개인적으로 테스트하고자 하는 사람들을 위한 환경 구성 단계를 작성하였다.

4. 프로젝트 결과

4.1. 서비스 구현 결과

4.1.1. 개발도구 및 환경

Front-end: 정지현
개발 언어: JavaScript
개발 도구: React Native, Xcode, Android Studio

Back-end: 정민혁
개발 언어: Python
프레임워크: Fast API
서버: Linux: Ubuntu AWS EC2
DBMS: Maria DB on AWS RDS
이미지 업로드 서버: AWS S3

4.1.2. 초기 계획 대비 변한 내용

초기 요구사항 분석시에는 컨디셔닝 파트의 부상 기록을 다음과 같이 계획했었다.

- 신체 주요 관절 및 근육을 선택해 사용자가 자신의 부상 부위의 통증 정도와 운동 방해 정도를 입력할 수 있게 구현한다. 복수 선택 가능하게 구현한다.

기획팀의 의견을 반영하고 계획에 수정이 생기면서 구현 과정에서는 복수개의 세부 부상 기록을 남기는 형식으로 구현하였다.

4.1.3. 제작 기간 동안 겪은 어려움 및 극복과정

Frontend:

제작 기간 동안 프론트엔드가 겪은 어려움은 기획 및 디자인 부진이 주는 진행 차질 및 소통의 부재로 인한 잦은 수정이었다. 개발에 착수할 당시 기획과 디자인이 제대로 구상되어 있지 않았고 개발을 하며 기획 팀에게 물어보고 확인하는 과정이 계속되었다. 디자인 또한 개발을 하던 중 직접 수정하고 제작하는 일이 생겼다. 또한 개발에 착수할 당시 백엔드 측과 충분한 소통이 이루어지지 않았기 때문에 각자의 생각만을 가지고 구현하는 일이 생겼고 그렇기에 잦은 요청이 필요했는데 이 때 상태 구조를 크게 바꾸게 되어 이에 해당하는 로직들이 전부 에러가 생기는 일이 생겼다.

이러한 어려움을 해결하기 기획팀에게 구체적으로 의견을 제시하고 빠른 대응을 요구했다. 디자인 또한 프론트 개발을 하면서 직접 피그마 수정을 하는 등의 노력을 통해 개발을 이어나갔다.

두 번째 어려움은 기술적 어려움이었다. 서비스 구조상 여러 주제의 글을 쓰다 보니 각 글을 쓸 때마다 서버에 정보를 보내고 업데이트 내용을 받아오는 게 over Fetching 이라고 생각되었다. 또한 props 를 처리해야 할 내용이 많았기 때문에 전역 상태 관리가 필요하다고 생각이 들어 Redux 를 도입했다. Redux 를 사용하며 전역 관리와 서버 요청을 관리하는 것이 어려웠지만 CreateAsync 미들 웨어를 통해 Redux 비동기 처리를 진행하였다.

Backend:

웹 프로그래밍에 큰 관심이 없어서 웹 프레임워크를 제대로 다뤄본 적이 없는데 그나마 공부를 조금 하고 있었던 프레임워크 Django 가 아닌 FastAPI 를 사용하면서 프로젝트 초기에 웹 지식과 프레임워크 자체를 익히느라 개발이 늦춰졌고, 프런트 엔드와 속도를 맞출 수 없었다. 하지만 FastAPI 가 굉장히 사용이 편리하고 배우기 쉬웠기 때문에 기본 사용법을 익힌 이후로는 프런트 엔드에서 테스트를 위해 우선 필요한 API 를 물어보고 우선 개발하여 개발 템포를 맞출 수 있었다.

또 다른 난관은 수정에 대한 요청과 이로 인한 에러의 연속이었다. API 를 만들고 프런트와 테스트를 하였으나 프런트 쪽 사용의 편의성을 위해 프런트에서 일부 API 의 수정을 부탁받았다. 처음 수정하는 것은 어렵지 않을 거라 생각했다. 요구받은 형태로 수정을 하다 보니 이미 선언된 모델이나 테이블의 구조를 변경해야 하는 경우도 있었고, 입력 파라미터 형식을 바꾸는 경우도 있었다. 그러다 보니 소스코드 내에서 데이터 형식의 일관성이 깨지면서 일부 소스코드에 대한 형식 때문에 발생한 에러나 문법오류들로 인해 오랜 시간 개발이 지체되는 일이 있었다. 이는 바쁜 와중에도 서로 지속적으로 zoom 을 통해 실시간으로 통신 테스트를 하면서 어떤 종류의 에러가 발생하는지 확인하면서 극복해냈다.

협업:

개발 협업은 FE 와 BE 사이에서 이루어졌다. 처음 개발 단계에서는 각자 진행할 부분을 독립적으로 진행했다. 시간이 흐른 후 프론트와 백엔드를 연동해서 테스트해보니 데이터 구조가 맞지 않는 등의 문제가 생겼다. 또한 서로 다른 개발 성격때문에 개발 단계에서 오해가 생기기도 하였다. 결국 이 모든 것은 소통의 부재, 소통 방법의 잘못됨에서 비롯됨을 깨닫고 'Notion', 'Slack' 등의 협업 툴을 이용하였다. 자신이 진행하고 있는 부분을 상대가 언제든지 알 수 있도록 공유하고 알리는 규칙을 만들었다. 이러한 노력으로 효과적인 협업을 할 수 있었다.

