모두의 달성

기획 제안서



Team4: NyangNyangPunch

Member:

201321001 박승현

201321011 장동익

201321028 조건희

201321035 황의권

목차

[1. 프로젝트 개요 4](#_Toc527748377)

[1.1 문제 정의 4](#_Toc527748378)

[1.1.1 꾸준한 실천이 필요한 목표 달성의 어려움 4](#_Toc527748379)

[1.1.2 목표 달성을 함께할 주변인의 부재 4](#_Toc527748380)

[1.1.3 계획 실천 여부 확인의 어려움 5](#_Toc527748381)

[1.2 문제 해결을 위한 솔루션 5](#_Toc527748382)

[1.3 주 타겟 및 STAKE HOLDER 6](#_Toc527748383)

[1.3.1 타겟 6](#_Toc527748384)

[1.3.2 이해관계자 6](#_Toc527748385)

[1.3.3 타겟 및 이해관계자가 얻는 가치 7](#_Toc527748386)

[2. 기존사례분석 및 사업화 가능성 8](#_Toc527748387)

[2.1 유사제품 및 서비스 분석 8](#_Toc527748388)

[2.1.1 옵티(Obti) 8](#_Toc527748389)

[2.1.2 윌뱅크 (Willbank) 9](#_Toc527748390)

[2.1.3 금연 시작 11](#_Toc527748391)

[2.2 유사제품과의 차별성 12](#_Toc527748392)

[2.2.1 사용자 경험에서의 차별 12](#_Toc527748393)

[2.2.2 기술적 차별성 13](#_Toc527748394)

[2.2.3 비즈니스 모델에서의 차별성 14](#_Toc527748395)

[2.3 사업화 가능성 14](#_Toc527748396)

[2.3.1 비즈니스 모델 14](#_Toc527748397)

[2.3.2 타겟 시장 분석 16](#_Toc527748398)

[2.3.3 예상 사업 규모 17](#_Toc527748400)

[3. USER STORIES 18](#_Toc527748401)

[3.1 주요 시나리오 18](#_Toc527748402)

[3.1.1 사용자 관점의 주요 시나리오 18](#_Toc527748403)

[3.1.2 관리자 관점의 주요 시나리오 22](#_Toc527748404)

[3.2 User Stories 23](#_Toc527748405)

[3.3 Product Backlog 25](#_Toc527748406)

[4. 구현 30](#_Toc527748407)

[4.1 필요 기술 30](#_Toc527748408)

[4.1.1 기술 스택 / 프레임 워크 30](#_Toc527748409)

[4.1.2 언어 / 개발 환경 34](#_Toc527748410)

[4.1.3 외부 기술 37](#_Toc527748411)

[4.2 시스템 구조 41](#_Toc527748412)

[4.2.1 전체 시스템 구조도 41](#_Toc527748413)

[4.2.2 소프트웨어 구조 42](#_Toc527748414)

[4.2.3 시스템 플로우 48](#_Toc527748415)

[4.3 API 정의 51](#_Toc527748416)

[4.3.1 외부 API 51](#_Toc527748417)

[4.3.2 Restful API 51](#_Toc527748418)

[4.4 데이터 정의 54](#_Toc527748419)

[5. 과제수행 계획 57](#_Toc527748420)

[5.1 일정 계획 57](#_Toc527748421)

[5.2 업무 분장 61](#_Toc527748422)

[5.2.1 개발분야 업무 분장 61](#_Toc527748423)

[5.2.2 비 개발분야 업무 분장 62](#_Toc527748424)

[5.3 협업방안 62](#_Toc527748425)

[5.3.1 Sprint 계획을 위한 구글 스프레드 시트 63](#_Toc527748426)

[5.3.2 Sprint 진행을 위한 Trello 사용 방안 64](#_Toc527748427)

[5.3.3 소스 변경 관리 및 코드 리뷰를 위한 Slack + Github 사용 방안 65](#_Toc527748428)

[5.3.4 이슈 관리를 위한 Slack + Github 사용 방안 66](#_Toc527748429)

[5.3.5 빌드 자동배포를 위한 Github + Jenkins 사용 방안 67](#_Toc527748430)

[5.4 성과 측정 방안 67](#_Toc527748431)

[5.4.1 정량적 KPI 67](#_Toc527748432)

[5.4.2 정성적 KPI 68](#_Toc527748433)

[[그림 1] 옵티(Obti) 9](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748542)

[[그림 3] 금연 시작[그림 1] 옵티(Obti) 9](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748543)

[[그림 2] 윌뱅크 (Willbank) 10](#_Toc527748544)

[[그림 3] 금연 시작 12](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748545)

[[그림 4] 비즈니스 모델 16](#_Toc527748546)

[[그림 5] 윌뱅크 (Willbank) 17](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748547)

[[그림 6] 금연 시작 17](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748548)

[[그림 7] 옵티 (Obti) 17](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748549)

[[그림 8] 회원가입 화면 20](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748550)

[[그림 9] 찾기 화면 20](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748551)

[[그림 10] 컨텐츠 참여 화면 21](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748552)

[[그림 11] 컨텐츠 참여 화면 (기간 선택) 21](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748553)

[[그림 12] 금연 컨텐츠 참여 현황 시나리오 22](#_Toc527748554)

[[그림 13] 본인 인증 시나리오 23](#_Toc527748555)

[[그림 14] 타인 인증 체크 화면 24](#_Toc527748556)

[[그림 15] MEAN Stack 33](#_Toc527748557)

[[그림 16] Android 35](#_Toc527748558)

[[그림 17] Kotlin 37](#_Toc527748559)

[[그림 18] HTML, Javascript, CSS 38](#_Toc527748560)

[[그림 19] WebStorm 39](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748561)

[[그림 20] Android Studio 39](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748562)

[[그림 21] Firebase Cloud Messaging 41](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748563)

[[그림 22] Firebase 41](file:////Users/parkseunghyun/Desktop/PSH/ajou/4-1/CapD/팀프/냥냥펀치-2.docx#_Toc527748564)

[[그림 23] HTTP Adaptive Streaming 42](#_Toc527748565)

[[그림 24] Exoplayer 43](#_Toc527748566)

[[그림 25] 전체 시스템 구조도 45](#_Toc527748567)

[[그림 26] 소프트웨어 구조 46](#_Toc527748568)

[[그림 27] MVC 패턴 50](#_Toc527748569)

[[그림 28] 로그인 시나리오에서 시스템 플로우 52](#_Toc527748570)

[[그림 29] 컨텐츠 참여 시나리오에서 시스템 플로우 53](#_Toc527748571)

[[그림 30] Sprint Backlog(Iteration 1) 64](#_Toc527748572)

[[그림 31] 협업 툴 간 상호작용 67](#_Toc527748573)

[[표 1] User story 26](#_Toc527748574)

[[표 2] Product Backlog 28](#_Toc527748575)

[[표 3] Sprint Backlog (Iteration2) 31](#_Toc527748576)

[[표 4] 외부 API 55](#_Toc527748577)

[[표 5] Restful API 56](#_Toc527748578)

[[표 6] User Schema 59](#_Toc527748579)

[[표 7] Content Schema 60](#_Toc527748580)

[[표 8] 개발 일정 61](#_Toc527748581)

[[표 9] Sprint Backlog (Iteration 2) 65](#_Toc527748582)

[[표 10] 개발 분야 업무 분장 66](#_Toc527748583)

[[표 11] 비 개발분야 업무 분장 67](#_Toc527748584)

[[표 12] 구글 스프레드 시트 설명 68](#_Toc527748585)

[[표 13] Trello 보드 설명 69](#_Toc527748586)

[[표 14] Github & Slack 사용 규칙 (소스 관련) 70](#_Toc527748587)

[[표 15] Github & Slack 사용 규칙 (이슈 관련) 71](#_Toc527748588)

[[표 16] 정량적 KPI 72](#_Toc527748589)

[[표 17] 정성적 KPI 73](#_Toc527748590)

# 1. 프로젝트 개요

## 1.1 문제 정의

### 1.1.1 꾸준한 실천이 필요한 목표 달성의 어려움

새해에는 많은 사람들이 각자 달성하고자 하는 목표를 야심 차게 세운다. 하지만 많은 사람들이 목표를 달성하지 못하고 실패한다. 일상생활에서의 목표에서도 마찬가지이다. 금연, 체중감량, 운동 등의 목표를 해내고 싶어하는 사람이 많지만, 매번 목표를 이루지 못하는 일이 반복된다. [[1]](#endnote-1)

사람들이 목표를 달성하지 못하는 이유는 대부분 자신의 의지를 쉽게 조절하지 못하기 때문이다. 인간의 뇌는 특별한 동기가 없이 단조로운 반복을 하면 쉽게 싫증이 나고 이것은 목표 달성의 실패로 귀결된다. 게다가 다른 사람들과 함께 목표를 달성하고자 하는 것과 달리, 혼자만의 목표 달성 계획은 다른 사람의 기대가 없고 쉽게 자신의 행동을 합리화하는 경향이 있어 실패하기 더 쉬운 문제점이 있다.[[2]](#endnote-2)

### 1.1.2 목표 달성을 함께할 주변인의 부재

목표 달성을 위해 많은 사람들이 자신과 함께할 동료를 찾는다. 이는 목표 달성을 위한 외적 동기를 얻기 위함이다. 목표 실천을 위해 정한 계획을 꾸준히 실천하고 있는 지 서로 확인해주고, 의지가 부족할 때 서로에게 의지를 북돋워 주기 위함이기도 하다. 하지만 목표 달성을 함께하고자 하는 동료를 주변에서 찾기란 쉽지 않다. 동료를 찾아 함께할 때에도 성공에 대한 보상, 실패에 대한 페널티가 없기에 각자의 실천의지가 확고하지 않다면 실천계획은 금방 흐지부지되고 만다.

### 1.1.3 계획 실천 여부 확인의 어려움

특정 목표들의 계획 실천 여부는 타인이 확인하기 굉장히 어렵다. 특히 금연의 경우, 누군가 온종일 감시를 해주는 것이 아니라면 본인이 정말로 금연을 실천 중 인지에 대해 인증하기가 어렵다. 함께 금연할 동료를 찾더라도, 정말로 서로가 금연을 하고 있는지에 대해서는 본인 외에는 아무도 모른다. 상대방을 속일 수 있다는 점에서 금연과 같은 목표 실천은 쉽게 외적인 동기를 잃기 쉽고, 실패 가능성이 크다.

## 1.2 문제 해결을 위한 솔루션

앞서 정의한 문제를 해결하기 위해, 모두의 달성 플랫폼은 아래와 같은 솔루션을 제시한다. 우선, 모두의 달성 플랫폼은 꾸준한 실천이 필요한 목표를 달성하고자 하는 사람들에게 강력한 외적 동기를 유발한다. 서로의 목표 달성 의지를 지속적으로 환기해줄 타인을 플랫폼을 통해 모으고, 서로의 목표 실천을 지속적으로 확인하도록 해준다. 이를 통해 혼자서 목표달성을 시도할 때 쉽게 실패하는 문제를 해결 가능하며, 목표달성을 함께할 주변인의 부재 역시 해결할 수 있다. 그리고 타인의 계획 실천 여부를 짧은 영상으로 판단할 수 있도록 플랫폼 사용 규칙을 정의해 따르도록 한다. 이로 인해 타인이 목표를 실천했는지 확인하기가 어렵다는 점 역시 해결할 수 있다.

## 1.3 주 타겟 및 STAKE HOLDER

### 1.3.1 타겟

모두의 달성 플랫폼의 타겟은 특정 목표를 달성하고자 하지만, 혼자서는 계획을 꾸준히 실천하기 어려워하여 외적인 동기를 필요로 하는 사람이다. 특히 여러 사람과 함께 합심해 실천해 나가며 타인으로부터 외적인 동기를 부여 받고자 하는 사람을 의미한다.

### 1.3.2 이해관계자

모두의 달성 플랫폼의 Stakeholder로는 플랫폼 사용자와 각종 목표 달성 컨텐츠에 연관된 기관 혹은 기업이 있다. 예시로 금연 목표 달성 컨텐츠에 연관될 수 있는 기관과 기업으로는 보건복지부와 니코틴검사기 판매 기업 등이 있다.

### 1.3.3 타겟 및 이해관계자가 얻는 가치

#### 1.3.3.1 모두의 달성 플랫폼 사용자

모두의 달성 플랫폼 사용자는 꾸준한 계획 실천 여부를 서로 확인하는 시스템을 통해 서로의 의지를 북돋을 수 있다. 또한, 타인이 꾸준히 계획을 실천하고 있는 모습을 보며 계획 실천에 대한 다짐을 스스로 상기함으로써 내적 동기를 부여 받을 수 있다. 추가로 상벌 제도를 통해 내적인 동기 뿐만 아니라 외적인 동기도 부여 받을 수 있다.

이를 통해 중도에 포기하지 않고 목표 달성을 위한 계획을 지속적으로 반복할 수 있도록 도움을 받음으로써 목표 달성 확률을 높인다.

#### 1.3.3.2 관련 기관 혹은 기업

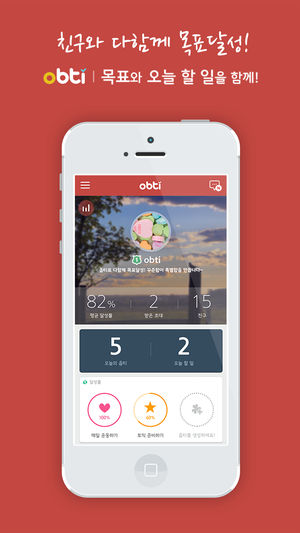
컨텐츠 관여를 통한 사업 혹은 공익의 목적을 달성할 수 있다. 예시로 금연 목표 달성 컨텐츠의 경우, 계획 실천 인증 영상에서 니코틴 측정기를 사용한다. 그래서 니코틴 측정기 판매 기업은 수익을 창출할 수 있다. 또한 보건소와 정부는 모두의 달성 플랫폼을 이용한 금연사업으로 금연 사업의 참여자를 획기적으로 늘리고, 모두의 달성 플랫폼을 통해 금연 달성률을 높여 금연 사업의 목적을 달성할 수 있다.

# 2. 기존사례분석 및 사업화 가능성

## 2.1 유사제품 및 서비스 분석

아래는 현재 상용화되어 있는 서비스 중, 모두의 달성 서비스와 유사한 서비스를 제공하고 있는 애플리케이션이다. 목표 달성을 도와주는 서비스 애플리케이션 이라는 공통점이 있다.

