|  |
| --- |
|  |



|  |
| --- |
| **TUPLI ~Youtube Playlist~** |
| **명세 기술서** |
|  |
|  |
|  |
|  |
|  |



**목차**

[I. 개요 2](#_Toc95286426)

[1. 배경 2](#_Toc95286427)

[II. 명세 달성현황 2](#_Toc95286428)

[1. SubPJT I 2](#_Toc95286429)

[가) 로그인 기능 2](#_Toc95286430)

[나) 회원 관리 3](#_Toc95286431)

[다) Page Not Found 페이지 5](#_Toc95286432)

[라) Error 페이지 5](#_Toc95286433)

[2. SubPJT II 6](#_Toc95286434)

[가) 계정 설정 6](#_Toc95286435)

[나) 유저 페이지 7](#_Toc95286436)

[다) 팔로우 기능 9](#_Toc95286437)

[라) 알림 센터 10](#_Toc95286438)

[마) 헤더 및 네비게이션 12](#_Toc95286439)

[바) 검색 기능 14](#_Toc95286440)

[사) 뉴스피드 기능 15](#_Toc95286441)

[아) 댓글 기능 17](#_Toc95286442)

[3. SubPJT III 18](#_Toc95286443)

[가) 유저 페이지 (추가) 18](#_Toc95286444)

[나) 큐레이션 세트 설정 19](#_Toc95286445)

[다) 게시글 큐레이션 기능 21](#_Toc95286446)

[라) 큐레이션 리스트 21](#_Toc95286447)

[마) 트렌드 24](#_Toc95286448)

# 개요

## **배경**

공통 프로젝트는 사용자 경험(UX) 중심의 웹 서비스를 완성하는 것이 목표입니다. 본 기술서에서는 이번 프로젝트에서 그 목표 달성을 위해서 어떤 상황과 조건이 주어졌고, 프로젝트를 진행하면서 이를 어떻게 달성해 나갔으며 무엇을 보고 느꼈는지에 대해 정리해보고자 합니다.

# 명세 달성현황

## **SubPJT I**

### 로그인 기능

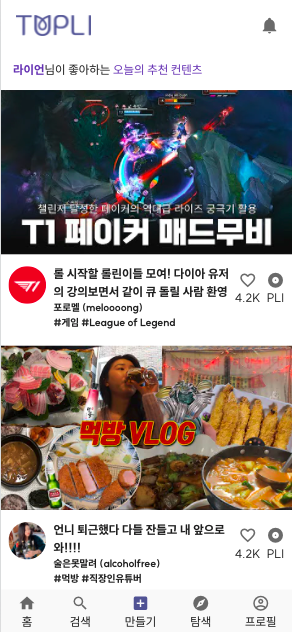
로그인 페이지에 필요한 UX 디테일입니다.

**<고려사항>**

* **랜드페이지를 로그인 페이지가 아닌 홈(‘/’)으로 구현**

유저가 처음 TUPLI 사이트에 진입하였을 때 보이는 랜드 페이지를 로그인 페이지가 아닌 홈 페이지로 하여 서비스 진입 장벽을 낮추었습니다.

또한, 회원과 손님에 따라 표시하는 화면을 다르게 하여 차후에 표시할 피드 요소나 추천, SNS 기능을 활용할 수 있게 하였습니다.



<그림1. 랜드페이지>

**<Req. 1-1 로그인 페이지 커스터마이징>**

* **모바일에서 입력 시 이메일 Input의 첫 글자가 대문자가 되는 현상으로 인해 로그인 실패가 발생하지 않도록 구현**

Vuetify의 text-fields로 받아지는 변수를 method로 보내서, 로그인 창에서 이메일 입력 시 첫 글자는 소문자로 변경되도록 toLowerCase 처리하였습니다.

* **로그인 실패 시 사용자에게 실패 사유 에러메세지 노출**

아이디 또는 비밀번호가 달라서 로그인에 실패하였을 경우 “아이디 또는 비밀번호를 확인해주세요” 메시지를 출력합니다. “아이디를 확인해주세요” 또는 “비밀번호를 확인해주세요”처럼 어느 입력이 틀렸는지 짚어주지 않는 이유는 보안을 강화하기 위함입니다.

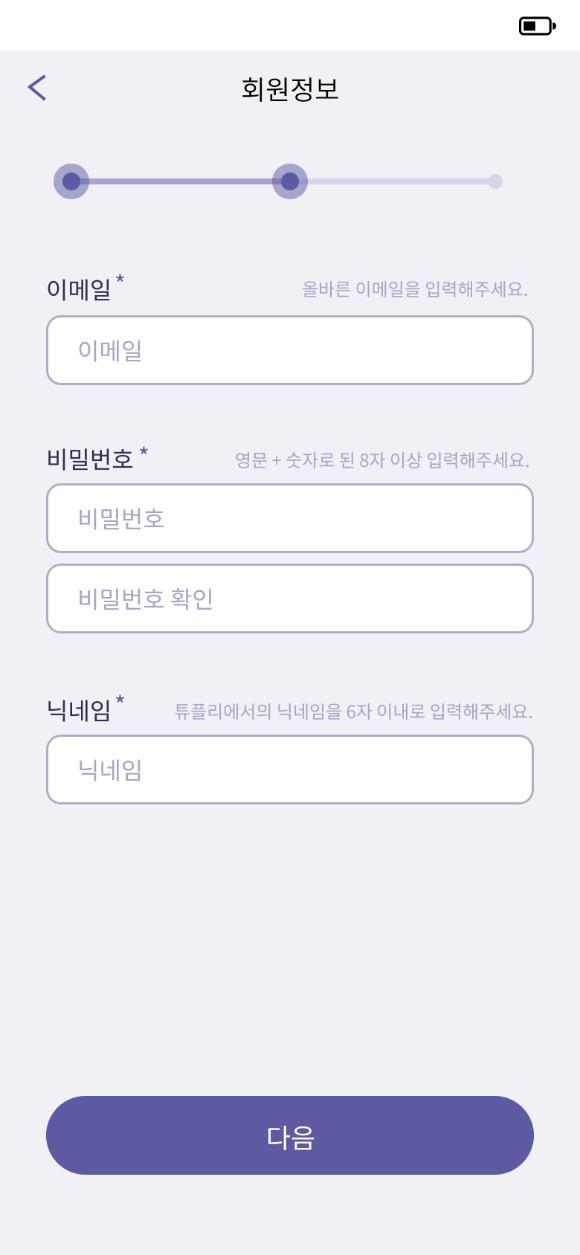
* **이메일 형식 입력 및 비밀번호 입력 기준 충족 시에만 ‘로그인’ 버튼 활성화**

프론트엔드에서는 JavaScript 정규표현식을 통해 이메일 형식 및 비밀번호 입력 기준을 정해주었습니다. 해당 기준을 충족하지 않으면 로그인 버튼을 활성화하지 않습니다. 이에 더불어서 입력이 미흡할 경우에는 입력 틀을 붉게 변화시키고 안내문을 표시함으로써, 사용자가 현재 어느 부분이 잘못되었는지 직관적으로 알 수 있게 처리하였습니다.

백엔드 단에서는 기본적으로 Spring Validation Bean을 이용하여 이메일 형식 및 비밀번호 입력 기준을 정해주었습니다. 또한, 계정 관리는 서비스 운영에 매우 중요한 부분이기 때문에, 프론트에서 체크한 부분도 백엔드 서버에서 다시 한 번 체크함으로써 외부 툴이나 프론트의 제한을 무시한 요청도 처리할 수 있게 구성하였습니다.