4.1.4. 앱 릴리즈

구글 플레이스토어와 애플 스토어 심사가 모두 통과 되었다. 심사 reject 때문에 배포 기간 연기를 우려했으나 심사를 넣기 전 많은 검토를 한 덕분에 큰 reject 사유 없이 통과할 수 있었다.

배포 후 경희대학교 태권도 학과 선수단 테스터들을 통해 앱 테스트 기간을 갖고 있다.

5. 결론

5.1 기대효과

핸드폰을 통해 편리하게 언제, 어디서나 훈련 일지를 작성할 수 있다. 다이어리를 들고 다니면서 훈련 일지를 작성하는 것은 불편할 수 있으나, 현대인의 대부분은 거의 항상 핸드폰을 지니고 다니기 때문에 언제라도 훈련 일지를 기록할 수 있다. 또한 일반적으로 글로 훈련 일지를 작성하는 것은 필기구가 필요할 뿐만 아니라 버튼을 통해 이미 유형화 되어있는 내용을 기록하거나 타이핑하는 것보다 훨씬 오랜 시간이 들기 때문에 시간 절약 효과를 가져온다.
또한 이로 인해 사용자가 훈련 일지를 작성하는 것을 덜 번거롭게 여기고 훈련 일지 작성에 임하게 된다. 지속적인 기록은 사용자 스스로 훈련 및 신체, 심리, 부상 상태를 꾸준하게 모니터링 하게 해준다. 따라서 사용자는 훈련의 효과가 향상됨은 물론, 자신에게 이상 상태가 발생했거나, 상황이 악화되는 것을 조기에 발견하고 건강 회복 및 재활 훈련을 받을 수 있다.

5.2 발전 가능한 기능

운동 선수들의 컨디션 관리를 위해 다양한 기능을 고려해볼 수 있다.

첫 번째로 코치님 관리 버전을 추가할 수 있다. 운동 선수들의 기록을 코치님이 한 눈에 보고 직접 피드백을 달 수 있다. 코치님은 자신의 선수들의 컨디션과 부상 기록을 함께 관리하고 의학적 피드백을 줄 수 있다.

두 번째로 컨디셔닝 파트에 통계를 시각적으로 보여줄 수 있도록 한다. 현재 선수의 질적 기록을 받고 있는데 이 기록들이 쌓이면 주간, 월간 별로 선수들의 컨디션과 부상 상태를 모을 수가 있다. 통계를 시각화하여 보여줌으로써 유저는 해당 기록을 통해 팀닥터와의 상담등을 편리하게 할 수 있다.

세 번째로 사용자 편의를 고려하여 다른 소셜 로그인 기능을 추가로 도입하는 방안이 있다. 네이버, 구글 등의 사용자가 많은 서비스 API 를 추가 사용하여 회원가입을 확장할 수 있다.

네번째로 모은 통계자료를 정리하여 동의한 사용자를 대상으로 회원가입 계정의 유형에 따라 이메일로 뉴스레터 형식의 자료를 주기적으로 전달해주는 기능을 고려할 수 있다.

6. 참고문헌

[1] 석류; 임신자. 성찰적 훈련일지를 통한 반성적 사고수준의 변화 분석. 한국여성체육학회지, 2016, 30.3: 331-352.

[2] 허정훈. 운동선수 자기관리 측정도구의 구조적 타당화와 인과모형 검증. 박사학위논문, 중앙대학교 대학원, 2001.

[3] 허정훈. 운동선수 자기관리 검사지 (ASMQ) 개발. 한국스포츠심리학회지, 2003, 14.2: 95-109.

[4] 이명선; 박세윤. 육상 투척선수의 심리기술훈련 적용 효과. 한국스포츠심리학회지, 2011, 22.2: 77-92.

[5] 박해원. 진단분류모형을 적용한 운동선수 컨디션 6 관리 측정 척도 개발.국내박사학위논문 충북대학교, 2019. 충청북도

[6] 배프 저, 배프의 오지랖 파이썬 웹 프로그래밍

[7] wikipedia, Node.js, https://en.wikipedia.org/wiki/Node.js , last edited 12 September 2021, viewd 16 September 2021.

[8] what-does-micro-mean, https://flask.palletsprojects.com/en/2.0.x/foreword/#what-does-micro-mean , 2010, viewed 28 September 2021.

[9]FastAPI, https://fastapi.tiangolo.com/ko/#performance , viewed 16 September 2021.

[10]2020 년 Python 개발자 설문조사 결과, https://www.jetbrains.com/ko-kr/lp/python-developers-survey-2020/ , viewed 28 Septemeber 2021.

[11]Redux, https://react-redux.js.org/ , viewed 21 September 2021.