### 2.1.1 옵티(Obti)

** **

옵티(Obti)는 목표 달성을 도와주는 플랫폼으로서 현재 가장 많은 사용자를 보유하고 있는 서비스 애플리케이션이다.[[3]](#endnote-3) 사용자는 목표를 자유롭게 설정 가능하며, 목표 달성을 위한 구체적인 실천 계획 역시 스스로 정한다. 또한 목표를 언제까지 달성할 지 역시 본인이 직접 정할 수 있고, 목표 실천을 함께할 인원을 자유롭게 모집할 수도 있으며, 특정 회원에게 초대 메시지를 보낼 수도 있다.

[그림 1] 옵티(Obti)

자신의 목표 실천 현황을 사진, 위치정보를 통해 다른 사용자들에게 공개가능하다는 점과 서로에게 응원과 격려의 글을 남길 수 있다는 점에서 어느 정도 실천 의지를 북돋아준다. 하지만 이 서비스는 타인의 실천 여부를 확인해줄 의무가 없기에 서로간의 교류가 원활하게 이루어지지 않고, 동기부여 효과가 미미하다. 또한 설정한 목표를 이루지 않아도 불이익이 없기 때문에 외적 동기가 없어 쉽게 포기할 수 있다는 문제점이 있다.

### 2.1.2 윌뱅크 (Willbank)

****

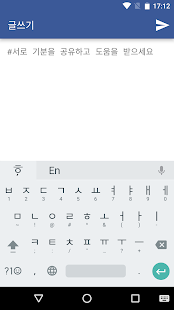
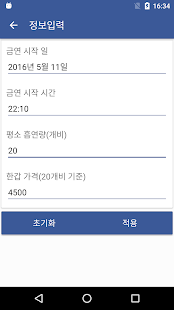
[그림 2] 윌뱅크 (Willbank)

윌뱅크(Willbank)는 사용자가 원하는 목표와 달성 기간을 정하고, 도전 금을 걸어 목표를 이룰 수 있도록 관리감독해주는 서비스 플랫폼이다.[[4]](#endnote-4) 자신이 이루고자 하는 목표와 기간을 설정하고 도전 금을 예치하면 도전이 시작된다. 사용자는 도전 기간 동안 목표를 수행하는 모습을 매일 인증사진으로 찍어 올린다. 사용자 서로의 도전 내용과 인증사진이 공개되어 있어 다른 사용자들이 확인할 수도 있고, 다른 사용자들의 실천 과정을 확인할 수도 있다.

윌뱅크는 사용자가 목표를 수행해야 할 기간 동안 도전 금을 보관한다. 그리고 사용자가 도전 기간 동안 게시해야 할 인증사진 숫자의 80% 이상을 게시하면 도전 금을 전액 돌려준다. 도전 기간 동안 게시한 인증 사진의 숫자가 80% 미만이더라도 성공할 때까지 재도전을 해서 전 금액을 돌려받을 수도 있다. 사용자는 목표 포기 역시 가능한데, 포기 시 수수료를 제외한 나머지를 반환 받을 수 있다.

도전 금과 인증사진 게시를 통한 동기부여 요소가 있지만, 외부의 개입 없이 개인의 의지만으로 계획을 실천해야 한다. 목표 달성에 실패하더라도 다시 돈을 돌려받을 목적으로 인증사진을 조작할 수 있을 우려가 있고, 재도전이 가능하다는 점에서 강력한 도전의식이 사라져 쉽게 나태해질 수 있다. 그리고 관리감독을 해주지만 실제로 도전자의 인증사진을 감독하기엔 감독 인원수 부족이나 판단 근거가 객관적이지 못하는 문제점이 있다.

### 2.1.3 금연 시작

** **

[그림 3] 금연 시작

금연 시작은 사용자가 금연을 하는 동안 얻은 금전적 비용과 시간을 보여줌으로써 금연을 장려하는 금연 서비스 애플리케이션이다.[[5]](#endnote-5) 사용자는 자신의 금연 시작일과 평소 흡연량 정보를 입력하여 금연 기간 동안 늘어난 수명 상태 정보를 제공받는다. 이러한 건강 상태 정보는 지속적으로 업데이트 되기 때문에 사용자는 주기적으로 정보를 제공받을 수 있다. 금연 시작에서 제공하는 커뮤니티 기능은 금연자들이 소통할 수 있는 장을 열어준다. 사용자들은 이를 통해 서로 금연에 관련한 정보, 팁을 주고 받으며 생각을 공유할 수 있다. 하지만 해당 애플리케이션은 금연 여부를 직접 입력해야 하는 점에서 실제로 사용자가 금연을 했는지 정확히 알 수 없다. 또한 강력한 금연 의지가 없는 사용자의 경우 동기부여의 부재로 금연에 성공하기 어렵다.

## 2.2 유사제품과의 차별성

### 2.2.1 사용자 경험에서의 차별

유사제품들의 공통적인 문제점은 동기부여의 부족과 사용자들 간의 교류의 부재이다. 옵티의 경우에는 목표를 달성하지 않아도 불이익이 없어 계획을 꾸준히 실천해나갈 동기를 얻을 수 없다. 또한 타인의 실천 여부를 확인해 줄 의무가 없기 때문에 사용자들 간 교류의 필요성이 없어지고, 사실상 목표 실천 현황 공개의 의미가 없게 되어 개인 다이어리의 형태로 전락하였다.

윌뱅크의 경우에는 도전 금과 인증 사진 게시를 통한 동기부여 요소가 있지만, 개인의 의지만으로 계획을 실천해야 하며 실패 시 재도전이 가능하다는 점에서 동기부여가 경감된다. 또한 인증 사진을 조작할 가능성도 있어 계획 실천 여부 확인이 어렵다는 문제점을 해결하지 못하는 모습을 보인다.

금연 시작의 경우 역시 사용자가 금연을 하는 동안 얻은 금전적 비용과 늘어난 수명을 보여줌으로써 금연을 하도록 동기부여하지만, 외적인 동기부여가 부족하다. 직접 금연 여부를 입력한다는 점도 그 진위를 알 수 없다.

모두의 달성 플랫폼에서는 이러한 외적 동기부여 요소의 부재와 사용자들 간 교류 부재를 해결한 사용자 경험을 제공한다. 같은 목표를 가진 여러 명의 사람들과 함께 목표 달성을 진행한다. 특정 시간마다 본인의 계획 실천 인증이 의무화 되며, 타인의 계획 실천 인증을 확인해주는 일을 의무적으로 수행하여야 한다. 계획을 꾸준히 실천해 최종 목표를 달성할 경우 보상이 주어진다는 점, 실패 시 패널티가 주어진다는 점, 그리고 나의 계획 실천 모습을 항상 타인이 보고있다는 점에서 확실한 외적 동기부여를 제공한다.

또한 인증 영상의 규칙을 서비스 제공자 측에서 정의하고 타인들이 인증 영상을 검토함으로써 조작의 우려와 계획 실천 여부 확인의 어려움을 해결한다. 이처럼 모두의 달성 플랫폼은 같은 목표를 가진 사람들이 서로를 감시하고, 계획을 잘 실천하고 있는 지 상호인증하는 차별적인 경험을 제공하여 목표달성을 돕는다.

### 2.2.2 기술적 차별성

모두의 달성 플랫폼은 앞서 설명한 솔루션을 위해 동영상 스트리밍을 통한 인증을 수행한다. 스트리밍은 큰 용량의 오디오나 동영상을 압축 후 분할하여 전송하기 때문에 하드디스크에 저장하지 않고 다운로드와 동시에 재생하는 것이 가능하다.[[6]](#endnote-6) 이는 사용자의 하드디스크 용량에도 영향을 미치지 않는다. 이와 같은 동영상 인증 서비스를 구현함으로써 사용자의 인증 조작 위험을 방지하고 외적 동기를 강하게 부여하여 타 서비스들과의 차별성을 가진다.

### 2.2.3 비즈니스 모델에서의 차별성

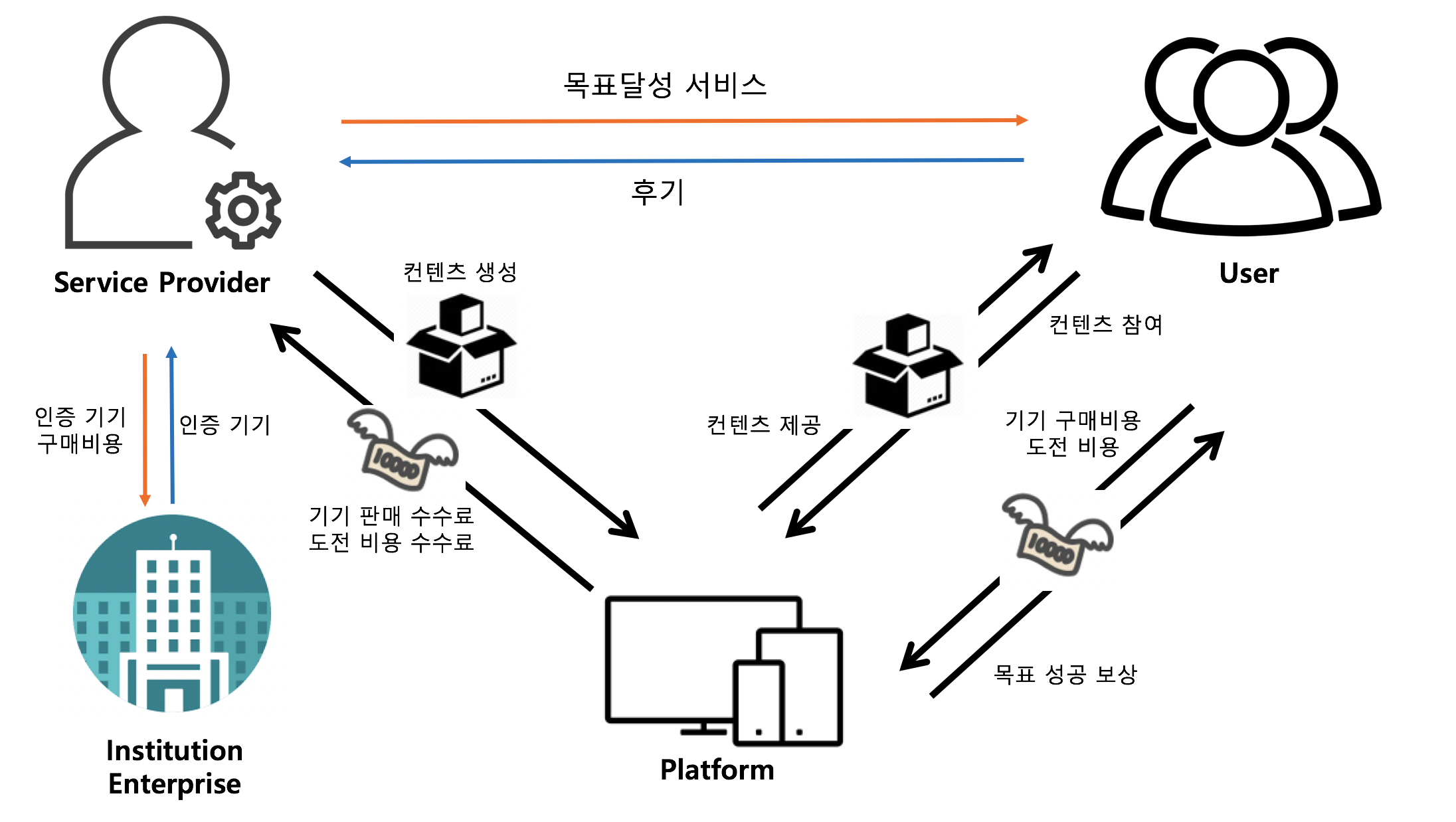
앞서 언급한 유사 서비스의 경우, 서비스 제공자와 사용자 사이의 수익 모델이 매우 빈약하다. 옵티와 금연 시작은 완전히 배너광고에 의존한 수익 모델을 가지고 있다. 윌뱅크의 경우 목표 달성을 위해 도전 금을 예치하고, 실패 시 이로부터 수수료를 얻는 수익 모델이 있지만 끝없는 재도전 허용으로 인해 수익을 기대하기 어렵다. 또한 도전자들은 예치금을 자유롭게 설정할 수 있는데, 도전적인 금액을 예치하는 경우가 굉장히 적기 때문에 수수료를 통한 수익을 기대할 수 없다.

반면 모두의 달성은 기업 및 기관이 수익모델에 연관되어 수익의 원천이 된다. 예시로 금연 목표 달성의 경우 니코틴 측정기 판매 기업이 연관되며, 컨텐츠 참여에 니코틴 측정기가 필수적이다. 모두의 달성 서비스 제공 측은 사용자들이 기업으로부터 니코틴 측정기를 구매할 수 있도록 중개해줌으로써 중개 수수료를 통한 수익을 얻을 수 있다. 모두의 달성 내 각종 컨텐츠의 기획은 이와 같이 기업, 모두의 달성 서비스 제공측, 사용자로 이루어진 비즈니스 모델을 따른다. 이러한 비즈니스 모델은 타 플랫폼과는 다르게 확실한 수입을 보장하고, 지속적인 서비스를 가능하게 하는 원천이 된다.

## 2.3 사업화 가능성

### 2.3.1 비즈니스 모델

[그림 4]는 모두의 달성의 비즈니스 모델을 시각화한 그림이다.



[그림 4] 비즈니스 모델

* **기업, 기관**
  + 서비스 제공자에게 인증에 필요한 기기를 판매하여 이윤을 창출한다.
* **서비스 제공자**
  + 플랫폼을 통해 사용자에게 컨텐츠를 생성하여, 사용자에게 컨텐츠를 제공한다.
  + 사용자가 컨텐츠에 참여하는 과정에서 발생하는 기기 구매비용 및 도전 비용의 수수료를 통해서 이윤을 창출한다.
* **사용자**
  + 제공된 컨텐츠에 참여하여 본인의 목표를 달성할 수 있고, 목표 성공에 대한 보상을 통해 이윤을 창출한다.

### 2.3.2 타겟 시장 분석

### 

### 

[그림 5] 윌뱅크 (Willbank)

[그림 6] 금연 시작

[그림 7] 옵티 (Obti)

### 

### 위 세 플랫폼은 현재 구글 플레이 스토어에 등록되어있는 모두의 달성 플랫폼과 비슷한 제품인 금연 시작, 옵티 그리고 윌뱅크이다. 각 플랫폼은 다운로드 수가 약 10만 명, 5만 명 그리고 1천 명 정도 이다. 세 플랫폼의 동향만 살펴보아도 적어도 10만 명 이상의 사용자가 특정한 목표 달성을 위해 목표 달성 플랫폼을 이용한다는 것을 알 수 있다.