### 회원 관리

회원가입 페이지에 필요한 UX 요구사항입니다.

**<Req. 2-1 회원가입 페이지 제작>**

<그림2. 회원가입 페이지>

* **가입 필수 항목 모두 입력 시에만 ‘가입 완료’ 버튼 활성화**

가입 필수 항목(이메일, 비밀번호, 비밀번호 확인, 닉네임, 이름)을 전부 입력해야만 ‘회원가입’ 버튼이 활성화되어 가입을 진행할 수 있도록 구현하였습니다.

* **이메일, 비밀번호 등 입력 기준이 충족되지 못할 시 오류 메시지 노출**

가입 필수 항목을 전부 입력하지 않을 시, 어느 항목이 누락됐는지 메시지를 출력해줍니다.

* **모바일에서 입력 시 이메일 Input 의 첫 글자가 대문자가 되는 현상이 발생하지 않도록 구현**

Vuetify의 text-fields로 받아지는 변수를 method로 보내서, 회원가입창에서 이메일 입력 시 첫 글자는 소문자로 변경되도록 toLowerCase 처리하였습니다.

* **회원가입 완료 시 사용자에게 피드백 노출**

회원가입 완료 시 입력한 정보를 토대로 자동으로 로그인되고, 환영 메시지가 있는 창이 표시되며, 메인 페이지로 이동합니다. 이로 하여금 사용자는 회원가입이 완료되었음을 알 수 있습니다.

**<Req. 2-2 회원가입 완료 페이지 제작>**

* **회원가입을 완료하면 “회원가입 인증 메일이 발송되었습니다. 이메일을 확인해주세요” 메시지 노출**

소셜 로그인으로 가입하였을 경우에 해당 이메일로 TUPLI에 회원가입을 축하한다는 메일이 발송됩니다. 일반 회원으로 가입하였을 경우, 이메일 인증을 바로 받는 것은 회원 가입의 진입장벽을 올리는 행위라고 생각하기 때문에, 관련 서비스나 메일 서비스를 받고자 하는 사람들에 한해서 프로필에 신청하여 인증을 받아 이용하는 2단계 가입 시스템을 도입하였습니다.

* **메일 재발송, 메일함으로 이동 등 사용자 편의 사항을 고려하여 필요한 버튼, 링크 등을 적용**

필요한 서비스나 화면이 있으면 바로 이동할 수 있는 식으로 라우터를 최대한 활용하였습니다. 그 외에도 버튼을 누르면 재발송, 새로고침등의 API 재전송을 고려한 설계 버튼 등을 적용하였습니다.

**<Req. 2-3 비밀번호 변경 페이지 제작>**

* **비밀번호 입력 기준이 충족되었을 때 ‘저장’ 버튼 활성화**

JavaScript 정규화 식을 이용하여 비밀번호 입력 기준을 정하였고, 이를 충족하였을 경우에만 ‘저장’ 버튼이 활성화 되도록 하였습니다.

* **비밀번호 변경 성공 시 사용자에게 피드백 노출**

성공하였을 경우 다음 페이지로 넘어가게끔 구현하였습니다. 따라서 사용자는 비밀번호가 바뀌었음을 알 수 있습니다.

* **비밀번호 변경 실패 시 사용자에게 실패 사유 에러메세지 노출**

비밀번호 작성 기준에 충족되지 않았을 경우, ‘비밀번호는 8글자 이상으로 작성해주세요’ 메시지를 출력합니다. 비밀번호 현재 비밀번호가 틀렸을 경우 ‘비밀번호를 확인해주세요’를 출력, 변경할 비밀번호의 2차 확인이 틀렸을 경우 ‘새 비밀번호가 다릅니다’를 출력합니다.

### Page Not Found 페이지

**<Req. 3-1 Page Not Found 페이지>**

* **존재하지 않는 URL 요청 시 Page Not Found 페이지로 이동**

존재하지 않는 페이지로 이동하였을 경우 커스텀한 Page Not Found 페이지로 이동하고, “요청하신 페이지를 찾을 수 없습니다. 입력하신 주소를 확인해주세요.” 라는 양해의 메시지를 노출합니다. 그리고 3초 뒤 메인 페이지로 이동합니다.

### Error 페이지

**<Req. 4-1 Error페이지>**

* **웹 페이지에 오류가 발생하는 경우 Error 페이지로 이동**

웹 페이지에 오류가 발생한 경우 커스텀한 Error 페이지로 이동하게 되고, “서버에 일시적인 에러가 발생하였습니다. 불편을 겪게 해드려서 대단히 죄송합니다.” 라는 메시지를 노출합니다. 그리고 3초 뒤 메인 페이지로 이동합니다.

## **SubPJT II**

## 

### 계정 설정

계정 설정에 관련된 UX 요구사항입니다.

**<Req. 1-1 계정 설정>**

* **계정 설정 화면에서는 닉네임, 비밀번호, 한 줄 소개 변경 가능**

프로필 편집 화면에서 닉네임과 자기소개 글, 그리고 프로필 이미지를 변경할 수 있습니다.

비밀번호는 설정 탭 -> 비밀번호 변경 화면에서 변경할 수 있습니다. 비밀번호 변경에는 기존 비밀번호와 새로운 비밀번호, 비밀번호 확인 세 종류의 정보를 요구합니다.

<그림3. 설정 탭 속 비밀번호 변경 페이지>

* **한 번 가입된 이메일 정보는 변경 불가**

이메일 정보는 회원의 아이디이자, 고유한 정보로서 수정할 수 없도록 설정하였습니다.

* **계정 설정에서 입력된 데이터가 없을 경우 ‘저장’ 버튼 비활성화**

프로필 편집중에, 반드시 필요한 정보를 입력하지 않으면 ‘저장’ 버튼이 활성화되지 않도록 설정하였습니다. 또한, 프로필 편집 페이지로 이동하면 기존 정보가 입력된 상태로 페이지가 로딩되며 내용 변경이 일어나면 ‘저장’ 버튼이 활성화됩니다.

* **비밀번호 변경 이메일 전송 후 사용자에게 피드백 노출**

비밀번호가 수정될 경우 “비밀번호 수정이 완료되었습니다” 라는 팝업을 노출시킵니다.

* **계정 설정 변경 완료 시 사용자에게 피드백 노출**

프로필 편집이 완료되었으면, ‘프로필 정보가 변경되었습니다.’ 라는 문구를 노출합니다.

* **설정이 변경된 상태에서 저장하지 않고 화면을 벗어날 경우 경고 메시지 표시**

사용자의 행동에 제지를 건다는 점에서 불쾌한 경험이 될 수 있다고 생각하여 적용하지 않았습니다. 다만, 반대로 프로필 변경 도중에 실수로 페이지에서 이탈하게 되어 편집중인 정보를 모두 잃는 상황을 막기 위해, Yes / No 로 선택할 수 있는 팝업창에 ‘저장하시겠습니까?’ 라는 메시지를 노출하여 불쾌한 경험을 막기 위한 추가적인 노력을 했습니다.

### 유저 페이지

유저 페이지에 관련된 UX 요구사항입니다.

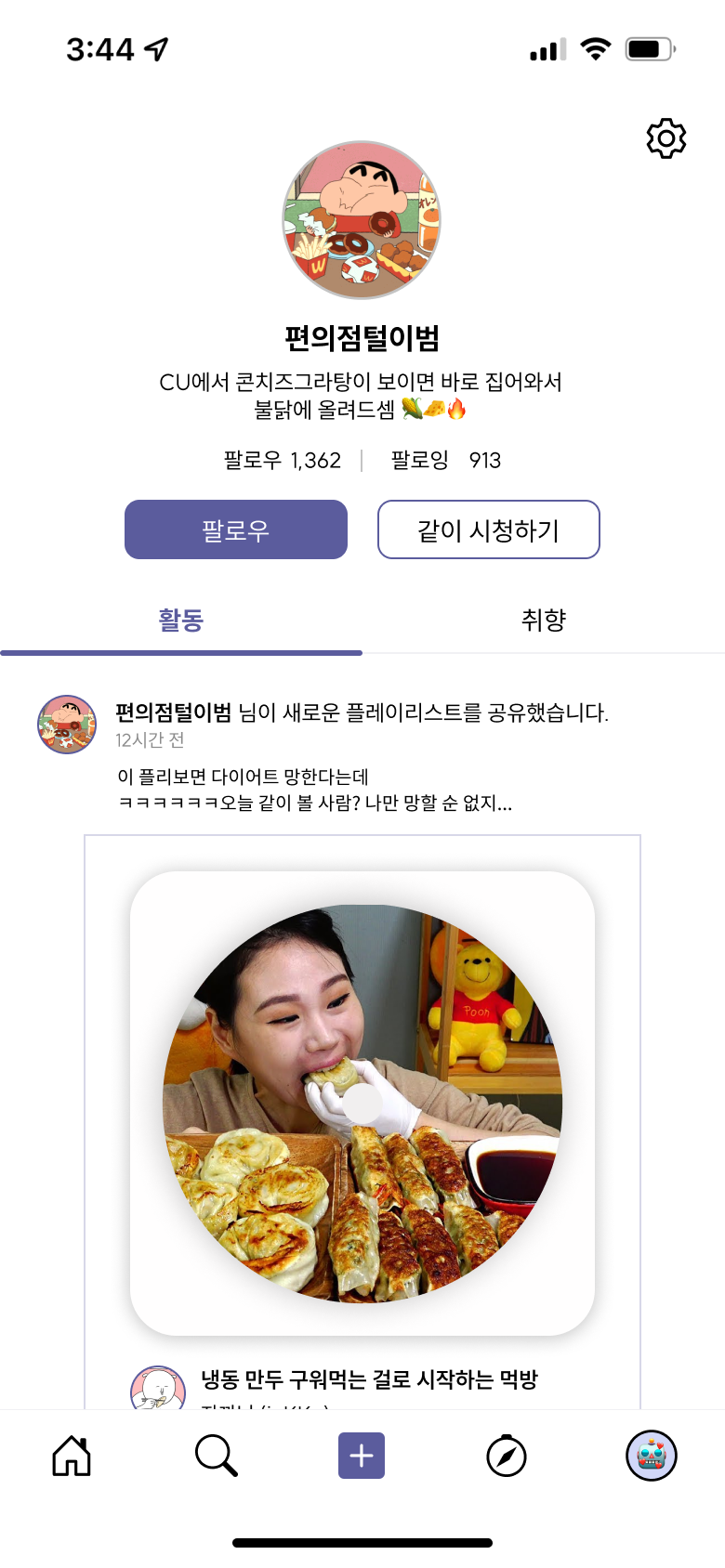
**<Req. 2-1 본인 프로필>**

* **최상단에 유저 정보, 받은 좋아요 수, 팔로우/팔로워 수 노출**

본인 프로필에 접속하면 유저 정보, 받은 좋아요 수, 팔로우/팔로워 수를 노출합니다.

* **프로필 정보 수정 버튼 노출**

본인 프로필에 접속하면 ‘프로필 편집’하기 버튼이 있고, 정보를 수정한 뒤 ‘변경’ 버튼을 누르면 수정이 완료됩니다.

**<Req. 2-2 타 유저 프로필>**

<그림4. 타 유저 프로필에서 찾을 수 있는 팔로우 버튼>

* **팔로우한 유저가 아닌 경우 팔로우 요청 버튼 노출**

타 유저의 프로필 화면에 접속했을 때, 팔로우한 유저가 아니라면 “팔로우” 버튼을 노출합니다.

* **팔로우한 유저인 경우 팔로우 취소 버튼 노출**

타 유저의 프로필 화면에 접속했을 때, 이미 팔로우한 유저라면 “팔로잉” 버튼을 노출합니다. 해당 “팔로잉” 버튼을 누를 시, 팔로우 취소가 됩니다.

* **프로필 최상단에 유저 정보, 받은 좋아요 수, 팔로우/팔로워 수 노출**

타인의 프로필에 접속하면 유저 정보, 받은 좋아요 수, 팔로우/팔로워 수를 노출합니다. 유저에 대한 좋아요 표시는 팔로우로 대체하였습니다.

* **팔로우, 좋아요 등의 상태 변화가 즉각 나타날 수 있도록 프론트엔드에서 우선 처리**

팔로우, 좋아요 등의 상태변화를 프론트엔드에서 우선적으로 처리하여 사용자가 빠르게 체감할 수 있도록 하였습니다. 실질적인 DB 변화 또한 즉각적으로 적용됩니다.

### 팔로우 기능

팔로우 기능에 관련한 UX 요구사항입니다.

**<Req. 3-1 팔로잉 리스트>**

* **로그인한 본인이 팔로우하는 유저의 리스트 노출**

본인의 프로필 화면에서 ‘팔로잉’ 버튼을 누르면 본인이 팔로우하는 유저의 리스트를 확인할 수 있게 하였습니다.

* **팔로잉 상태 변화를 프론트엔드에서 우선 처리**

타 유저를 팔로우 하였을 때, 팔로우가 되었다는 변화를 프론트엔드에서 우선적으로 처리하여 사용자가 빠르게 체감할 수 있도록 하였습니다.

**<Req. 3-2 팔로워 리스트>**

* **로그인한 본인을 팔로우하는 유저의 리스트 노출**

본인의 프로필 화면에서 ‘팔로워’ 버튼을 누르면 본인을 팔로우하는 유저의 리스트를 확인할 수 있게 하였습니다.

* **팔로워 상태 변화를 프론트엔드에서 우선 처리**

팔로워를 맞팔로우 하거나, 팔로우 취소를 할 경우 그 변화를 프론트엔드에서 우선적으로 처리하여 사용자에게 빠른 체감 속도를 느낄 수 있도록 하였습니다.

### 알림 센터

**<Req. 4-1 알림 센터>**

* **알림 센터는 2개의 탭(알림, 요청)으로 구성**

TUPLI에서의 요청은 기본적으로 대상의 동의를 구하지 않아서 따로 이를 저장하거나 관리할 필요가 없습니다. 허락을 구한다’는 사용자 행동에 망설임이 될 요소는 없는게 좋다고 생각했기 때문입니다. 그렇게 요청이 없다면 할 수 없는게 생기지 않을까 생각하실 수 있지만, 행동의 허락을 구하거나 기다리는 것보다 같은 효과를 가지는 다른 행동을 취하게 함으로써 이를 더 좋은 방법으로 구현하였습니다.

예를 들어 팔로우의 경우 허락을 구하는 것보다는 대상에게 본인이 팔로우 됨을 알리고, 팔로우한 사람의 프로필로 넘어갈 수 있게 하여 맞팔로우할 수 있는 환경을 조성하고, 맞팔로우함으로써 얻을 수 있는 이득을 별도로 적용하여 기존의 방식으로 할 수 있는 서비스를 제공함과 동시에 사용자 행동을 방해하는 요소를 제거하였습니다.