목표 달성 플랫폼은 다이어리 기능만 제공하던 예전의 플랫폼에서 타인 간 교류시스템을 통한 동기부여와 목표 의식을 중요하게 여기는 추세로 변화하고 있다. 하지만 현재의 목표 달성 플랫폼은 강한 외적 동기부여를 하지 못하고 조작할 수 없는 사용자 간의 인증시스템까지는 찾아볼 수 없었다. 그렇기 때문에 모두의 달성 플랫폼은 기존 플랫폼이 제공하는 것 외에도 강한 외적 동기부여와 확고한 목표의식을 가질 수 있도록 인증 시스템을 구축할 예정이다. 따라서 현재 시장에서도 경쟁력을 갖출 것으로 예상된다.

마지막으로, 모두의 달성 플랫폼 사용자는 컨텐츠 수행을 위해 앱 내부에서 니코틴 측정기와 같은 인증 수단을 구매한다. 모두의 달성 플랫폼은 이 과정에서 생기는 수수료를 통해 이윤을 창출한다. 이러한 수익 구조는 커져가는 ‘인 앱 구매’ 시장에서 많은 이윤을 얻을 것이라고 분석된다.

### 2.3.3 예상 사업 규모

초창기에는 사용자가 적기 때문에 인건비, 회사 유지비, 애플리케이션 유지 등을 위해 투자회사로부터 엔젤투자 받은 자본금으로 플랫폼을 운영할 것이다. 투자금을 자본으로 서비스를 운영하고 점차 사용자를 늘려나갈 것이다. 앞서 살펴본 세 가지 플랫폼들의 문제점을 보완하여 차별성을 가진 모두의 달성 플랫폼은 기존의 목표달성 플랫폼을 사용하고 있는 사용자들을 유치할 수 있을 것이다. 또한, 새로 목표 달성 플랫폼을 시작하는 사용자까지 합하여 모두의 달성 플랫폼의 최종 목표는 약 10만 명 이상의 사용자를 확보하는 것이다. 사용자가 많아진 시점부터는 도전 비용에 대한 수수료와 컨텐츠 관련 제품 판매 수수료를 통한 수입을 기반으로 사업을 운영할 것이다. 그리고 신규 컨텐츠에 대한 새로운 투자를 통해 사업을 더욱 확장, 지속해나갈 것이다.

스마트 폰 사용자가 늘어남에 따라 스마트 폰 애플리케이션 시장은 점차 증가하는 추세이고 앞서 말했듯이 앱 내에서의 구매(인 앱 구매)가 늘어나는 추세이다.[[7]](#endnote-7) 즉, 애플리케이션을 사용하는 사람과 애플리케이션 내에서 사용하는 비용이 점점 늘고 있으므로 모두의 달성 플랫폼은 많은 사용자를 기반으로 충분한 재정을 확보할 것으로 기대된다.

# 3. USER STORIES

## 3.1 주요 시나리오

### 3.1.1 사용자 관점의 주요 시나리오



1. 모두의 달성 회원가입 시나리오
   1. 모두의 달성 애플리케이션에 접속한다.
   2. Sign Up버튼을 누른다.
   3. 인적사항을 작성하여 회원가입을 한다.

[그림 8] 회원가입 화면

1. 컨텐츠 정보 확인 시나리오
   1. 모두의 달성 애플리케이션에 접속한다.
   2. E-mail과 Password를 입력해 로그인한다.
   3. 돋보기 아이콘을 클릭한다.
   4. 정보를 알고자 하는 컨텐츠를 선택한다.
   5. 해당 컨텐츠의 정보를 확인한다.

[그림 9] 찾기 화면

1. 컨텐츠 참가 시나리오



[그림 10] 컨텐츠 참여 화면

[그림 11] 컨텐츠 참여 화면 (기간 선택)

* 1. 모두의 달성 애플리케이션에 접속한다.
  2. E-mail과 Password를 통해 로그인한다.
  3. 참가하고 싶은 컨텐츠를 선택한다.
  4. 컨텐츠 참여 기간을 선택한 후 참가한다.

1. 본인 컨텐츠 참여 현황 확인 시나리오



[그림 12] 금연 컨텐츠 참여 현황 시나리오

* 1. 모두의 달성 애플리케이션에 접속한다.
  2. E-mail과 Password를 통해 로그인한다.
  3. 참여 중인 컨텐츠와 본인의 과거 인증 영상을 확인한다.
  4. 자세한 정보를 원하는 컨텐츠를 선택한다.
  5. 선택한 컨텐츠의 정보(달력, 남은 인증 시간. 컨텐츠 설명, 공지사항 등)를 확인한다.

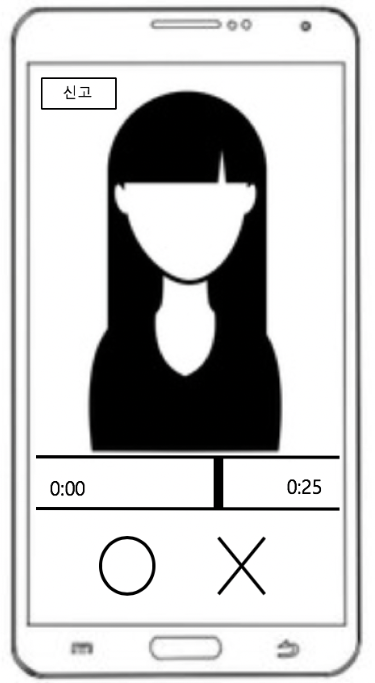
1. 본인 컨텐츠 인증 시나리오



[그림 13] 본인 인증 시나리오

* 1. 본인의 인증 시간이 되면 푸시 알람을 통해 확인한다.
  2. 푸시 알람을 선택하거나 앱을 통해 인증을 시작한다.
  3. 카메라를 통해 인증 동영상을 촬영한다.
  4. 재촬영을 원할 시 재촬영을 한다.
  5. 영상 촬영이 완료되면 전송 버튼을 눌러 모두의 달성 애플리케이션에 업로드한다.

1. 타인 컨텐츠 인증 체크 시나리오



[그림 14] 타인 인증 체크 화면

* 1. 모두의 달성 애플리케이션에 접속한다.
  2. E-mail과 Password를 통해 로그인한다.
  3. 참여 중인 컨텐츠를 선택한다.
  4. 선택한 컨텐츠의 타인 인증 대기 영상을 선택한다.
  5. 영상이 스트리밍되는 것을 시청한다.
  6. 인증 영상을 확인 후 인증 성공 여부를 선택한다.
  7. 이상한 영상을 올린 경우, 신고 버튼을 통해 신고한다.

### 3.1.2 관리자 관점의 주요 시나리오

1. 컨텐츠 생성 시나리오
   1. 관리자는 웹 페이지에 로그인하여 관리자 기능에 접근한다.
   2. 컨텐츠 생성 버튼을 클릭한다.
   3. 컨텐츠의 주제 및 인증 방식, 인원 수 등을 결정한다.
   4. 컨텐츠를 생성한다.
2. 컨텐츠 그룹 현황 확인 시나리오
   1. 관리자는 웹 페이지에 로그인하여 관리자 기능에 접근한다.
   2. 확인하고 싶은 컨텐츠의 그룹을 선택한다.
   3. 그룹의 현재 달성률, 잔여 인원 등 현재 진행 상황을 확인한다.
3. 신고 접수 확인 및 제재 처리 시나리오
   1. 관리자는 웹 페이지에 로그인하여 관리자 기능에 접근한다.
   2. 신고 접수 페이지로 이동하여 악성 유저 신고 접수를 확인한다.
   3. 신고가 반복되는 특정 악성 유저의 이용을 중지시킨다.

## 3.2 User Stories

[표 1] User story

|  |  |
| --- | --- |
| **Story ID** | **User Story** |
| **1** | 유저는 애플리케이션에 대한 Overview 및 Guide를 읽는다. |
| **2** | 유저는 회원가입을 함으로써 모두의 달성 사용권한을 획득한다. |
| **3** | 유저는 비밀번호 찾기 기능을 통해 잊어버린 비밀번호를 재설정할 수 있다. |
| **4** | 유저는 로그인한다. |
| **5** | 유저는 페이스북, 구글 로그인 지원을 받는다. |
| **6** | 유저는 자신이 속한 컨텐츠를 확인할 수 있다. |
| **7** | 유저는 공지사항을 확인한다. |
| **8** | 유저는 자신의 정보를 확인한다. |
| **9** | 유저는 자신의 계정을 관리한다. |
| **10** | 유저는 찾기 기능을 통해 원하는 컨텐츠를 찾을 수 있다. |
| **11** | 유저는 찾기 기능을 통해 다른 유저를 찾을 수 있다. |
| **12** | 유저는 각 컨텐츠의 전체 진행 현황을 확인한다. |
| **13** | 유저는 본인의 목표 달성 현황을 확인한다. (달력/ 그래프) |
| **14** | 유저는 각 컨텐츠의 참여방법을 확인한다. |
| **15** | 유저는 컨텐츠에 참여신청을 한다. |
| **16** | 유저는 컨텐츠 별로 제시된 시작일 중, 원하는 시작일을 택할 수 있다. |
| **17** | 유저는 컨텐츠 별 준비 요건에 대한 결제를 한다. |
| **18** | 유저는 컨텐츠 참여신청을 완료한다. |
| **19** | 유저는 실천여부를 카메라를 통해 인증한다. |
| **20** | 유저는 자신의 영상에 대한 인증확인여부를 확인한다. |
| **21** | 유저는 자신의 영상을 오인증한 익명의 유저들을 신고한다. |
| **22** | 유저는 자신의 인증 영상에 대한 응원메세지를 확인한다. |
| **23** | 유저는 목표 달성에 성공한다. |
| **24** | 유저는 목표 달성 후기 및 설문조사를 한다. |
| **25** | 유저는 보상을 확인한다. |
| **26** | 유저는 타인의 실천여부를 스트리밍을 통해 확인해준다. |
| **27** | 유저는 타인의 영상을 보며 신고를 한다. |
| **28** | 유저는 타 유저들을 아이디를 통해 검색해 개인 보드를 접근할 수 있다. |
| **29** | 유저는 자신 및 타 유저들의 인증 영상 보드를 확인가능하다. |
| **30** | 유저는 타인에게 응원 메시지를 보낸다. |
| **31** | 유저는 목표 달성에 실패한다. |
| **32** | 유저는 자신이 받은 패널티를 확인한다. |
| **33** | 유저는 자신이 왜 실패한거같은 지에 대한 의견을 작성하고, 설문조사를 한다. |
| **34** | 관리자는 로그인하여 관리자 기능에 접근한다. |
| **35** | 관리자는 신고 접수를 받을 수 있다. |
| **36** | 관리자는 인증 영상에 대한 신고 접수를 한다. |
| **37** | 관리자는 유저 신고를 접수한다. (오인증에 대한 신고) |
| **38** | 관리자는 상습적인 No유저를 탐지한다. |
| **39** | 관리자는 유저를 사용 중지시킨다. |
| **40** | 관리자는 각 컨텐츠 그룹의 현황을 조회할 수 있다. |
| **41** | 관리자는 유저를 조회할 수 있다. |
| **42** | 관리자는 유저들의 설문조사 및 피드백을 확인한다. |
| **43** | 관리자는 목표달성 성공유저들에 대한 최종 확인을 진행한다. |
| **44** | 관리자는 목표달성 성공유저들에게 보상을 지급하는 것을 confirm한다. |
| **45** | 관리자는 목표달성에 실패한 유저들을 확인한다. |
| **46** | 관리자는 목표달성 실패유저들에 대한 최종 확인을 진행한다. |
| **47** | 관리자는 컨텐츠를 관리할 수 있다. |
| **48** | 관리자는 새로 기획된 컨텐츠를 등록한다. (준비 요건, 상품, 금액 결제 기능 등등) |
| **49** | 관리자는 폐쇠할 컨텐츠를 등록해제한다. |