* **알림 뱃지 카운터는 Firestore를 이용하여 실시간 업데이트**

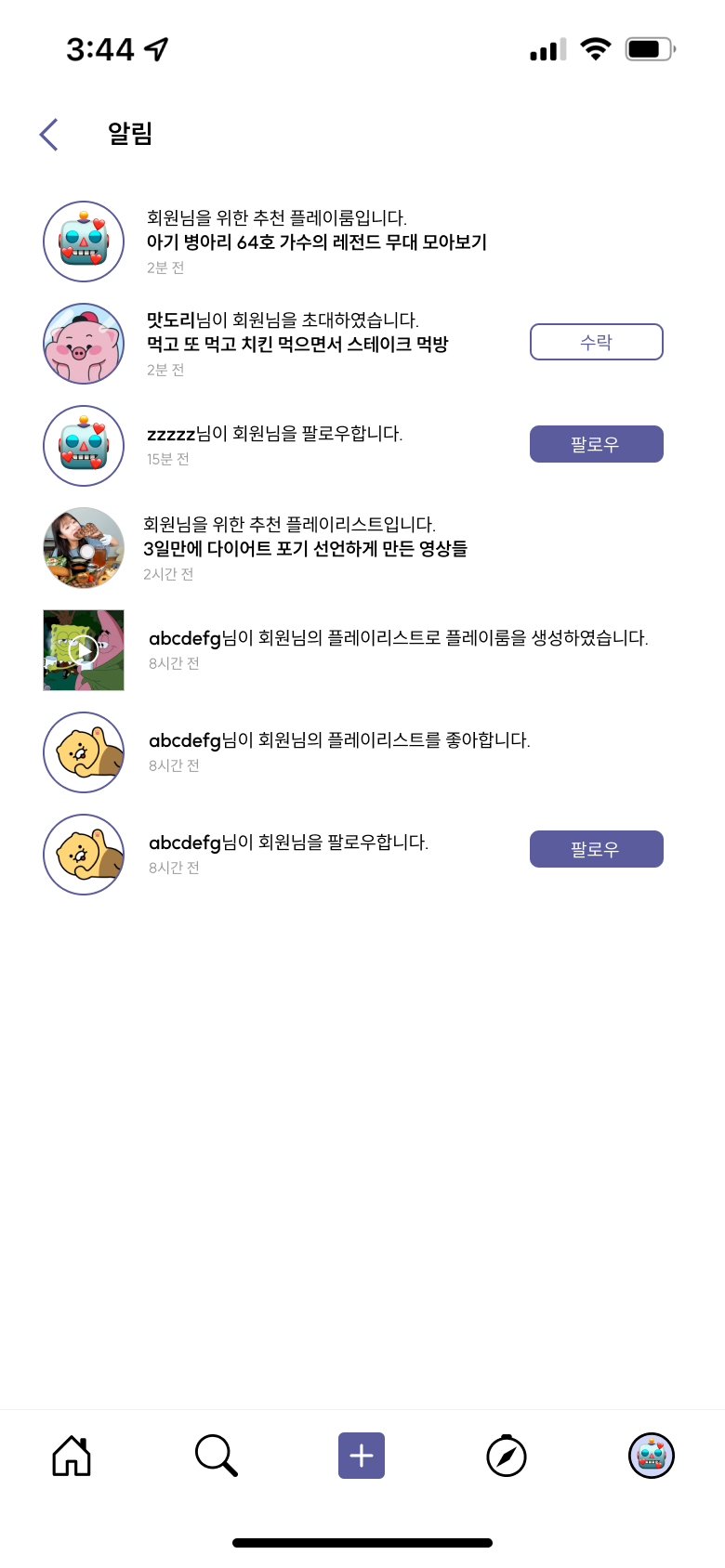
알림 메시지 관련 데이터는 Firebase Realtime Database에 저장되며 DB에 변동이 생기면 실시간으로 적용되어 알림 뱃지의 카운트를 갱신합니다.

* **유저가 확인하지 않은 신규 알림이 존재할 경우 ‘새 알림’ 표시**

아직 읽지 않은 알림의 개수가 알림 뱃지에 표시됩니다.

* **각 탭 전환 시 해당 탭이 선택되어 있음을 강조**

TUPLI에서 알림 센터는 탭 전환이 필요없이, 하나의 탭에서 알림과 요청에 대한 메시지가 시간순으로 출력되게 하였습니다.

****

<그림5. 알림 센터에서 수신한 알람메세지>

**<Req. 4-2 알림 탭>**

* **신규 알림/기존 알림을 구분하여 노출**

TUPLI의 알림은 속성 isRead를 가져서 이것을 통해, 이 알람을 읽었는지 안 읽었는지 구분할 수 있습니다. 색상, 순서를 통한 구분은 고도화 단계에서 이루어질 예정입니다.

**<Req. 4-3 요청 탭>**

* **<Req. 4-1> 에서 언급한 것과 같이 요청은 구현하지 않았습니다.**

대상의 요청을 구한다는 행위가 행위 자체 동기를 저해할 수 있다고 생각했습니다. TUPLI에서는 차단과 맞팔로우를 구현함으로써 같은 행위를 하더라도 조금 더 좋은 UX를 경험할 수 있게 구축하였습니다.

* **본인을 팔로우 요청한 유저의 리스트를 노출**

요구사항 : 요청 탭에 접근하면, 본인을 팔로우 요청한 유저의 리스트를 확인할 수 있도록 하였습니다.

달성방식 : 알림 탭에 접근하면, 본인을 팔로우한 유저의 알림내역을 볼 수 있고, 그 알림을 클릭함으로서 대상을 쉽게 맞팔로우 할 수 있는 환경을 조성하였습니다.

* **유저 리스트와 함께 승인, 거절 버튼 노출**

본인을 팔로우한 유저의 리스트 옆에는 맞팔로우를 할 수 있는 버튼을 같이 표시해주었습니다. 팔로우 승인의 의미가 맞팔로우로 대체되고, 팔로우를 거절하는 행위는 필요하지 않다고 판단하여 구현하지 않았습니다. 대신 특정 유저에 대한 차단 기능이 있어서, 팔로우 하기 싫은 사람을 차단할 수 있습니다.

* **긍정적인 액션인 ‘승인’ 버튼을 우측에 배치**

자신을 팔로우한 유저의 우측에는 ‘맞팔로우’ 버튼이 존재합니다. 자신을 팔로우하지 않은 유저는 ‘팔로우’ 버튼이 노출됩니다.

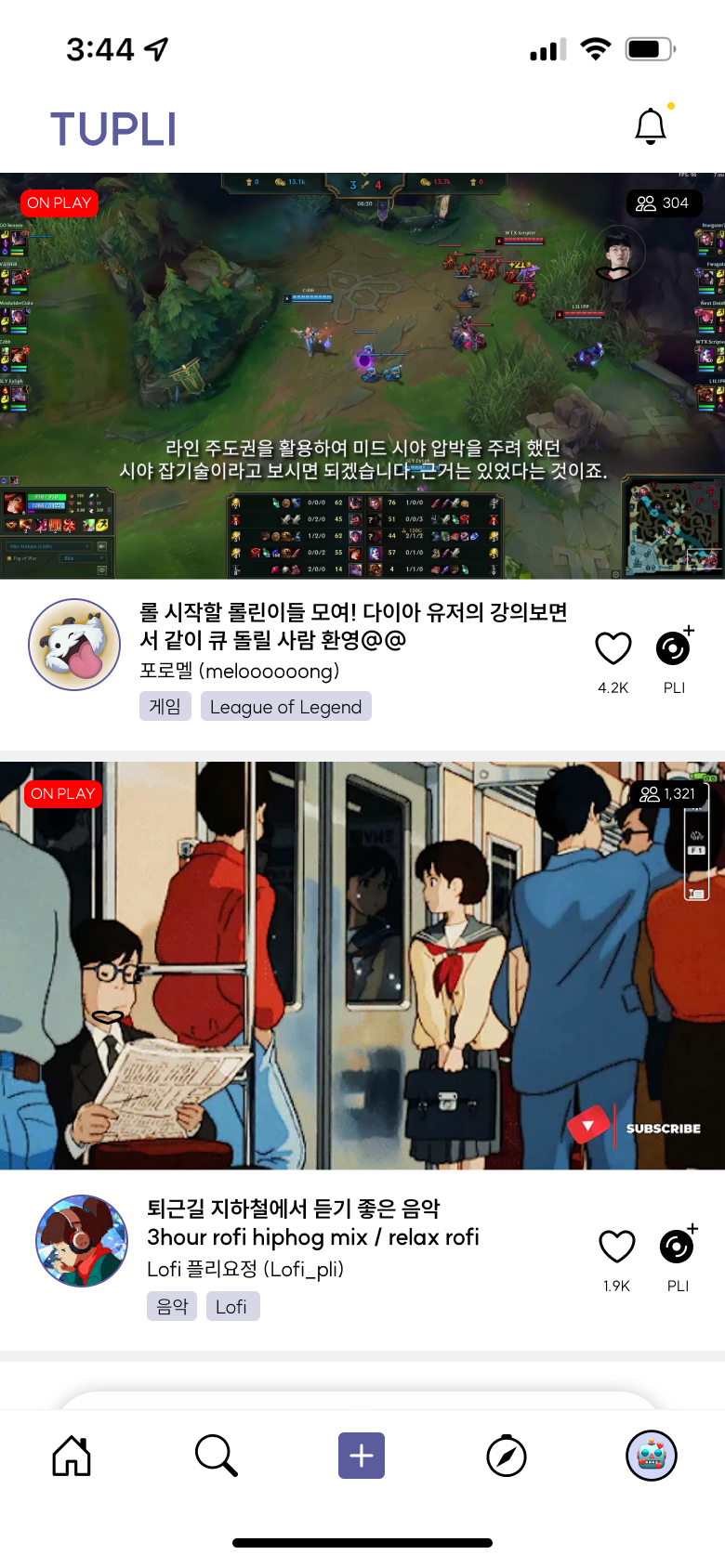
* **승인/거절 시 사용자에게 피드백 노출**

누군가에게 승인을 받아야 하는 입장이 된다던가, 거절당한다는 불쾌한 경험을 막기 위해서 요청 탭을 작성하지 않았습니다. 특정 유저를 팔로우하거나, 맞팔로우 하였을 때, 알림을 보내어 유저에게 피드백을 표시합니다.

### 헤더 및 네비게이션

**<Req. 5-1 헤더>**

* **기본 헤더 구성: 검색 버튼, 네비게이션, 페이지명, 뒤로가기 버튼**

****TUPLI 웹사이트는 모바일에 친화적인 서비스이기 때문에, 헤더를 구성하는 파트를 화면의 하단에 배치하였습니다. 홈버튼, 검색버튼, 만들기버튼, 탐색버튼, 프로필버튼이 있는 네비게이션 바를 하단에 배치하였습니다. 뒤로가기 버튼은 필요한 경우 상단 좌측에 표시됩니다. 또한, 안드로이드 스마트폰이라면 네비게이션바의 뒤로가기를 통해 뒤로가기 버튼을 이용한 효과를 볼 수 있습니다.

<그림6. 하단의 네비게이션 바>

* **뒤로가기 버튼 클릭 시, ‘직전 페이지’로 이동**

기본적으로 히스토리 모드를 사용하기 때문에, 브라우저에서 제공하는 뒤로 가기 버튼 클릭시 직전 페이지로 이동할 수 있도록 구현하였습니다.

좌측 상단에 표시되는 ‘뒤로가기’ 버튼을 클릭하면 직전 페이지로 이동할 수 있도록 구현하였습니다. 뒤로가기가 필요하지 않은 페이지의 경우엔 특정 페이지로 redirect 되는 경우도 존재합니다.

또한, 직전 페이지가 무엇임에 따라 기존 검색 결과가 남아있게 한다던가 하는 식으로 유저의 불편함을 줄이고 같은 검색을 하거나 로딩이 느려질 수 있는 문제를 해결하였습니다.

### 검색 기능

**<Req. 6-1 유저 검색>**

* **검색 버튼 클릭 시 최근 검색한 닉네임을 최신순으로 노출**

검색 버튼을 클릭하고, 계정 탭으로 이동할 시, 최근 검색한 검색어를 최신순으로 노출시키도록 하였습니다.

* **유저 닉네임 검색 시 검색어와 유사한 값을 자동완성으로 노출**

검색칸에 키워드 입력 시 존재하는 유사한 유저 닉네임 뿐만 아니라 플레이리스트 제목, 플레이룸 제목, 영상 제목을 자동완성으로 노출시킬 수 있도록 구현하였습니다. 또한, 각각 항목에 대해 탭을 만들어서 따로 검색할 수 있게 하였습니다.

* **유저 닉네임 검색 시 해당 닉네임으로 검색된 유저를 검색 결과로 노출**

검색하고 싶은 항목에 대한 탭으로 이동한 뒤, 키워드 입력 후 검색을 하면 해당 검색어를 포함하는 결과값을 출력합니다.

* **검색 결과에서 유저 클릭 시 해당 유저 페이지로 이동**

검색 결과에서 유저 클릭 시, 해당 유저 프로필로 이동합니다.

* **닉네임 등 사용자에게 있어서 중요한 정보는 강조하여 노출**

검색 결과 자체가 사용자에게 있어서 중요한 정보들의 모음이므로 따로 강조하여 노출시킨 파트는 없습니다.

### 뉴스피드 기능

**<Req. 7-1 게시글(피드)>**

* **유저의 닉네임, 피드의 게시일, 기입한 코멘트, 인용된 URL의 메타데이터 노출**

게시글(피드) 목록에 작성자의 닉네임, 피드 게시일, 코멘트를 노출하도록 구현하였습니다. 인용된 URL 메타데이터가 같이 노출되면 디자인상 불편하다고 판단하여 배제하였습니다.

* **좋아요 수, 댓글 수, 스크랩 수, 공유 버튼 노출**

게시글(피드) 목록에 좋아요 수, 댓글 수, 스크랩 수, 공유 버튼을 노출하였습니다.

* **좋아요를 획득한 피드는 ‘~님 외 ~명이 좋아합니다’ 문구 노출**

모바일 화면 디자인상 ‘~님 외 ~명이 좋아합니다’ 라는 문구는 너무 길고 어울리지 않다고 판단하였습니다. 좋아요를 획득한 피드는 하트가 채워지며, 좋아요 수가 표시됩니다.

* **하트 버튼 클릭 시 좋아요 처리, 재클릭 시 좋아요 취소**

게시글의 좋아요 버튼을 클릭하면 해당 게시글에 좋아요 처리가 되고, 다시 클릭하면 좋아요가 취소되도록 구현하였습니다.

* **댓글 버튼 클릭 시 댓글 창으로 이동**

게시글 덧글 보기 버튼을 클릭하면 해당 게시글에 달린 덧글을 확인할 수 있습니다.

* **스크랩 버튼 클릭 시 스크랩 처리**

게시글에 관련한 기능을 최소화 하여 스크랩 기능은 제외하였습니다. 게시글에 대한 기능은 좋아요 기능만으로 충분하다고 판단하였습니다.