## 3.3 Product Backlog

[표 2] Product Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **User Story** | **Priority** | **Story Point** |
| **1** | 유저는 애플리케이션에 대한 Overview 및 Guide를 읽는다. | 50 | 1.0 |
| **2** | 유저는 회원가입을 함으로써 모두의 달성 사용권한을 획득한다. | 150 | 3.0 |
| **3** | 유저는 비밀번호 찾기 기능을 통해 잊어버린 비밀번호를 재설정할 수 있다. | 70 | 5.0 |
| **4** | 유저는 로그인한다. | 150 | 5.0 |
| **5** | 유저는 페이스북, 구글 로그인 지원을 받는다. | 50 | 2.0 |
| **6** | 유저는 자신이 속한 컨텐츠를 확인할 수 있다. | 90 | 3.0 |
| **7** | 유저는 공지사항을 확인한다. | 100 | 1.5 |
| **8** | 유저는 자신의 정보를 확인한다. | 120 | 2.0 |
| **9** | 유저는 자신의 계정을 관리한다. | 120 | 2.0 |
| **10** | 유저는 찾기 기능을 통해 원하는 컨텐츠를 찾을 수 있다. | 120 | 5.0 |
| **11** | 유저는 찾기 기능을 통해 다른 유저를 찾을 수 있다. | 100 | 4.0 |
| **12** | 유저는 각 컨텐츠의 전체 진행 현황을 확인한다. | 90 | 3.0 |
| **13** | 유저는 본인의 목표 달성 현황을 확인한다. (달력/ 그래프) | 90 | 4.0 |
| **14** | 유저는 각 컨텐츠의 참여방법을 확인한다. | 130 | 2.0 |
| **15** | 유저는 컨텐츠에 참여신청을 한다. | 130 | 2.0 |
| **16** | 유저는 컨텐츠 별로 제시된 시작일 중, 원하는 시작일을 택할 수 있다. | 110 | 1.5 |
| **17** | 유저는 컨텐츠 별 준비 요건에 대한 결제를 한다. | 90 | 3.0 |
| **18** | 유저는 컨텐츠 참여신청을 완료한다. | 130 | 1.0 |
| **19** | 유저는 실천여부를 카메라를 통해 인증한다. | 150 | 6.0 |
| **20** | 유저는 자신의 영상에 대한 인증확인여부를 확인한다. | 150 | 2.0 |
| **21** | 유저는 자신의 영상을 오인증한 익명의 유저들을 신고한다. | 100 | 2.0 |
| **22** | 유저는 자신의 인증 영상에 대한 응원메세지를 확인한다. | 50 | 1.0 |
| **23** | 유저는 목표 달성에 성공한다. | 100 | 2.0 |
| **24** | 유저는 목표 달성 후기 및 설문조사를 한다. | 50 | 1.0 |
| **25** | 유저는 보상을 확인한다. | 100 | 2.0 |
| **26** | 유저는 타인의 실천여부를 스트리밍을 통해 확인해준다. | 150 | 6.0 |
| **27** | 유저는 타인의 영상을 보며 신고를 한다. | 70 | 1.5 |
| **28** | 유저는 타 유저들을 아이디를 통해 검색해 개인 보드를 접근할 수 있다. | 80 | 1.5 |
| **29** | 유저는 자신 및 타 유저들의 인증 영상 보드를 확인가능하다. | 80 | 2.0 |
| **30** | 유저는 타인에게 응원 메시지를 보낸다. | 50 | 1.0 |
| **31** | 유저는 목표 달성에 실패한다. | 100 | 2.0 |
| **32** | 유저는 자신이 받은 패널티를 확인한다. | 100 | 1.5 |
| **33** | 유저는 자신이 왜 실패한거같은 지에 대한 의견을 작성하고, 설문조사를 한다. | 50 | 1.0 |
| **34** | 관리자는 로그인하여 관리자 기능에 접근한다. | 150 | 4.0 |
| **35** | 관리자는 신고 접수를 받을 수 있다. | 90 | 1.5 |
| **36** | 관리자는 인증 영상에 대한 신고 접수를 한다. | 90 | 1.5 |
| **37** | 관리자는 유저 신고를 접수한다. (오인증에 대한 신고) | 90 | 1.5 |
| **38** | 관리자는 상습적인 No유저를 탐지한다. | 90 | 1.0 |
| **39** | 관리자는 유저를 사용 중지시킨다. | 90 | 1.0 |
| **40** | 관리자는 각 컨텐츠 그룹의 현황을 조회할 수 있다. | 100 | 2.5 |
| **41** | 관리자는 유저를 조회할 수 있다. | 70 | 1.5 |
| **42** | 관리자는 유저들의 설문조사 및 피드백을 확인한다. | 70 | 1.5 |
| **43** | 관리자는 목표달성 성공유저들에 대한 최종 확인을 진행한다. | 120 | 2.0 |
| **44** | 관리자는 목표달성 성공유저들에게 보상을 지급하는 것을 confirm한다. | 120 | 2.0 |
| **45** | 관리자는 목표달성에 실패한 유저들을 확인한다. | 120 | 2.0 |
| **46** | 관리자는 목표달성 실패유저들에 대한 최종 확인을 진행한다. | 120 | 2.0 |
| **47** | 관리자는 컨텐츠를 관리할 수 있다. | 150 | 5.0 |
| **48** | 관리자는 새로 기획된 컨텐츠를 등록한다. (준비 요건 상품 혹은 금액 결제 기능 등등) | 150 | 5.0 |
| **49** | 관리자는 폐쇠할 컨텐츠를 등록해제한다. | 100 | 2.0 |

[표 2]의 User Story별 Priority와 Story point를 바탕으로 매 Sprint 시에 Sprint backlog 설정과 Task로 세분화할 예정이다.

[표 3]은 현재 진행 중인 Iteration 2의 Task로 세분화된 User Story이다.

[표 3] Sprint Backlog (Iteration2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Story**  **ID** | **User Story** | **Prior**  **-ity** | **Task ID** | **Title** | **Story Point** |
| 1 | 유저는 애플리케이션에 대한 Overview 및 Guide를 읽는다. | 50 | 1.1 | 애플리케이션 설명 UI를 구현한다. (Front) | 0.5 |
| 1 | 유저는 애플리케이션에 대한 Overview 및 Guide를 읽는다. | 50 | 1.2 | 애플리케이션 설명 기능을 구현한다. (Front) | 0.5 |
| 2 | 유저는 회원가입을 함으로써 모두의 달성 사용권한을 획득한다. | 150 | 2.1 | 회원 가입 UI를 구현한다. (Front) | 1.0 |
| 2 | 유저는 회원가입을 함으로써 모두의 달성 사용권한을 획득한다. | 150 | 2.2 | 회원가입 기능을 구현한다. (Front) | 1.0 |
| 2 | 유저는 회원가입을 함으로써 모두의 달성 사용권한을 획득한다. | 150 | 2.3 | 회원가입 기능을 구현한다. (Back) | 1.0 |
| 4 | 유저는 로그인한다. | 150 | 4.1 | 로그인 UI를 구현한다. (Front) | 1.0 |
| 4 | 유저는 로그인한다. | 150 | 4.2 | 로그인 기능을 구현한다. (Front) | 2.0 |
| 4 | 유저는 로그인한다. | 150 | 4.3 | 로그인 기능을 구현한다. (Back) | 2.0 |
| 8 | 유저는 자신의 정보를 확인한다. | 120 | 8.1 | 사용자 페이지 UI를 구현한다. (Front) | 2.0 |
| 9 | 유저는 자신의 계정을 관리한다. | 120 | 9.1 | 사용자 페이지 기능을 구현한다. (Front) | 1.0 |
| 9 | 유저는 자신의 계정을 관리한다. | 120 | 9.2 | 사용자 계정 관리 기능을 구현한다. (Back) | 1.0 |
| 10 | 유저는 찾기 기능을 통해 원하는 컨텐츠를 찾을 수 있다. | 120 | 10.1 | 찾기 기능 UI를 구현한다. (Front) | 1.0 |
| 10 | 유저는 찾기 기능을 통해 원하는 컨텐츠를 찾을 수 있다. | 120 | 10.2 | 찾기 기능을 구현한다. (Front) | 2.0 |
| 10 | 유저는 찾기 기능을 통해 원하는 컨텐츠를 찾을 수 있다. | 120 | 10.3 | 찾기 기능을 구현한다. (Back) | 2.0 |
| 14 | 유저는 각 컨텐츠의 참여방법을 확인한다. | 130 | 14.1 | 참여방법을 보여주는 UI를 구현한다. (Front) | 1.0 |
| 14 | 유저는 각 컨텐츠의 참여방법을 확인한다. | 130 | 14.2 | 참여방법을 보여주는 기능을 구현한다. (Front) | 1.0 |
| 15 | 유저는 컨텐츠에 참여신청을 한다. | 130 | 15.1 | 참여 신청 UI를 구현한다(Front) | 0.5 |
| 15 | 유저는 컨텐츠에 참여신청을 한다. | 130 | 15.2 | 참여 신청 기능을 구현한다. (Front) | 0.5 |
| 15 | 유저는 컨텐츠에 참여신청을 한다. | 130 | 15.3 | 참여 신청 기능을 구현한다. (Back) | 1.0 |
| 18 | 유저는 컨텐츠 참여신청을 완료한다. | 130 | 18.1 | 참여 완료 기능을 구현한다(Front) | 0.5 |
| 18 | 유저는 컨텐츠 참여신청을 완료한다. | 130 | 18.2 | 참여 신청 기능을 구현한다. (Back) | 0.5 |
| 19 | 유저는 실천여부를 카메라를 통해 인증한다. | 150 | 19.1 | 인증 푸시 알람을 구현한다. | 3.0 |
| 19 | 유저는 실천여부를 카메라를 통해 인증한다. | 150 | 19.2 | 카메라 인증 기능을 구현한다. | 2.0 |
| 19 | 유저는 실천여부를 카메라를 통해 인증한다. | 150 | 19.3 | 비디오를 서버로 업로드한다. | 1.0 |
| 20 | 유저는 자신의 영상에 대한 인증확인여부를 확인한다. | 150 | 20.1 | 내 인증 영상 UI를 구현한다. (Front) | 2.0 |

# 4. 구현

## 4.1 필요 기술

### 4.1.1 기술 스택 / 프레임 워크

#### 4.1.1.1 MEAN Stack



[그림 15] MEAN Stack

MEAN Stack은 MongoDB, Express, AngularJS, Node.js의 약자로 웹 개발의 Front 부터 Back을 아우르는 기술 스택이다.[[8]](#endnote-8) MEAN Stack을 사용하여 본 프로젝트의 서버는 Node.js와 Express, 데이터 베이스는 MongoDB를 사용하여 구현할 예정이다. 또한 관리자 페이지의 Front는 AngularJS (본 프로젝트에서는 Angular2를 사용할 예정)를 사용하여 구현할 예정이다.

본 프로젝트에서 MEAN Stack을 채택한 이유는 MEAN Stack의 장점이 본 프로젝트 진행에 많은 도움이 된다고 생각해서다. MEAN Stack의 장점은 다음과 같다.

* MEAN Stack의 가장 큰 장점은 4가지 기술 모두 Javascript를 뿌리로 한다는 점이다.[[9]](#endnote-9) 덕분에 Javascript 한가지 언어로 빠른 개발이 가능하다.

* MEAN Stack은 MVC 디자인 패턴을 강요한다.[[10]](#endnote-10) MVC 디자인 패턴은 애플리케이션을 Model, View, Controller 3 파트로 구분하여 각 컴포넌트 간의 Coupling을 낮춘다. 이는 4명이서 진행하는 모두의 달성 프로젝트에서 역할 분담을 쉽게 해준다는 장점이 있다. 또한 각 컴포넌트 간의 Coupling 낮기 때문에 지속적으로 요구사항이 바뀌는 Agile 개발프로세스에서 다른 컴포넌트에 영향을 주지 않고 필요 부분만 수정하면 된다는 장점이 있다.

* MEAN Stack은 AngularJS를 통해 SPA (Single Page Application) 구현이 가능하다.[[11]](#endnote-11) SPA는 하나의 페이지로 구현되어 사용자 요청에 필요한 컨텐츠(화면 조각)만 바꾸어 웹의 렌더링 속도를 높이고, 트래픽을 낮춘다. 이러한 장점을 통해 깔끔한 사용자 인터페이스(UI)를 구현할 수 있고, 사용자 경험(UX)을 향상시킬 수 있다. 비록 AngularJS 는 관리자 페이지 제작에만 관여되지만 관리자 또한 서비스의 stackeholder이므로 SPA로 구현하기로 하였다.

#### 4.1.1.2 Android



[그림 16] Android

본 프로젝트의 최종 목표는 모두의 달성 모바일 앱을 네이티브 앱 형태로 배포하는 것이다. 때문에 앱 개발 이전에 대표적으로 사용되는 두개의 모바일 환경인 안드로이드와 IOS 중 하나를 선택해야한다. 본 4조는 두개의 모바일 환경인 안드로이드와 IOS를 비교하며 분석하였다.

* 우선 모두의 달성의 타겟을 분석해보았다. 모두의 달성 플랫폼은 여러 컨텐츠를 제공하는데, 금연 컨텐츠의 니코틴 측정기와 같이 대부분의 컨텐츠가 인증 수단을 필요로 한다. 때문에 사용자들은 컨텐츠를 이용하기 전에 인증 수단을 구매하는 절차를 거친다. 본 프로젝트에서는 이러한 구매 절차의 편리성을 위해 인증 수단에 관련된 모든 이해관계자를 국내로 한정 지었다. 모든 이해관계자를 국내로 한정 지었기 때문에 우선 국내에서 가장 큰 비중을 차지하는 모바일 운영체제를 조사했다. 그 결과 국내 스마트폰 os 점유율은 안드로이드가 76.7%로 IOS의 23.05% 보다 높은 것으로 나왔다.[[12]](#endnote-12) (17년 기준) 국내 스마트폰 사용자가 대부분 안드로이드을 사용하므로 본 프로젝트에서는 안드로이드 앱 개발을 선택하였다.
* 다음은 개발 언어를 살펴보았다. 안드로이드의 경우 주요 개발 언어로 Java와 Kotlin이 있다. Java는 현재 개발자들이 가장 많이 사용하는 언어이고, Kotlin은 Java의 여러 단점을 보완하며 안드로이드 개발에 새롭게 사용되는 언어다.[[13]](#endnote-13) 본 4조는 Java 경험이 IOS 개발에 필요한 Objective-C와 Swift 보다 더 많고, Java와 유사하며 Java의 여러 문제를 보완한 Kotlin 을 습득하는 것이 IOS에 사용되는 언어보다 빠를 것이라 판단하였다. 이런 점을 고려하여 다섯번의 Iteration 내에 프로젝트를 완성하기 위해 Java와 Kotlin를 사용하는 안드로이드 모바일 환경을 선택하였다.

### 4.1.2 언어 / 개발 환경

#### 4.1.2.1 Kotlin



[그림 17] Kotlin

Kotlin은 IntelliJ IDEA의 JetBrains에서 개발한 안드로이드 개발 언어다. Kotlin은 기존의 안드로이드 개발 언어로 자주 사용되던 Java의 여러 문제점을 해결해 나가며 현재 안드로이드 공식 언어로 채택되었다. 본 프로젝트에서 Java 대신 Kotlin을 사용한 이유는 다음과 같다.

* Kotlin은 Null Pointer Exception에서 안전하다. 기본적으로 Kotlin은 null 값을 허용하지 않기 때문에 앱을 실행하기 전에 null값을 안전하게 처리할 수 있다.
* Kotlin은 간결하다. Kotlin은 java보다 짧게 코드를 작성할 수 있고, 이를 통해 가독성을 높일 수 있다.
* Kotlin 언어와 Android Studio는 같은 JetBrains 사에서 만들어져 Android Studio 3.0에서 Kotlin을 완벽하게 지원한다. 때문에 본 프로젝트에서 사용할 Android Studio를 통해 Kotlin을 편하게 사용할 수 있다.

#### 4.1.2.2 HTML, CSS, Javascript



[그림 18] HTML, Javascript, CSS

본 프로젝트의 서버를 담당할 MongoDB, Node.js, Express와 관리자 페이지의 Front를 담당할 AngularJS는 모두 Javascript를 사용한다. (AngularJS에서 사용되는 Typescript는 Javascript ECMA 표준을 따르는 확장 언어다.) Front 부터 Back까지 모든 동작을 관여하는데 Javascript를 사용한다.

관리자 페이지의 경우 AngularJS를 통해 구현할 예정이다. 이 때 관리자 페이지의 View를 구현하기 위해 HTML과 CSS를 사용한다. HTML (Hyper Text Markup Language)은 View의 뼈대를 작성하기 위한 문서이고, CSS (Cascading Style Sheets)는 HTML 문서의 스타일을 꾸밀 때 사용되는 언어다.