* **좋아요, 스크랩 버튼 클릭 시 사용자에게 피드백 노출, 피드백은 프론트엔드에서 우선처리.**

게시글에 대해 좋아요, 스크랩 버튼을 누르면 프론트엔드에서 먼저 처리되어 사용자에게 빠른 체감을 줄 수 있도록 하였습니다.

* **공유 완료 시 사용자에게 피드백 노출**

게시글을 스크랩 하는 기능은 기능의 단순화를 위해 제외하였습니다.

* **팔로잉 유저가 게시한 피드의 메타 데이터를 최신순으로 5개까지 노출**

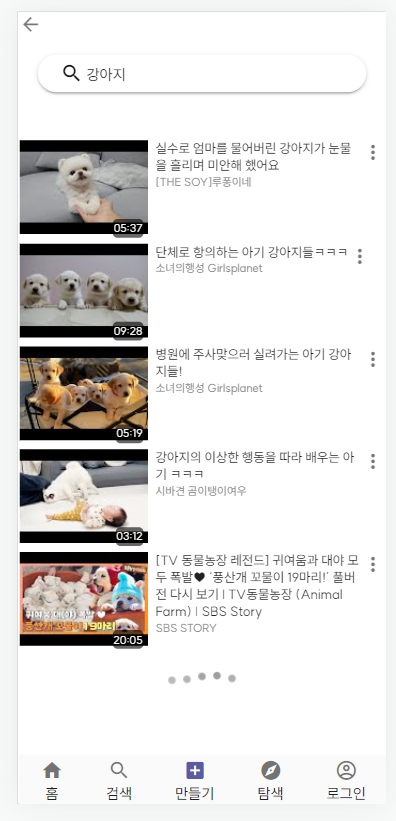
뉴스피드 목록에서 팔로잉한 유저가 작성한 글은 최신순으로 최대 5개까지 노출할 수 있도록 구현하였습니다.

* **팔로잉 유저가 게시한 피드를 최신 순으로 노출**

뉴스피드 목록에서 보여지는 피드 또는 플레이리스트는 팔로잉한 유저가 작성한 글을 우선으로 노출시키고, 최신 5개의 피드를 전부 확인하고 나면 관심있는 분야를 중심적으로 노출시키도록 구현하였습니다

* **피드들은 무한 스크롤 적용, 무한 스크롤 적용 시 로딩되는 동안 멈춘 것처럼 느껴지지 않도록 유의**

뉴스피드 목록을 보기 위해 하단으로 스크롤 할 때, 로딩된 데이터 량의 1/3만큼 읽었다면 다음 노출시킬 데이터를 미리 요청하여 로딩되는 동안 멈춘 것처럼 느껴지지 않도록 구현하였습니다.

**** <그림7. 무한 스크롤 적용 화면>

### 댓글 기능

**<Req. 8-1 댓글 기능>**

* **댓글 리스트 노출, 유저 닉네임, 댓글 내용 노출**

덧글 리스트에 접속하면 덧글을 작성한 유저의 닉네임, 그리고 내용을 노출하도록 합니다.

* **댓글 작성 기능**

게시글 하단에 덧글을 작성할 수 있는 칸이 있습니다. 입력 칸에 작성하고 싶은 내용을 입력한 뒤 ‘게시’ 버튼을 누르면 게시글에 덧글을 달 수 있습니다.

* **댓글 삭제: 본인의 댓글 삭제 가능**

자신이 작성한 댓글은 삭제할 수 있는 버튼이 노출되고, 그 버튼을 누르면 덧글을 지울 수 있습니다.

* **댓글 작성 중 유저가 뒤로가기 클릭 시 작성 중단 여부 확인**

덧글 작성시에 뒤로가기 클릭 시, 작성 중단 여부를 확인하는 기능은 사용자에게 너무 많은 alert를 표시하고 불편함을 유발할 수 있을 것으로 판단하여 제외하였습니다.

* **댓글 작성 시 데이터가 입력되었을 때만 ‘작성’버튼 활성화**

아무 내용도 덧글 입력 칸에 담지 않았을 경우에는 ‘게시’ 버튼이 활성화되지 않도록 하여 아무 내용이 없는 글을 업로드하지 못하도록 막았습니다.

* **댓글 게시 후 사용자에게 피드백 노출**

덧글을 게시하고 나면 자신이 게시한 글을 바로 확인할 수 있도록 자신이 덧글을 작성한 게시물의 상세화면으로 이동합니다.

* **본인 외 댓글의 삭제는 불가, 문제가 되는 댓글은 숨김으로 처리**

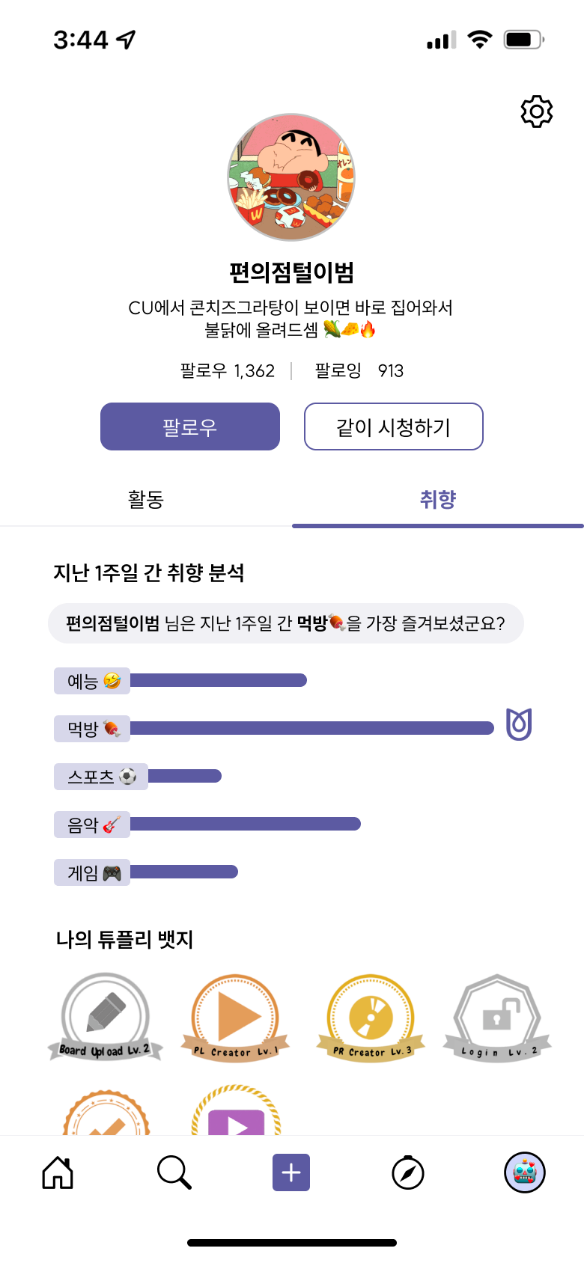
자신의 덧글이 아닌 경우는 ‘삭제’ 버튼이 노출되지 않습니다. 게시글 주인은 문제가 되는 덧글을 신고할 수 있습니다. 신고된 메일은 관리자가 확인하고 처리합니다.

* **댓글 삭제 후 직전 행동(삭제)을 취소할 수 있도록 피드백 노출**

덧글은 내용이 많지 않고 삭제를 하더라도 게시글을 삭제하는 것보다 비교적 복구가 쉬운편이라고 생각하였고, 삭제 후 삭제 취소 버튼을 노출시키는 것이 UX 체감상 깔끔하지 못하다고 판단하여 구현하지 않았습니다.