#### 4.1.2.3 개발 환경



[그림 19] WebStorm

[그림 20] Android Studio

프로젝트를 진행하는데 필요 기술 조사를 모두 마친 후 어떤 IDE (Integrated Development Environment) 를 사용하여 개발할지 조사했다. IDE는 코딩, 디버그, 컴파일, 배포 등 개발에 관련된 모든 작업을 처리하는 환경을 제공하는 중요한 소프트웨어로 개발에 있어 신중해야 할 부분 중 하나다.[[14]](#endnote-14) 본 4조는 여러 IDE를 조사하며 프로젝트 진행에 적합한 IDE를 채택하였다.

우선 모두의 달성 모바일 앱을 개발하기 위한 안드로이드 개발은 Android Studio를 채택하였다. 그 이유는 다음과 같다.

* Android Studio는 그래픽 레이아웃 편집기 기능이 강력하다.[[15]](#endnote-15) 사용자는 모바일 앱을 사용자 인터페이스를 통해 제일 먼저 접하게된다. 때문에 사용자 인터페이스는 앱의 첫 인상이며 서비스의 성공을 좌우를 결정하는 중요한 요소가된다. 본 4조는 이러한 점을 고려하여 Android Studio의 강력한 그래픽 레이아웃 편집기를 사용하여 단기간 내에 쉽고 질 높은 사용자 인터페이스를 구성하기로 하였다.
* Android Studio는 간편한 Test 기능을 제공한다.[[16]](#endnote-16) 개발에 있어 TDD (Test Driven Development)는 중요한 요소 중 하나다. TDD는 소스 코드의 특정 모듈이 의도된 방향으로 정확히 작동하는지 검증하는 절차다. 각각의 모듈을 검증함으로써 전체 프로그램의 안정성을 높일 수 있고, 디버깅 시간을 단축하여 개발시간을 줄일 수 있다. Android Studio는 이러한 TDD 작업을 간단히 수행할 수 있도록 설계되어있다. 본 4조는 모두의 달성 앱의 모듈들을 각각 Unit 테스트할 계획이며 짧은 기간 동안 모두 진행할 수 있도록 기능을 제공하는 Android Studio를 선택하였다.

다음으로 관리자 사이트와 서버를 개발 (MEAN Stack)은 Webstorm을 사용하여 개발하기로 하였다. 그 이유는 다음과 같다.

* Webstorm은 최신 기술과의 호환성이 높다.[[17]](#endnote-17) 본 프로젝트에서 개발할 관리자 페이지의 Front는 AngularJS를 통해 개발할 예정이다. Webstorm은 AngularJS와 같은 최신 기술과의 호환성이 높아 개발에 매우 편리하다.
* Javascript와 Node.js를 위한 강력한 디버거를 제공한다. 본 프로젝트의 서버는 Node.js 플랫폼을 통해 개발된다. 또 전체 기술 스택에서 Javascript가 사용된다. 이러한 점에서 강력한 디버거를 제공하는 Webstorm을 선택하였다.

### 4.1.3 외부 기술

#### 4.1.3.1 Firebase



[그림 21] Firebase Cloud Messaging

[그림 22] Firebase

모두의 달성은 컨텐츠의 목표를 달성하기 위해 컨텐츠에 맞게 자신을 인증을 하고, 또 타인의 인증을 체크해주는 플랫폼이다. 자신을 인증하는 경우 지정된 시간 내에 인증하지 않으면 패널티가 존재하기 때문에 모바일 앱은 사용자에게 푸시 알람을 제공한다. 또 타인의 인증 영상을 체크해주는 경우도 푸시 알람을 제공하여 빠른 체크를 가능하게 한다. 이러한 푸시 알람 기능은 서버에서 클라이언트로 요청하는 기술로 본 프로젝트에서는 Firebase에서 제공하는 FCM(Firebase Cloud Messaging)기술을 사용하기로 하였다.[[18]](#endnote-18)

Firebase에서 제공하는 FCM(Firebase Cloud Messaging)은 무료로 안정적으로 메세지를 전송할 수 있는 교차 플랫폼 메세지 솔루션이다. 이를 통해 모두의 달성 사용자 (클라이언트)들에게 인증 시간임을 알리거나 상대방의 인증을 체크해달라는 메세지를 전송할 수 있게 된다. FCM은 안드로이드에서도 지원하여 푸시 알람을 구현하는 적합하다고 판단하였다.

#### 4.1.3.2 Exoplayer

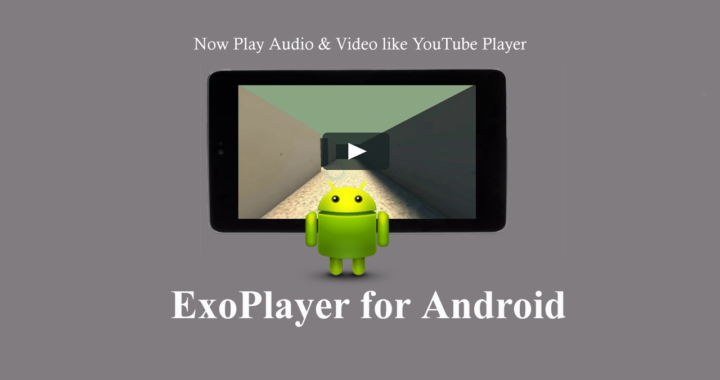
모두의 달성의 핵심 기술로는 스트리밍 기술이 있다. 자신의 영상을 인증하거나 상대방의 인증 영상을 확인할때 서버에 저장된 영상을 스트리밍으로 전송해야한다. 본 4조는 이를 위한 스트리밍 프로토콜을 조사한 다음, Adaptive HTTP Streaming 프로토콜을 사용하기로 결정하였다. 또한 안드로이드에서 위의 프로토콜을 통한 스트리밍을 수신하기 위한 오픈소스인 Exoplayer를 사용하기로 하였다.



[그림 23] HTTP Adaptive Streaming

모두의 달성의 스트리밍 기술은 HTTP Adaptive Streaming 으로 채택하였다.[[19]](#endnote-19) RTMP/RTSP 스트리밍의 Bandwidth 효율성과 비디오 화질의 변경이 용이하다는 장점과 HTTP Progressive Download의 미디어 서버와 같은 비싼 장비를 사용하지 않는 점과 RTMP 프로토콜 같은 복잡한 기술을 사용하지 않는다는 장점을 결합 한 방식이 HTTP Adaptive Streaming이다. 또한, HTTP Adaptive Streaming은 사용자의 네트워크 상태에 따라 Adaptive한 스트리밍이 가능하게 한다. 그리고 동영상 콘텐츠를 다양한 해상도로 인코딩해 저장해주기 때문에 사용자가 동영상을 재생할 때 적절한 해상도를 제공해주고, 네트워크 상태에 따라 적절한 전략으로 컨텐츠의 소스를 선택해 최적의 스트리밍 서비스를 제공한다.

또한 버퍼링을 감수하고라도 고해상도로 영상을 시청하고 싶은 유저들에게 선택권을 제공할 수 있어 유용하다. 본 프로토콜을 제공하기 위한 Manifest 포맷으로는 대표적으로 Apple에서 독자적으로 개발한 HTTP Live Streaming(HLS)과, MPEG가 표준화한 Dynamic Adaptive Streaming over HTTP(DASH) 등이 있다.

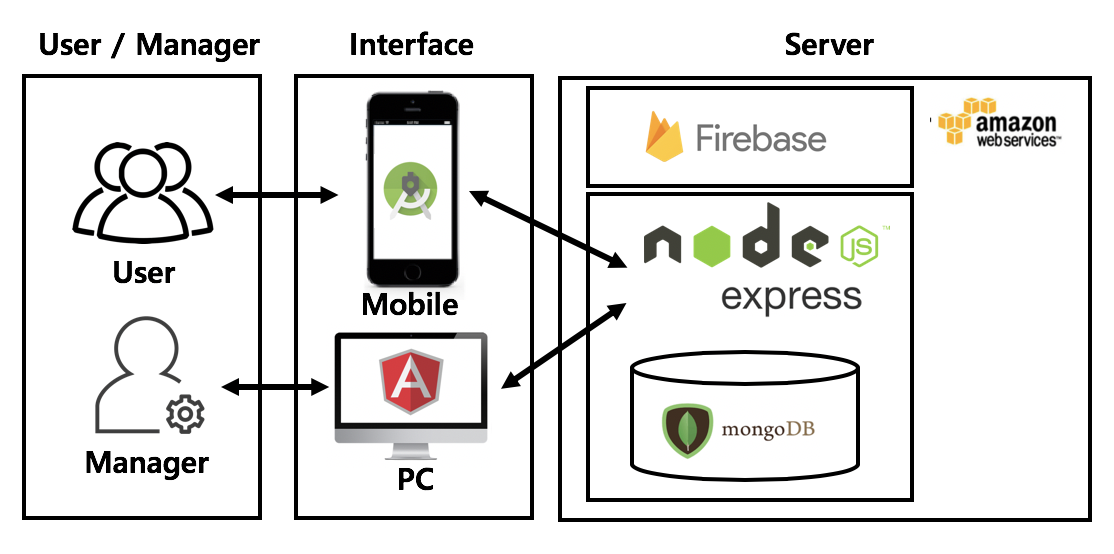


[그림 24] Exoplayer

위 HTTP Adaptive Streaming 기술을 사용하기 위해 모두의 달성에서는 Google이 제작한 안드로이드 멀티미디어 스트리밍 오픈소스인 Exoplayer를 사용한다.[[20]](#endnote-20) 기존 안드로이드에서 제공하는 내장 멀티미디어 프레임워크인 MediaPlayer는 customizable하지 않고, black box의 형태로 만들어져 있어 디버깅이 어렵다는 단점이 있다. Exoplayer는 customizable하고 디버깅이 쉬워 이러한 단점을 보완한다. 또한 DASH와 HLS를 둘 다 지원하기 때문에, 구현과정에서 Manifest 포맷을 선택적으로 사용할 수 있다.

## 4.2 시스템 구조

### 4.2.1 전체 시스템 구조도

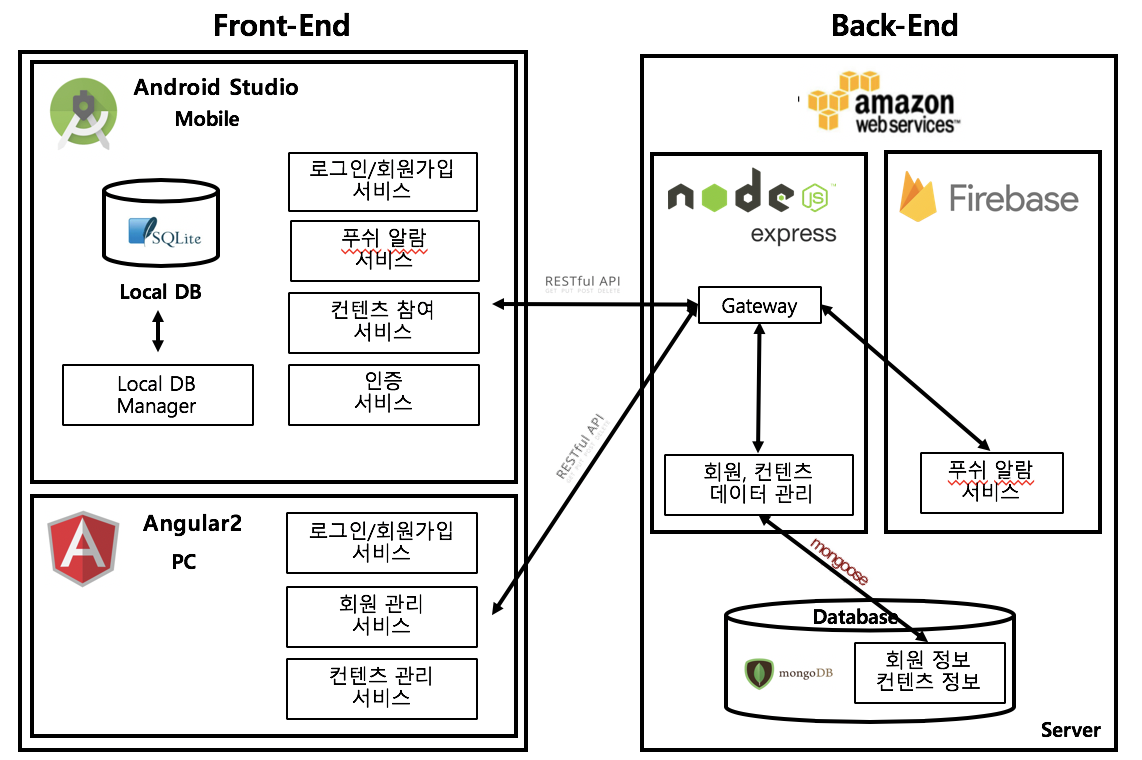


[그림 25] 전체 시스템 구조도

모두의 달성의 전체 시스템 구조는 크게 3가지로 나뉜다. 첫째 모두의 달성을 사용하고, 관리하는 User/Manager, 두 번째 각각 사용자에게 서비스를 제공하기 위한 인터페이스, 마지막으로 모두의 달성 서비스를 운영하기 위한 서버가 있다. 모두의 달성의 구체적인 시스템 구조는 다음과 같다.

* User는 실제 모두의 달성을 사용하는 사용자다. 사용자는 배포된 안드로이드 애플리케이션을 통해 모바일 환경에서 서비스를 사용할 수 있다.
* Manager는 모두의 달성 서비스와 사용자를 관리하는 관리자다. 관리자는 웹 애플리케이션을 통해 PC 환경에서 서비스를 관리하고, 사용자 정보를 관리할 수 있다.
* Server는 Amazon Web Service 위에 Node.js 서버와 MongoDB 데이터베이스가 작동하는 구조이다. Client인 User 또는 Manager로 부터 요청이 들어오면 데이터베이스의 정보를 바탕으로 응답한다. 또한 FCM(Firebase Cloud Messaging)을 이용하여 User에게 수시로 푸시 알람을 전송한다.

### 4.2.2 소프트웨어 구조



[그림 26] 소프트웨어 구조

전체 소프트웨어 구조는 크게 Front랑 Back으로 나뉜다. Front는 모두의 달성을 사용하는 사용자가 보고 접하는 모든 영역을 책임지는 부분이다. Back은 Front가 잘 동작하도록 Front의 요청에 맞게 데이터를 전달하는 역할을 한다. 모두의 달성의 Front와 Back의 구체적인 설명은 다음과 같다.