## **SubPJT III**

### 가) 유저 페이지 (추가)

**<Req. 1-1 유저 페이지(본인, 타 유저 모두 해당)>**

<그림8. 유저 프로필 화면>

* **유저 페이지에 본인이 게시한 피드들을 목록으로 나열하여 노출**

본인 프로필 화면에 접속하면 ‘활동’ 과 ‘취향’ 탭이 나뉘어져있는데, ‘활동’ 탭을 클릭하면 자신이 게시한 피드들을 확인할 수 있습니다.

* **유저 페이지에 타 유저의 피드를 좋아요 한 것을 목록으로 노출**

프로필 화면에서 ‘설정’ 페이지로 이동한 뒤, ‘좋아요한 게시물’을 클릭하면 자신이 좋아요 한 게시글의 목록을 볼 수 있습니다.

* **유저 페이지에 자신의 큐레이션 세트를 목록으로 노출**

프로필 화면에서 자신이 저장한 플레이리스트, 플레이룸, 게시글을 따로 모아서 확인할 수 있습니다. 또한 좋아요를 누른 플레이리스트, 플레이룸, 게시글을 볼 수 있습니다.

* **각 탭 전환 시 해당 탭이 선택되어 있음을 강조**

선택된 탭은 색이 칠해져 있어서 사용자는 어느 탭의 화면을 보고 있는지 확인할 수 있도록 하였습니다.

### 나) 큐레이션 세트 설정

**<Req. 2-1 큐레이션 세트 설정>**

* **유저가 추가한 다수의 키워드를 하나의 세트로 묶어 이름을 지정하여 관리**

유저가 플레이리스트나 플레이룸을 작성할 때, 해당 플레이리스트나 플레이룸의 태그를 직접 입력할 수 있습니다. 이 태그를 통하여 추후 같은 태그를 사용하거나 설정된 플레이리스트, 플레이룸을 검색할 수 있습니다.

조금 더 편한 키워드 입력 환경을 위해 TUPLI는 2022 인스타그램에서 자주 쓰이는 해쉬태그를 기반으로 한 태그들을 모아 추천해줍니다. 물론, 추천 태그를 사용하지 않고도 태그를 작성할 수 있습니다.

* **검색 대상 설정: 구글 커스텀 서치 및 RSS 검색 대상 설정**

플레이리스트, 플레이룸을 작성할 때 사용하는 유튜브 영상을 Youtube DATA API를 통해Search할 수 있습니다. 이 기능을 구현하는데에 구글 커스텀 서치 또는 RSS 기능이 필요하지 않았습니다. 따라서 CSE와 RSS을 TUPLI에서는 사용하지 않았습니다.

* **삭제: 기존에 추가된 큐레이션 세트 혹은 개별 키워드를 삭제**

플레이리스트, 플레이룸 삭제 시 해당 플레이리스트, 플레이룸에 사용한 큐레이션 세트는 CASCADE하게 함께 삭제됩니다. 또한, 플레이리스트, 플레이룸의 정보를 수정할 경우 개별 또는 전체 키워드를 수정할 수 있습니다.

* **큐레이션 세트 이름 추가 및 변경 시 사용자에게 피드백 노출**

플레이리스트 및 플레이룸의 이름 또는 태그 등을 변경할 시, 사용자에게 ‘변경 적용되었습니다’ 라는 메시지를 노출시킵니다.

* **키워드만 입력해도 세트를 설정할 수 있도록 기본 검색 대상 제공**

플레이리스트 및 플레이룸을 생성하려고 할 때, 영상의 키워드만 입력해도 기본 분야 검색 대상이 제공되어 빠른 입력이 가능하도록 구현했습니다.

* **키워드 / 검색 대상 추가 시 사용자에게 피드백 노출**

영상을 큐레이션 하여 플레이리스트 작성 시, 영상의 키워드 / 검색 대상 추가 시, 하단에 선택된 영상의 썸네일과 제목, 그리고 업로드한 사람의 아이디가 보여지고 우측 버튼을 잡고 상하로 움직이면 업로드할 영상의 순서를 변경할 수 있습니다. 영상을 선택한 뒤 최하단의 삭제 버튼을 누르면 추가되었던 영상이 삭제됩니다.

* **큐레이션 데이터 저장 시 데이터가 없을 경우 ‘저장’ 버튼 비활성화**

영상을 큐레이션 하여 플레이리스트를 작성할 시, 추가된 데이터가 없다면 ‘생성’ 버튼이 비활성화됩니다. 따라서 아무런 내용이 없는 플레이리스트를 생성할 수 없도록 하였습니다.

* **큐레이션 세트 / 개별 키워드 삭제 후 직전 행동 (삭제 등)을 취소할 수 있도록 피드백 노출**

Youtube API를 이용하여 큐레이션 세트에 추가한 뒤, 삭제하였을 경우 바로 완전히 삭제되지 않고 흐릿하게 남아있어서 다른 작업을 하기 전에 되살릴 수 있도록 하였습니다. 키워드를 삭제한 뒤에도 다른 작업을 하기 전에 되살릴 수 있습니다.

**<Req. 2-2 고급 검색 설정(Google Custom Search Engine)>**

* **고급 검색(Google CSE)에 필요한 정보를 다음 항목으로 설정 (CES 이름, CES ID, Google API Key)**

TUPLI 내에서 구현한 검색 쿼리로도 충분히 기능 구현이 가능하고, 오히려 내부 평가적 요소를 사용하기 때문에 다른 검색 엔진을 섞는 것은 방해된다고 판단하였습니다. 또한 CSE가 내세우는 특정 키워드를 이용한 모든 탐색 기능은 필요치 않기 때문에 CSE 기능은 사용하지 않았습니다.

**<Req. 2-3 고급 검색 설정 (RSS)>**

* **고급 검색(RSS)에 필요한 정보를 다음 항목으로 설정 (RSS 이름, RSS 주소)**

TUPLI의 서비스는 Youtube API를 이용하여 영상 데이터를 얻어오는 것 외에는 타 사이트에서의 컨텐츠 수집을 할 필요가 없다고 판단하였습니다. 따라서 RSS 기능을 추가하지 않기로 하였습니다.

### 다) 게시글 큐레이션 기능

**<Req. 3-1 게시글 큐레이션 실행>**

* **큐레이션 세트를 통해 게시글 큐레이션 실행 시, 로딩 화면 노출**

TUPLI는 태그를 통해서 영상들과 플레이리스트들을 분류하여 저장할 수 있습니다. 추후 해당 태그가 인기 키워드에 오르면 해당 태그를 가진 영상들과 플레이리스트가 홈 화면 상위에 노출됩니다.

* **스크랩 버튼 클릭시 피드를 ‘스크랩 리스트’에 보관**

게시글은 는 좋아요 표시를 하거나, 저장을 하면 자신의 프로필 -> 설정 에 들어가서 좋아요 표시를 했거나 저장한 게시글을 모아서 확인할 수 있습니다.

* **스킵 / 스크랩 버튼 클릭 시 사용자에게 피드백 노출**

게시글을 좋아요 하거나, 저장할 경우 메시지를 노출시켜 사용자가 인지할 수 있도록 하였습니다.