* Front는 사용자가 사용하는 모바일 애플리케이션과 관리자가 사용하는 웹 애플리케이션 두 종류로 나뉜다. 사용자가 사용하는 모바일 애플리케이션은 Kotlin 언어를 사용하여 Android Studio를 통해 개발할 예정이고, 관리자가 사용하는 웹 애플리케이션의 경우 MEAN Stack의 Front Framework인 Angular2를 사용하여 개발할 예정이다.

Front는 서비스를 제공하기 위해 HTTP request를 통해 Back에 데이터를 요청한다. 요청을 받은 Back은 요청된 정보를 데이터베이스에서 찾아 응답한다. 둘의 요청과 응답은 HTTP 함수와 Uri를 바탕으로 정의된 Restful API를 사용하여 정보를 주고 받을 예정이다.

* Back은 Amazon Web Service 위에 Express Framework를 사용한 Node.js 서버를 올리고, MongoDB를 사용한 데이터베이스를 구축할 예정이다. Front로 부터 HTTP request를 받은 서버는 NoSQL인 MongoDB에서 데이터를 찾거나 수정 후 다시 Front에 응답한다.

Node.js가 MongoDB에 있는 데이터를 접근하기 위해 Mongoose를 사용할 예정이다. Mongoose는 ODB(Object Data Mapping) 기반 Node.js 전용 라이브러리로 스택 전반으로 사용되는 javascript를 사용하여 MongoDB 접근을 가능하게 한다.

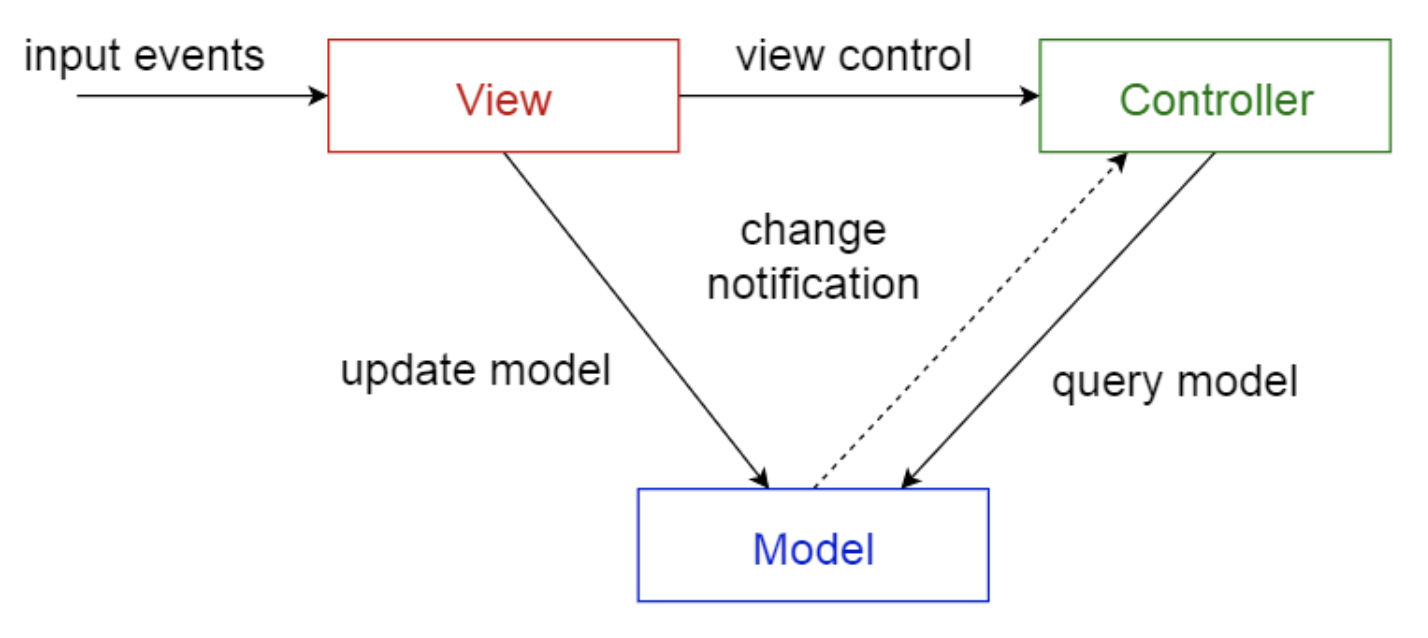
Firebase Cloud Messaging 는 Node.js와 연동하여 필요한 푸시 알람을 Node.js의 Gateway를 통해 Front의 애플리케이션으로 정보를 전달한다.

모바일 애플리케이션 (모두의 달성)은 크게 5개의 컴포넌트로 나뉜다. 각각의 컴포넌트는 모두의 달성이 사용자에게 전체 서비스를 제공하기 위해 필요한 서비스 단위들이다. 각각의 자세한 설명은 다음과 같다.

* 로그인/회원가입 서비스: 사용자에게 회원가입 기능을 제공하고, 회원가입이 완료되면 서비스를 참여할 수 있도록 로그인 기능을 제공한다.
* 푸시 알람 서비스: 컨텐츠에 참여 중인 사용자는 정해진 시간에 맞게 영상을 인증해야한다. 또한 참여 중인 컨텐츠의 다른 사용자의 영상을 인증하기도 한다. 이 때 사용자에게 푸시 알람을 제공하여 인증을 요청한다.
* 컨텐츠 참여 서비스: 사용자가 컨텐츠를 찾고, 참여하는 과정에 관련된 서비스를 제공한다. 사용자는 컨텐츠를 검색하고 콘텐츠의 자세한 정보와 관련된 인증 기기 정보를 제공받은 후 원하는 기간에 참여할 수 있다.
* 인증 서비스: 사용자에게 영상을 찍고 인증할 수 있도록 서비스를 제공한다.
* Local DB Manager: 안드로이드 내부에서 필요한 데이터를 관리한다. 안드로이드에 내장된 SQLite를 통해 데이터를 불러오고, 저장할 수 있다.

웹 애플리케이션의 경우 3개의 컴포넌트로 나뉜다. 각각의 컴포넌트는 관리자에게 관리 서비스를 제공하기 위한 서비스 단위다. 각각의 자세한 설명은 다음과 같다.

* 로그인/회원가입 서비스: 관리자가 자신이 관리자임을 인증하고 관리자 페이지에 접속할 수 있도록 로그인, 회원가입 서비스를 제공한다.
* 회원 관리 서비스: 모두의 달성에 등록된 사용자들을 관리할 수 있는 관리 서비스다. 이를 통해 악성 사용자를 관리하고, 회원 정보를 수정할 수 있다.
* 컨텐츠 관리 서비스: 모두의 달성 컨텐츠를 관리할 수 있는 관리 서비스다. 이를 통해 컨텐츠를 종료, 생성할 수 있다.



[그림 27] MVC 패턴

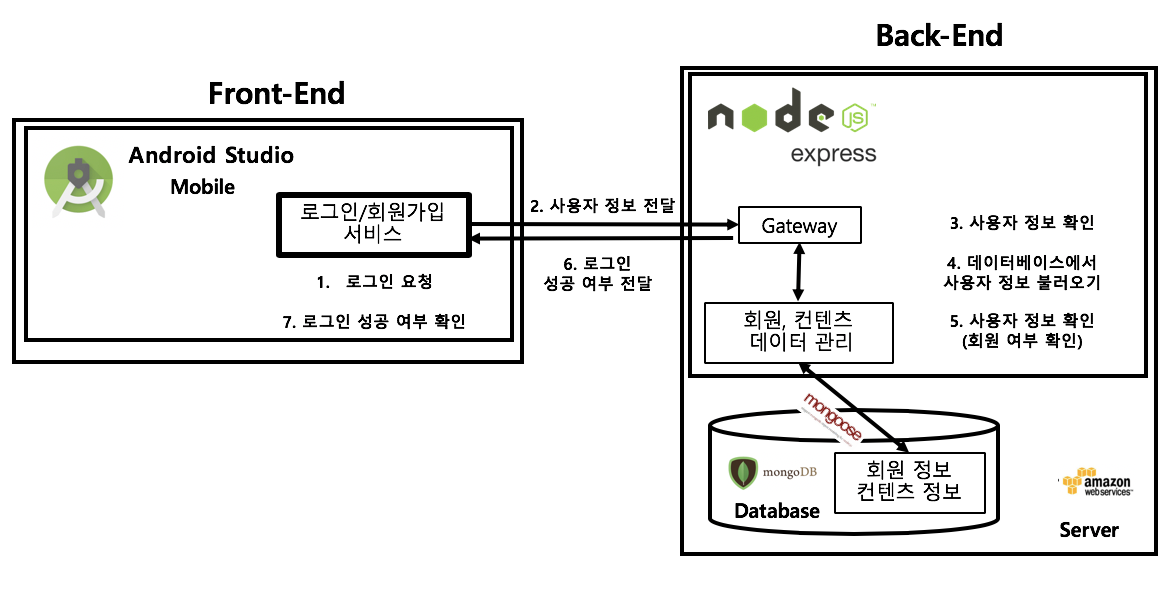
개발에 사용된 소프트웨어 패턴은 MVC(Model-View-Controller) 디자인 패턴이다.[[21]](#endnote-21) MVC 패턴은 사용자 인터페이스(UI)로부터 비즈니스 로직을 분리하여 사용자가 접하는 시각적인 요소와 내부에서 일어나는 비즈니스 로직을 분류한다. 이를 통해 각 컴포넌트간의 Coupling을 낮추어 서로의 영향을 덜 받도록 애플리케이션을 제작할 수 있다. 이러한 점에서 프로젝트 진행에 다음과 같은 장점이 있다.

* 팀 프로젝트에서 역할 분담이 용이하고 확실하다. 프로젝트는 크게 사용자에게 보여주는 Front와 보이지 않는 Back으로 나뉜다. 또 Front에서는 오로지 View를 담당하는 부분과 View를 통제하는 Controller로 나눌 수 있다. 때문에 본 프로젝트에 참여하는 4명의 팀원은 Back과 Front의 역할을 나눌 수 있고, 또 그 내부에서 역할 분담이 용이하게 된다.
* 바뀌는 요구사항의 대처가 빠르다. 본 프로젝트의 개발은 Agile 개발 프로세스를 통해 진행되므로 요구사항이 수시로 바뀔 수 있다. 이렇게 수시로 코드를 바꿔야할 때 MVC 패턴을 적용하면 서로의 영향을 주지 않으며 수정이 되어야할 부분만 수정하면 문제가 해결된다.

다음은 모두의 달성에서 해당하는 Model, View 그리고 Controller의 설명이다.

* Model: 모두의 달성 사용자 데이터, 모두의 달성 컨텐츠 데이터
* View: 모두의 달성의 사용자 인터페이스, 관리자 페이지의 사용자 인터페이스
* Controller: Model을 View에 보여주기 위한 로직

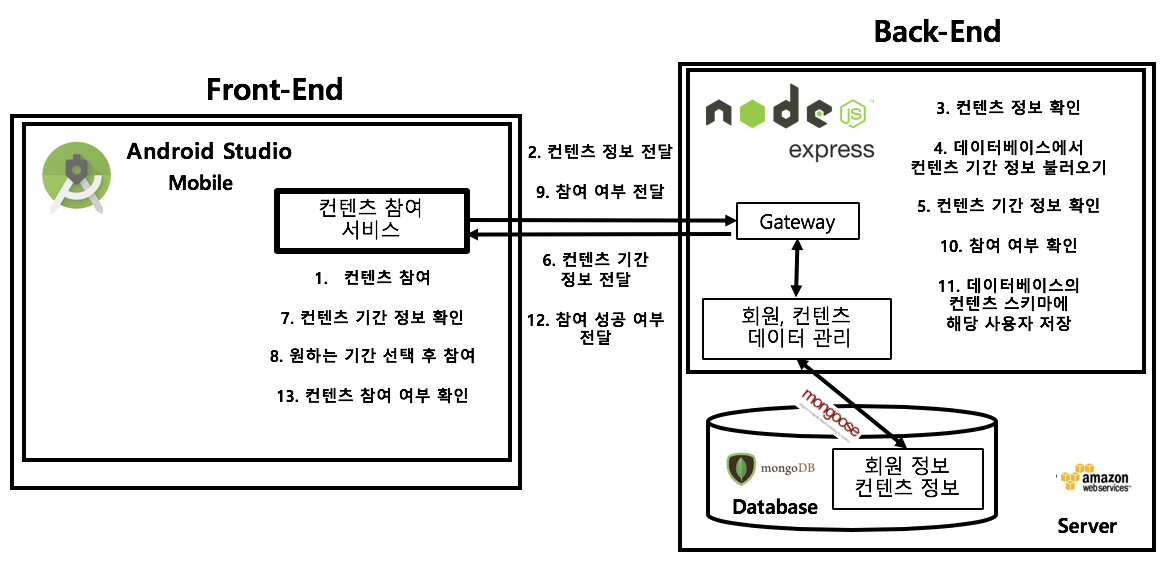
### 4.2.3 시스템 플로우

다음은 모두의 달성의 몇 가지 시나리오에 대한 구체적인 시스템 플로우이다. ****

[그림 28] 로그인 시나리오에서 시스템 플로우

로그인 시나리오

* 사용자는 모두의 달성 애플리케이션에 접속하여 이미 등록된 회원정보를 통해 로그인을 한다. 안드로이드의 View에 보이는 회원 정보 입력란에 자신의 E-mail과 비밀 번호를 입력한 후 로그인 버튼을 누른다. (1)
* 안드로이드의 Front는 Restful API의 정의된 POST, /login을 통해 Back에 요청한다. (2) 요청을 받은 Back은 Front로 부터 받은 사용자 정보를 확인한다. (3)
* Back은 받은 정보를 통해 Mongoose를 사용하여 데이터베이스에 접근 후 해당하는 사용자 정보가 존재하는지 확인한다. (4),(5)
* 사용자가 회원인지 여부를 판단한 Back은 들어온 요청에 로그인 성공 여부를 다시 Front에 응답한다. (6)
* 응답을 받은 Front는 로그인 성공 여부를 알린다. 성공 시 모두의 달성 서비스를 제공하고, 실패 시 회원 정보를 다시 입력 받는다.



[그림 29] 컨텐츠 참여 시나리오에서 시스템 플로우

컨텐츠 참여 시나리오

* 사용자는 원하는 컨텐츠를 찾은 후 참여 버튼을 누른다. (1)
* 참여 버튼 이벤트를 감지한 Front는 Restful API에 정의된 GET, /:contentID/contentJoin를 통해 Back에 컨텐츠 참여 요청을 한다. (2)
* 요청을 받은 Back은 컨텐츠 ID를 확인한 후 데이터 베이스에서 해당 컨텐츠 참여 기간 리스트를 불러온다. (3),(4)
* Back은 데이터베이스에서 받아온 컨텐츠 참여 기간 리스트를 확인 후 Front에 데이터와 함께 응답한다. (5),(6)
* Front는 받은 컨텐츠 참여 기간들을 사용자의 View에 보여준다. (7)
* 사용자는 보이는 컨텐츠 참여 기간 중 원하는 기간을 선택하여 참여 완료 버튼을 누른다. (8)
* 참여 완료 버튼 이벤트를 감지한 Front는 Restful API에 정의된 POST, /:contentID/contentJoinComplete를 통해 기간 데이터와 함께 Back에 컨텐츠 참여 완료 요청을 한다. (9)
* 요청을 받은 Back은 받은 기간 정보를 확인한다. (10)
* Back은 데이터베이스의 접근하여 해당하는 기간의 컨텐츠 스키마에 사용자 정보를 저장한다. (11)
* 다시 Back은 Front에 참여 완료 성공 여부를 응답한다. (12)
* 응답을 받은 Front는 컨텐츠에 참여 완료했는지 여부를 사용자에게 보여준다. (13)

## 4.3 API 정의

### 4.3.1 외부 API

[표 4] 외부 API

|  |  |  |
| --- | --- | --- |
| **API** | **전달 데이터** | **Description** |
| Admin FCM API[[22]](#endnote-22) | 인증 요청 메시지 | Admin FCM API는 Node.js 서버에서 지원하는 API다.  Admin FCM SDK를 사용하여 서버와 FCM을 통합할 수 있고, Admin FCM API를 통해 Firebase 서버 인증과 동시에 클라이언트 (사용자 기기)에 메시지를 보낼 수 있다. |
| Media Codec API[[23]](#endnote-23) | Video 관련 data (compressed data,  raw video data) | Google Exoplayer에서 사용하는 표준 비디오  구성 API |

### 4.3.2 Restful API

모두의 달성의 Front와 Back은 HTTP 통신을 통해 데이터를 주고 받는다. 데이터를 주고 받기 위한 통신을 할 때 Front와 Back은 서로 의도한 정보를 정확히 주고받기 위한 약속이 필요하다. 이러한 약속을 정의 해놓은 것을 Restful API라고 부른다.[[24]](#endnote-24) Restful API를 통해 Back은 웹, 앱의 다양한 브라우저와 통신할 수 있다.

Restful API를 사용하기 위해 Resource (URI)와 HTTP Method를 정의해 놓아야 한다. 각각은 Front와 Back 사이에서 이루어지는 하나의 의도를 가진 HTTP 통신이다. 본 4조에서 정의한 Restful API는 다음 [표 5]와 같다.

[표 5] Restful API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Restful API Type** | **Path** | **RequestData** | **ResponseData** | **Description** |
| POST | /login | userEmail  userPassword | success  failure | 로그인 |
| GET | /logout |  | success  failure | 로그아웃 |
| POST | /signup | userName  userEmail  userPassword  userNickname  userPhonenumber | success  failure | 회원가입 |
| GET | /getUserInfo |  | userName  userEmail  userNickname  userPhonenumber | 사용자 정보  불러오기 |
| GET | /getContentList |  | userJoinedContents | 참가 중인 컨텐츠 정보 불러오기 |
| GET | /getVideoList |  | userVideos | 사용자 인증 영상 불러오기 |
| POST | /userInfoEdit | userPassword  userNickname  userPhonenumber  userImage | success  failure | 사용자 정보  수정 |
| GET | /enterContent/:contentID |  | contentData  (JSON) | 컨텐츠 방 입장  (메인 페이지/ 찾기 페이지) |
| GET | /getAllContentList |  | contentList | 찾기 페이지에서  컨텐츠 목록  불러오기 |
| GET | /getAllUserList |  | userList | 찾기 페이지에서  사용자 목록  불러오기 |
| GET | /enterUser/:userID |  | userData  (JSON) | 유저 방 참가  (찾기 페이지) |
| POST | /:contentID/authorizeVideo | userVideo | success  failure | 동영상 인증하기  (자신) |
| POST | /:contentID/checkVideo | userInfo | videoInfo | 동영상 인증하기  (상대방) |
| GET | /:contentID/contentJoin |  | periodInfo | 컨텐츠 참가하기 |
| POST | /:contentID/contentJoinComplete | periodInfo |  | 컨텐츠 참가 완료 |

## 4.4 데이터 정의

모두의 달성에서는 User Schema, Content Schema 총 2개의 데이터베이스 스키마가 필요하다. User Scheme의 경우 사용자들의 정보를 담고 있고, Content Schema의 경우 컨텐츠의 정보를 담고있다. 본 프로젝트에서는 Mongoose를 사용한 MongoDB를 데이터베이스로 구축할 예정이므로 각각의 Schema를 Key-Value의 JSON 형태로 [표 6]과 [표 7]에 정의 하였다.

[표 6] User Schema

|  |  |  |
| --- | --- | --- |
| **User Schema** | | |
| **모두의 달성 시스템을 이용하는 사용자와 관리자에 대한 정보를 저장한다.** | | |
| **Description** | **KEY** | **Value** |
| **사용자 이름** | **name** | **String** |
| **사용자 E-mail** | **email** | **String** |
| **사용자 비밀번호** | **password** | **String** |
| **사용자 권한** | **authority** | **String** |
| **사용자 전화번호** | **phoneNumber** | **String** |
| **사용자 별명** | **nickname** | **String** |
| **사용자 프로필 사진** | **imagePath** | **String** |
| **사용자 신고 횟수** | **reportCount** | **Number** |
| **사용자 컨텐츠 목록** | **contentList** | **Array** |

[표 7] Content Schema

|  |  |  |
| --- | --- | --- |
| **Content Schema** | | |
| **모두의 달성 시스템이 지원하는 컨텐츠에 대한 정보를 저장한다.** | | |
| **Description** | **Key** | **Value** |
| **컨텐츠 ID** | **id** | **String** |
| **컨텐츠 시작 일자** | **startDate** | **Date** |
| **컨텐츠 종료 일자** | **endDate** | **Date** |
| **컨텐츠 달성률** | **achievementRate** | **Number** |
| **컨텐츠 사용자 목록** | **userList** | **Array** |
| **컨텐츠 완료 여부** | **isDone** | **Number** |

# 5. 과제수행 계획

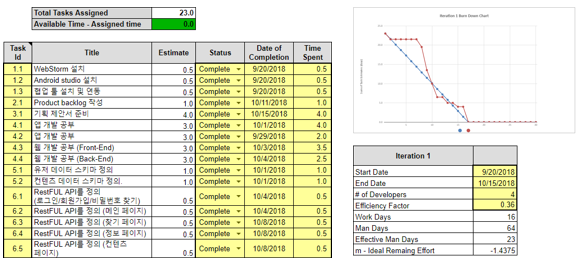
## 5.1 일정 계획

[표 8]은 개략적인 개발 일정을 나타낸 표이다.

[표 8] 개발 일정

|  |  |  |
| --- | --- | --- |
| **Iteration** | **일정** | **할당된 Product Backlog** |
| Iteration 1 | 2018/09/20~2018/10/15 | 프로젝트 개발환경 구축  Product Backlog 1차 작성  기획제안서 작성  개발을 위한 기술 학습  DB Schema 정의  RestFUL API 정의 |
| Iteration 2 | 2018/10/16~2018/10/30 | **<Application>**  User Story 1, 2, 4 - 로그인/회원가입 기능 개발  User Story 8, 9 - 사용자 페이지 개발  User Story 10, 11 - 컨텐츠 및 유저 찾기 기능 개발  User Story 14 - 컨텐츠 참여 방법 제공 개발  User Story 15, 16, 18 - 컨텐츠 참여 신청 기능 개발  **<Web 및 기타 >**  서버 및 디비 구축 |
| Iteration 3 | 2018/10/31~2018/11/14 | **<Application>**  User story 6, 7, 12, 13 - 컨텐츠 정보 확인 기능 개발  User story 19 - 푸시 알람 기능 부분 개발  User story 19, 20, 26 - 카메라 인증 기능 부분 개발  User story 20 – 인증 영상 UI 개발  **<Web 및 기타 >**  User story 34 - 관리자 로그인 기능 개발  User story 40, 41 - 컨텐츠 그룹 및 유저 조회 기능 개발  User story 47, 48, 49 - 컨텐츠 등록 등 관리 기능 개발 |
| Iteration 4 | 2018/11/15~2018/11/29 | **<Application>**  User story 3, 5 - 로그인 추가 기능 개발  User story 21, 27 - 영상 신고 기능 개발  User story 22, 30 - 응원 메시지 전달 기능 개발  User story 28 - 타 유저 개인 보드 접근 기능 개발  User story 29 – 인증 영상 보드 개발  **<Web 및 기타 >**  User Story 35, 36, 37, 38- 비정상 유저 및 영상신고 접수 기능 개발  User Story 39 - 유저 제재 기능 개발 |
| Iteration 5 | 2018/11/30~2018/12/13 | **<Application>**  User story 17 - 컨텐츠 준비 요건 결제 기능 개발  User story 23, 25 - 목표 달성 성공 및 보상 확인 기능 개발  User story 31, 32 - 목표 달성 실패 및 패널티 확인 기능 개발  User story 24, 33 - 목표 달성 성공/실패 설문조사 기능 개발  **<Web 및 기타 >**  User story 42 - 설문조사 및 피드백 확인  User story 43, 44 - 성공한 유저 확인 및 보상 지급 기능 개발  User story 45, 46 - 실패한 유저 확인 및 패널티 부여 기능 개발 |

Iteration 1의 경우, 3주간의 Sprint을 진행하였다. 프로젝트 개발환경 구축, 개발에 필요한 요소에 대한 자료조사, 개발을 위한 배경기술 공부 그리고 기획제안서를 작성하였고, 현재 종료된 상태이다. [그림 30]은 종료된 Iteration 1의 Sprint Backlog이다.



[그림 30] Sprint Backlog(Iteration 1)

이후 Iteration 2부터는 본격적으로 User story를 세분화해 Task로 나누고, 개발을 시작한다. 개발을 시작한 뒤로는 Sprint 기간을 2주로 줄여 더 잦은 피드백을 진행하고, 방향이 잘못될 경우 빠르게 다시 방향을 잡을 수 있도록 하였다. [표 9]는 Sprint 계획 회의를 통해 산출된 Iteration 2의 Sprint Backlog이다.

[표 9] Sprint Backlog (Iteration 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Id** | **Title** | **Estimate** | **Status** | **Assigned**  **To** |
| 1.1 | 애플리케이션 설명 UI를 구현한다. (Front) | 0.5 | To Do | 박승현 |
| 1.2 | 애플리케이션 설명 기능을 구현한다. (Front) | 0.5 | To Do | 박승현 |
| 2.1 | 회원 가입 UI를 구현한다. (Front) | 1.0 | To Do | 조건희 |
| 2.2 | 회원가입 기능을 구현한다. (Front) | 1.0 | To Do | 박승현 |
| 2.3 | 회원가입 기능을 구현한다. (Back) | 1.0 | To Do | 장동익 |
| 4.1 | 로그인 UI를 구현한다. (Front) | 1.0 | To Do | 박승현 |
| 4.2 | 로그인 기능을 구현한다. (Front) | 2.0 | To Do | 조건희 |
| 4.3 | 로그인 기능을 구현한다. (Back) | 2.0 | To Do | 황의권 |
| 14.1 | 사용자 페이지 UI를 구현한다. (Front) | 2.0 | To Do | 조건희 |
| 15.1 | 사용자 페이지 기능을 구현한다. (Front) | 1.0 | To Do | 박승현 |
| 15.2 | 사용자 계정 관리 기능을 구현한다. (Back) | 1.0 | To Do | 황의권 |
| 16.1 | 찾기 기능 UI를 구현한다. (Front) | 1.0 | To Do | 박승현 |
| 16.2 | 찾기 기능을 구현한다. (Front) | 2.0 | To Do | 조건희 |
| 16.3 | 찾기 기능을 구현한다. (Back) | 2.0 | To Do | 장동익 |
| 20.1 | 참여방법을 보여주는 UI를 구현한다. (Front) | 1.0 | To Do | 조건희 |
| 20.2 | 참여방법을 보여주는 기능을 구현한다. (Front) | 1.0 | To Do | 박승현 |
| 21.1 | 참여 신청 UI를 구현한다(Front) | 0.5 | To Do | 조건희 |
| 21.2 | 참여 신청 기능을 구현한다. (Front) | 0.5 | To Do | 박승현 |
| 21.3 | 참여 신청 기능을 구현한다. (Back) | 1.0 | To Do | 황의권 |
| 24.1 | 참여 완료 기능를 구현한다(Front) | 0.5 | To Do | 조건희 |
| 24.2 | 참여 신청 기능을 구현한다. (Back) | 0.5 | To Do | 장동익 |
| 25.1 | 인증 푸시 알람을 구현한다. | 3.0 | To Do | 황의권 |
| 25.2 | 카메라 인증 기능을 구현한다. | 2.0 | To Do | 조건희 |
| 25.3 | 비디오를 서버로 업로드한다. | 1.0 | To Do | 조건희 |
| 26.1 | 내 인증 영상 UI를 구현한다. (Front) | 2.0 | To Do | 박승현 |

이후 Iteration 3, 4, 5 역시 Sprint 계획 회의를 통해 Product Backlog를 세분화하여 Task로 나누고, Sprint Backlog를 위와 같이 구성될 예정이다.

## 5.2 업무 분장

### 5.2.1 개발분야 업무 분장

개발분야 업무 분장의 기준은 다음 [표 10]과 같다. 각각의 팀원은 주 개발 업무를 맡게되고, 담당 파트의 개발이 완료된 경우 부 개발 업무를 시작하게 된다.