### 라) 큐레이션 리스트

**<Req. 4-1 스크랩 리스트>**

* **유저가 큐레이션 실행으로 스크랩한 피드와 타 유저의 피드를 스크랩한 게시물들을 보관한 리스트 노출**

내가 좋아요 한 게시물과 내가 팔로우한 유저가 좋아요한 게시물을 확인할 수 있습니다. 또한 본인이 만든 플레이리스트를 프로필에 게재함으로써 비슷한 취향을 가진 사용자를 찾고, 자신의 취향을 분석하고 추천할 수 있는 시스템을 구축하였습니다.

* **유저가 스크랩한 게시물이 없는 경우, 스크랩한 피드가 없다는 문구 노출**

무언가를 스크랩한 결과를 Object.keys(결과물).length로 확인하여 0일 경우(비어 있을 경우), 내용이 없다는 안내문을 TUPLI 사이트 전체에 걸쳐 안내해 주었습니다.

* **스킵 버튼을 누를 경우 스크랩 리스트에서 삭제하고 스킵 리스트로 이동**

스킵 리스트를 스크랩 리스트 외에 따로 구현하는 것은 또다른 스크랩 리스트를 만드는 것이라고 생각하였고, 스크랩한 피드를 다시 읽고 뒤로가기 버튼을 이용해서 되돌아 나오면 되는 부분이라고 판단하여 스킵 버튼은 구현에서 제외되었습니다.

* **스크랩 리스트에서 스킵 버튼을 누를 경우 사용자에게 피드백 알림**

스킵 버튼은 구현에서 제외되었습니다.

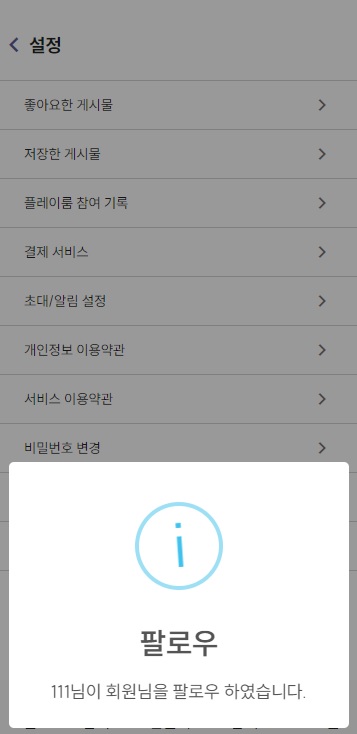
**<Req. 4-2 피드 발생>**

* **스크랩 리스트에서 게시물 선택하여 본인의 의견 입력 및 피드 발행**

주로 ‘좋아요’를 통해 모인 피드, 특히 플레이리스트와 일반 게시글에는 그에 관한 의견을 제시할 수 있게 댓글을 준비해두었습니다. 각각 피드의 성격이 다르기 때문에, 형태가 다른 댓글을 별도의 형태를 가지는 엔티티와 테이블로 구성하였습니다.

* **피드 발행 시 뉴스피드 및 본인과 팔로워 유저 페이지에 게시되며 팔로워에게 피드가 발행되었음을 알리는 알림 메시지 발송**

설정을 통해 알림 메시지 발송과 수신에 관한 범위를 설정하고, 특정 행위에 반응하여 실시간 알림을 포함한 알림을 발송하게 설정하였습니다.

**** <그림9. 알림 메시지를 수신하는 화면>

* **피드를 발행 완료하면 해당 게시물은 스크랩 리스트에서 삭제**

저장한 영상을 기반으로 플레이룸을 만들어도 재활용 가능성이 있기 때문에 삭제하지 않고 오히려 플레이룸에 해당 영상과 같은 정보를 가지는 내용을 복사하여 CASCADE하게 연결하였습니다. 이는 삭제와 수정을 편하게 하고, 수정, 삭제, 추가에 있어서 기존 정보에 종속적이지 않게 하는 효과를 가지고 있습니다.

* **긍정적인 액션인 ‘피드 발행’ 버튼은 강조하여 표시**

이 외에도 긍정적인 액션은 저희 TUPLI의 테마칼라를 이용하여 강조하였고, 부정적이거나 취소관련 버튼은 붉거나 희게 표시하였습니다.

* **댓글 작성 중 유저가 뒤로 가기 시 작성 중단 여부 확인**

가볍게 쓸 수 있는 모바일 SNS나 실시간 알림 등을 통한 환경을 고려하면 하던 일을 그만두고 다른 행위를 하는 경우는 충분히 있을 수 있다고 생각합니다. 댓글은 게시글과 다르게 그 내용이 길지 않기 때문에 오히려 말없이 삭제하고 이동하게 함으로써 작성 중단 여부를 확인하는 번거로움을 없앴습니다.

* **피드 발행 완료 후 사용자에게 피드백 노출**

발행 완료시, 게시글을 게재하였습니다. 등의 안내문을 노출합니다.

**<Req. 4-3 스킵 리스트>**

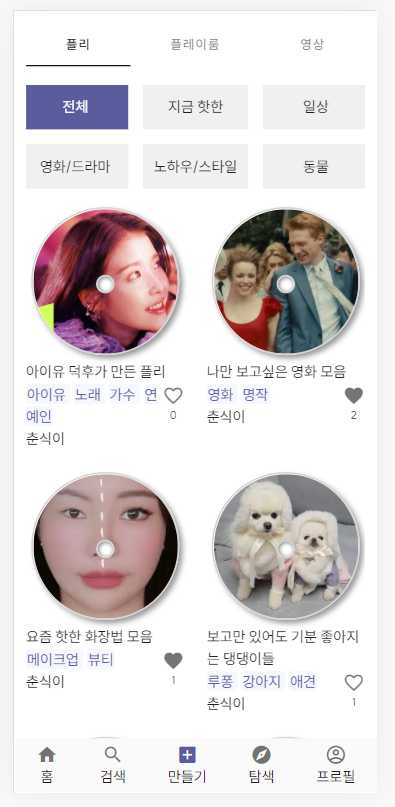
* **개인 취향 분석으로 스킵할 행위 자체를 줄이는 방향으로 노력**

유저의 행동(특정 취향의 유저를 팔로우, 특정 카테고리의 플레이리스트 공유, 특정 카테고리의 플레이룸에서 영상 시청 등)을 통해, 점차 유저의 취향을 분석해 나감으로써, 애초에 스킵 행위 자체를 하지 않을 수 있는 환경을 조성하고자 합니다.

### 마) 트렌드

**<Req. 5-1 트렌드>**

* **트렌드는 2개의 탭(피드, 키워드)으로 구성**

둘러보기 페이지에서 전체, 지금 뜨거운 정보 같은 공통 정보를 제외하고, 유저의 행동을 반영하거나 뜨거운 관심을 받는 키워드를 넷 선정하여 카테고리에 맞게 분류합니다. 키워드를 클릭하면 해당 키워드로 플레이리스트와 플레이룸이 검색됩니다. 둘러보기 페이지의 하단에는 인기 플레이리스트가 나열됩니다. TUPLI는 트렌디 컨텐츠를 피드와 키워드로 나누지 않고, 플레이리스트와 플레이룸, 그리고 동영상으로 나누었습니다.

<그림10. 트렌드와 분류를 적용한 구동 화면>

* **피드 탭: 게시된 피드 중 ‘좋아요’ 누적 획득 수 상위 10개를 노출**

TUPLI 는 게시글 피드 대신 플레이리스트 또는 플레이룸을 좋아요 순으로 나열하였습니다. 메인 화면에서도 트렌디한 키워드와 플레이리스트, 플레이룸을 우선으로 나열합니다.

* **각 탭 전환 시 탭이 선택되어 있음을 강조**

현재 보고있는 탭은 색이 칠해져서 사용자가 현재 어느 탭을 이용중인지 확인할 수 있습니다.