[표 10] 개발 분야 업무 분장

|  |  |  |
| --- | --- | --- |
| **팀원 명** | **주 개발 업무** | **부 개발 업무** |
| 박승현 | 안드로이드 개발 | 웹 페이지 프론트엔드 개발 |
| 장동익 | 서버 개발 | 안드로이드 개발 |
| 조건희 | 안드로이드 개발 | 서버 개발 |
| 황의권 | 서버 개발 | 웹 페이지 프론트엔드 개발 |

위의 틀을 기반으로, 현재 진행중인 Iteration 2에서의 업무 분장은 위의 표(Iter2 Sprint Backlog)와 같이 결정되었다. 매 Iteration마다 위의 틀을 기반으로 업무 분장이 이루어질 예정이다.

### 5.2.2 비 개발분야 업무 분장

비 개발분야의 업무 분장은 [표 11]과 같이 이루어져있다.

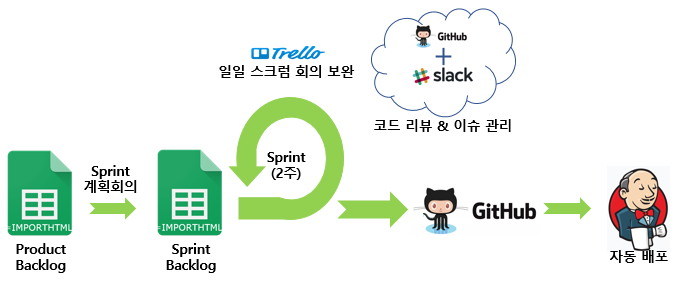
[표 11] 비 개발분야 업무 분장

|  |  |
| --- | --- |
| **팀원 명** | **비 개발분야 업무** |
| 박승현 | 팀장, 공모전 참가 일정 관리, 최종 발표 |
| 장동익 | 지원비 문서 관리, 예산 관리, 중간 발표 |
| 조건희 | 멘토 미팅 일정 관리, 컨셉 발표 |
| 황의권 | 팀 미팅 일정 관리, 기획 발표 |

## 5.3 협업방안

“모두의 달성” 플랫폼 개발 프로젝트를 진행하는 동안, 본 4조는 구글 스프레드 시트, Github, Slack, Trello를 사용해 협업을 진행한다.

아래의 [그림 31]은 각 협업 툴 간의 상호작용을 시각화 한 그림이다.



[그림 31] 협업 툴 간 상호작용

### 5.3.1 Sprint 계획을 위한 구글 스프레드 시트

팀원들이 함께 Product Backlog를 리스팅하고, Sprint 계획 회의 및 스크럼회의를 진행할 때 참고하기 위해 구글 스프레드 시트를 사용한다.[[25]](#endnote-25) 아래 [표 12]는 구글 스프레드 시트에서 유지하고 있는 각 시트와 시트에 대한 설명을 나타낸 표이다.

[표 12] 구글 스프레드 시트 설명

|  |  |
| --- | --- |
| **시트 명** | **설명** |
| Product Backlog | 모든 User story를 리스팅한다.  각 User story별로 Story Point를 책정한다. |
| Sprint Backlog | 매 Sprint 회의마다 선정된 User story를 Task로 세분화 한다.  Task 별 Priority 및 Time Estimate를 책정하고, 담당할 팀원을 정한다. |
| Iteration n (n = 1, 2, 3…) | n번째 Sprint회의 마다, 해당 Sprint 진행기간의 Work Days를 고려하여 최종적으로 계획된 Sprint Backlog를 담고, Burndown chart를 이용해 진행현황을 확인한다.  각 팀원은 자신이 맞은 Task의 Status를 Trello와 구글 스프레드시트에 함께 업데이트해야 한다. |

### 5.3.2 Sprint 진행을 위한 Trello 사용 방안

매 Iteration마다 Trello의 보드를 이용해 Task를 나열한다.[[26]](#endnote-26) 팀원들의 현 진행상황을 쉽게 확인함으로써, 데일리 스크럼회의를 매일 진행할 수 없는 상황을 보완하기 위함이다.

보드의 리스트는 Things to do, Doing, Done 3가지로 나눈다. 각각이 담는 카드의 종류와 명명 규칙은 아래 [표 13]과 같다.

[표 13] Trello 보드 설명

|  |  |  |
| --- | --- | --- |
| **리스트 명** | **카드의 종류** | **카드 명명 규칙** |
| Things to do | 새로 시작된 Sprint의 Sprint Backlog | 모든 Task 카드는  **“Task번호/YYYYMMDD/담당자 명”** 으로 명명한다.  YYYYMMDD는 Things to do로부터 Doing으로 이동시킨 날짜를 표기한다. |
| Doing | Things to do로부터 가져온 현재 진행중인 Task |
| Done | 완료된 Task |

### 5.3.3 소스 변경 관리 및 코드 리뷰를 위한 Slack + Github 사용 방안

Slack과 Github를 연동하여 서로간의 Github commit 현황을 공유한다.[[27]](#endnote-27) Github를 Slack에 연동시킴으로써, Git의 commit이 일어나는 즉시 팀원들은 Slack을 통해 확인 가능하다. 이후 Slack을 통해 commit log를 확인한 팀원들이 코드 리뷰를 진행하는 식으로 Slack과 Github를 사용한다. 다음 [표 14]는 Github와 Slack의 사용 규칙이다.

[표 14] Github & Slack 사용 규칙 (소스 관련)

|  |  |
| --- | --- |
| **사용Tool** | **Github와 Slack** |
| 규칙 | [1] master branch는 작동 가능한 코드만 유지하기 위해 각자 branch에서 작업  [2] branch 명은 “구현한 기능/Task번호”의 형식을 따른다. ex) Login/Task번호  [3] commit message는 “YYYYMMDD-이름-코멘트”의 형식을 따른다.  [4] commit을 한 팀원은 Pull request 를 생성해둔다.  [5] Slack을 통해 commit log를 확인한 팀원은 코드리뷰를 진행한다.  [5] 팀원의 코드 리뷰 후 Pull request에서 commit confirm을 하거나, 수정을 제안한다.  [6] 수정을 제안받은 경우 수정 한 뒤 [3]번부터 다시 시작한다. |

### 5.3.4 이슈 관리를 위한 Slack + Github 사용 방안

프로젝트 진행 중 발생하는 모든 이슈에 대해 Slack과 Github를 통해 커뮤니케이션한다. Github와 연동되어 있는 Slack을 통해 Github Issue를 생성가능하며, 이를 통해 팀원들간에 이슈를 공유한다. 아래 [표 15]와 같은 방식으로 진행된다.

[표 15] Github & Slack 사용 규칙 (이슈 관련)

|  |  |
| --- | --- |
| **사용Tool** | **Github와 Slack** |
| 규칙 | [1] 프로젝트 진행 중 이슈 발생 시, Slack에서 다음의 커맨드를 입력한다.  /github open [https://github.com/geonheec/Capstone\_Team4]  [2] Issue 생성 창이 나타나면, Title, Body를 작성한다.  Title은 명사형태로 제목을 작성하고, Body는 issue에 대한 상세 설명을 작성한다. |

### 5.3.5 빌드 자동배포를 위한 Github + Jenkins 사용 방안

Github를 Jenkins 서버에 연동하여 사용한다. Jenkins 툴을 이용하여 Github의 master branch 소스를 빌드하고, 배포한다.

## 5.4 성과 측정 방안

본 프로젝트의 진행기간 내로 측정 가능한 성과 측정 방안을 아래 [표 16]와 [표 17] 같이 제시한다.

### 5.4.1 정량적 KPI

[표 16] 정량적 KPI

|  |  |  |
| --- | --- | --- |
| **평가 항목** | **설명** | **목표** |
| 스트리밍 시작 전 Delay가 사용자가 불편함 없이 사용할 수준인가? | LTE 환경을 기준으로, 인증영상 스트리밍 시작 요청 후 스트리밍이 시작되기까지의 Delay가 사용자에게 불편함이 없는 수준이어야 한다. | Delay가 0.5초를 넘지 않도록 하여 사용자가 이용하는데에 불편함이 없어야 한다. |
| 인증 영상 촬영 후 서버로의 전송이 백그라운드에서 수행되도록 하였는가? | 촬영된 인증영상이 서버로 전송되는 과정을 백그라운드에서 실행시켜 사용자가 불편함을 느끼지 않도록 해야한다.  컨텐츠에 따라 동영상의 길이는 가변적이고, 긴 영상의 경우 전송에 오랜시간이 걸릴 수 있어 이는 필수적이다. | 촬영된 인증영상을 서버로 전송하는 과정이 백그라운드에서 수행되어야 한다. |

### 5.4.2 정성적 KPI

[표 17] 정성적 KPI

|  |  |  |
| --- | --- | --- |
| **평가 항목** | **설명** | **목표** |
| 타인과 함께하는 듯한 사용자 경험을 제공하였는가? | 타 목표달성 서비스와의 차별점으로써, 인증 영상 스트리밍 서비스를 통해 타인과 함께 계획실천을 하고있음을 사용자가 느낄 수 있어야 한다. | 이 플랫폼을 사용했을 때 정말로 타인에게 감시받는다는 느낌이 들고, 또한 타인을 감시한다는 느낌이 들도록 한다. |
| 영상을 통해 계획 실천여부를 확실히 판단 가능하도록 하였는가? | 타 목표달성 서비스와의 차별점으로써, 사용자가 촬영한 계획 실천 인증영상에 있어 조작의 여지가 없도록 기획하여야 한다. | 타인의 인증 영상을 보고, 계획을 실천했는 지에 대해 의심없이 판단가능해야 한다. |
| 10대부터 40대까지의 연령층 모두가 쉽게 서비스를 이용할 수 있도록 UI가 구현되어 있는가? | 본 서비스를 주로 이용할 10대~40대의 사용자 모두가 쉽게 서비스를 이용할 수 있는 사용자 경험을 제공해야 한다. | 캡스톤 디자인의 다른 조 학우님들과 교수님이 서비스를 처음 사용하실때 아무런 불편함 없이 수월하게 이용가능해야 한다. |

1. 김준래, 「새해 결심, 왜 ‘작심삼일’로 끝날까?」, TheScienceTimes, <https://www.sciencetimes.co.kr/?news=새해-결심-왜-작심삼일로-끝날까> [↑](#endnote-ref-1)
2. 최동진, 「새해다짐 ‘작심삼일’ 이유는 뇌의 브레이크 작동」, HelloDD, <http://hellodd.com/?md=news&mt=view&pid=47170> [↑](#endnote-ref-2)
3. 이미연, 「[눈에 띄는 앱] 목표를 함께 공유하고 해결하는 SNS ‘옵티(Obti)’」, Appetizer, <http://app.chosun.com/site/data/html_dir/2013/11/04/2013110401278.html> [↑](#endnote-ref-3)
4. 이가영, 「신년계획 작심삼일? 도전금 내고 목표 달성한다」, 증기이코노미, <http://www.junggi.co.kr/article/articleView.html?no=20759> [↑](#endnote-ref-4)
5. 김태헌, 「이주의 추천 앱 세계 금연의 날엔 ‘금연시작’ 」, 한국경제매거진, <http://magazine.hankyung.com/business/apps/news?popup=0&nid=01&c1=1013&nkey=2016053001070000271&mode=sub_view> [↑](#endnote-ref-5)
6. Jan Ozer, 「What is Streaming? 」, streamingmedia.com, <http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=74052> [↑](#endnote-ref-6)
7. 박수련, 「포털만큼 커진 앱 마켓… 한국인들 1년에 7조 이상 쓴다」, 중앙일보, <https://news.joins.com/article/22351582> [↑](#endnote-ref-7)
8. 박재호 외 2명, 『풀스택 개발자를 위한 MEAN 스택 입문』, 서울 한빛미디어, 2015 [↑](#endnote-ref-8)
9. 안개발자,「 [Mean stack] Mean stack이란 무엇인가? 」, <http://unordinarydays.tistory.com/116> [↑](#endnote-ref-9)
10. g00glen00b, 「Working with the Mean stack: MVC」, <https://g00glen00b.be/mean-mvc/> [↑](#endnote-ref-10)
11. 이웅모, 『Angular Essentials』, 루비페이퍼, 2018.07.04 [↑](#endnote-ref-11)
12. 김현아, 「국내 스마트폰 OS 점유율, 안드로이드IOS」, <http://www.edaily.co.kr/news/read?newsId=01305446612552816&mediaCodeNo=257> [↑](#endnote-ref-12)
13. 박민우 외 1명, 「 안드로이드 공식 언어가 된 Kotlin을 알아보자」, <https://academy.realm.io/kr/posts/kotlin-official-android-language/> [↑](#endnote-ref-13)
14. 위키백과, 「통합개발환경」, <https://ko.wikipedia.org/wiki/통합_개발_환경> [↑](#endnote-ref-14)
15. 김태은, 「안드로이드를 기반으로 하는 게임 콘텐츠 개발에 대한 연구」, 한국 디지털 콘텐츠 학회 논문지, 제17권 제2호, 2016.4 [↑](#endnote-ref-15)
16. 권영재, 「테스트 주도 개발방법론이란? 」, <https://nesoy.github.io/articles/2017-01/TDD> [↑](#endnote-ref-16)
17. philip, 「webstorm 소개」, <http://philiplog.tistory.com/39> [↑](#endnote-ref-17)
18. Laurence Moroney, 『The Definitive Guide to Firebase』, Apress,2017. [↑](#endnote-ref-18)
19. Michael Seufert외 5명, 「A Survey on Quality of Experience of HTTP Adaptive Streaming', IEEE Communications Surveys & Tutorials volume17」, 2015 [↑](#endnote-ref-19)
20. Google Developer, Exoplayer, <https://developer.android.com/guide/topics/media/exoplayer> [↑](#endnote-ref-20)
21. Steve Burbeck, Ph.D,

    「Applications Programming in smalltalk – 80 : How to use Model-View-Controller (MVC) 」, waybackmachine, [https://web.archive.org/web/20090801040629/http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html](https://web.archive.org/web/20090801040629/http:/st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html) [↑](#endnote-ref-21)
22. Firebase, 「Admin FCM API 소개」 ,<https://firebase.google.com/docs/cloud-messaging/admin/?hl=ko> [↑](#endnote-ref-22)
23. GoogleDeveloper, 「MediaCodec」, <https://developer.android.com/reference/android/media/MediaCodec> [↑](#endnote-ref-23)
24. Y.M.Pakr, A.K.Moon 외 3명, 「SOAP-based Web Services vs. RESTful Web Services」, 전자 통신 동향, ETRI, Vol.25, No.2, 2010. [↑](#endnote-ref-24)
25. [↑](#endnote-ref-25)
26. [↑](#endnote-ref-26)
27. [↑](#endnote-ref-27)