**2021 2nd Semester**

**Global Software Engineering**

**Final Report**

**너와 나의 연결고리 ( Linked list between U&I ),**

**효율적인 팀 협업을 위한 어플리케이션 개발**

**팀장** 32181771 박용관

**개발** 32184939 허찬용

32184893 한현민

**테스팅** 32197721 호준영

**문서 작성** 32183464 이재헌

**목차**

1. **개요**

* 아이디어 도출 과정
* 개발 목표
* 관련 시장 정보 탐색

1. **개발 내용 관련**

* 요구사항명세서(SRS) 작성
* 프레임워크 선정
* UML Diagram

1. **프로젝트 진행 전 논의사항**

* 프로젝트 일정 정립
* 결과물 예시 디자인
* 예상한 한계와 해결 방안
* 역할 분담
* 테스팅 방안에 대한 논의

1. **실제 프로젝트 진행 후**

* 실제 결과물
* 테스트 결과
* 참고 자료
* 프로젝트를 마무리하며

**1. 개요**

- 아이디어 도출 과정

코로나 사태로 우리의 사회 양상은 급격히 달라지기 시작했다.  
여러 사람이 모여서 하는 스터디 그룹, 공모전과 같은 활동들은 사람을 구하기도,  
적합한 장소를 선정하여 진행하는 과정에서도 모두 난항을 겪게 되었다.

우리의 프로젝트는 스터디 그룹과 공모전을 필요로 하는 학생들로 하여금  
사람들을 찾고, 팀을 구성하여, 학습을 진행하는 일련의 과정에서  
최대한 편리하고 체계적인 도움을 주고자 하였다.

현재 코로나 사태의 비대면 스터디나, 추후 사태가 종식된 경우에도 어플이 효용성을 잃지  
않도록 대면으로 프로젝트를 진행할 수 있는 기능 또한 제공하며,  
개개인의 관심 분야에 맞는 사람들을 직접 찾아 팀원을 매칭시켜주고, 일정을 수립하고,  
공동의 목표를 위한 토대를 제공하는 것이 어플의 궁극적 목표이다.  
이러한 어플을 만들기 위해  
여러 분야의 유용한 정보가 담긴 게시판, 팀 간 공지사항들, 프로젝트 진행 상황,   
팀원들이 공유하는 게시판, 일정 조정과 수립을 위한 캘린더, To-do List와 같은 기능들을  
구현 및 제공하여 현재의 사회적 상황과 관계없이 학생들의 학업에 선한 영향력을 주고자 한다.

처음엔 수업의 일환으로 진행하는 프로젝트라 생각하여 크게 의미를 두지 않았으나,  
결국 해당 프로젝트를 진행하는 우리 팀 또한 협업을 필요로 하고, 실제 협업을 진행하면서  
코로나 사태가 공모전과 스터디 그룹, 프로젝트에 끼치는 영향이 생각보다 더 크다는 것을  
몸소 체감하면서 해당 프로젝트에 더 몰입할 수 있었다.

- 개발 목표

개발 목표는 해당 어플을 구현함에 있어 가장 우선순위로 생각해야 할 것들을 기반으로  
어플이 제공해야 하는 핵심 기능들을 간추려 놓은 것이다.

* 모임 내 인원, 즉 팀원이 될 사람들을 쉽고 빠르게 구할 수 있게 한다.
* 다양한 종류의 모임을 만들 수 있게 한다.
* 팀장이 팀을 편리하게 관리할 수 있는 서비스를 제공한다.
* 직관적인 UI를 통해 사용자가 사용함에 있어 불편함이 없도록 한다.
* 팀 프로젝트를 진행하면서 필요한 기능들을 구현함으로써 프로젝트를 체계적으로 진행할 수 있는 서비스를 제공한다.

- 관련 시장 정보 탐색

어플은 그 특성상 시장에 이미 나와있는 다른 어플이 제공하는 기능들을 똑같이 제공하는 경우가 대다수 있고, 기능적으로 큰 차이가 없지만, 어플 간 UI 디자인, 다른 어플과 달리 해당 어플만이 제공하는 특수한 UX (사용자 경험) 등을 통해 시장에서 경쟁력을 갖추는 경우가 많다.

우리가 프로젝트를 통해 개발한 어플을 시장에 출시할 계획은 아니지만,  
그래도 시장에 제공되고 있는 기성 어플 여러 개를 참고하여, 어플 각각의 특징과,   
유사점과 차이점 등을 비교 및 분석하여 어플의 방향성을 정립하고자 하였다.

1. 텍스트이(가) 표시된 사진

   자동 생성된 설명캠퍼스픽

캠퍼스픽의 경우, 우리가 이미 이용하고 있는 대학생 커뮤니티 어플 ‘에브리타임’의   
연동 어플리케이션으로, 대학생들이 각종 공모전 일정과 대외활동, 채용일정 등을   
공유하여 다양한 스터디 그룹과 동아리 모집 게시판, 공모전 모집 게시판 등이 있어  
대학생들 사이에선 필수 어플리케이션으로 불릴 정도로, 커뮤니티를 사용자가 관심사별로  
직접 생성 가능하며 다른 이용자와 소통할 수 있다는 특징이 있다.

1. 아이캠펑

아이캠펑의 경우 공모전, 취업정보 등 대학생에게 필요한 각종 대외정보를  
사용자에게 맞춤형으로 제공하고, 비슷한 관심사를 가진 다른 사용자들과  
소통할 수 있는 커뮤니티 기능을 제공한다.

해당 두 어플 모두 주 사용자를 대학생으로 지정하여 개발한 어플로,  
대학생 사용자 개인에게 유용한 정보를 다량으로 제공하지만,   
사용자가 어플 내부의 기능으로 팀을 찾고, 공모전과 대외활동을 준비하는 프로세스는  
없는 것으로 확인되어, 사용자 개인에게 초점을 맞춘 것이 아닌 다수의 사용자 그룹에 초점을 맞춘 어플을 개발하면 괜찮겠다는 결론을 도출했다.

**2. 개발 내용 관련**

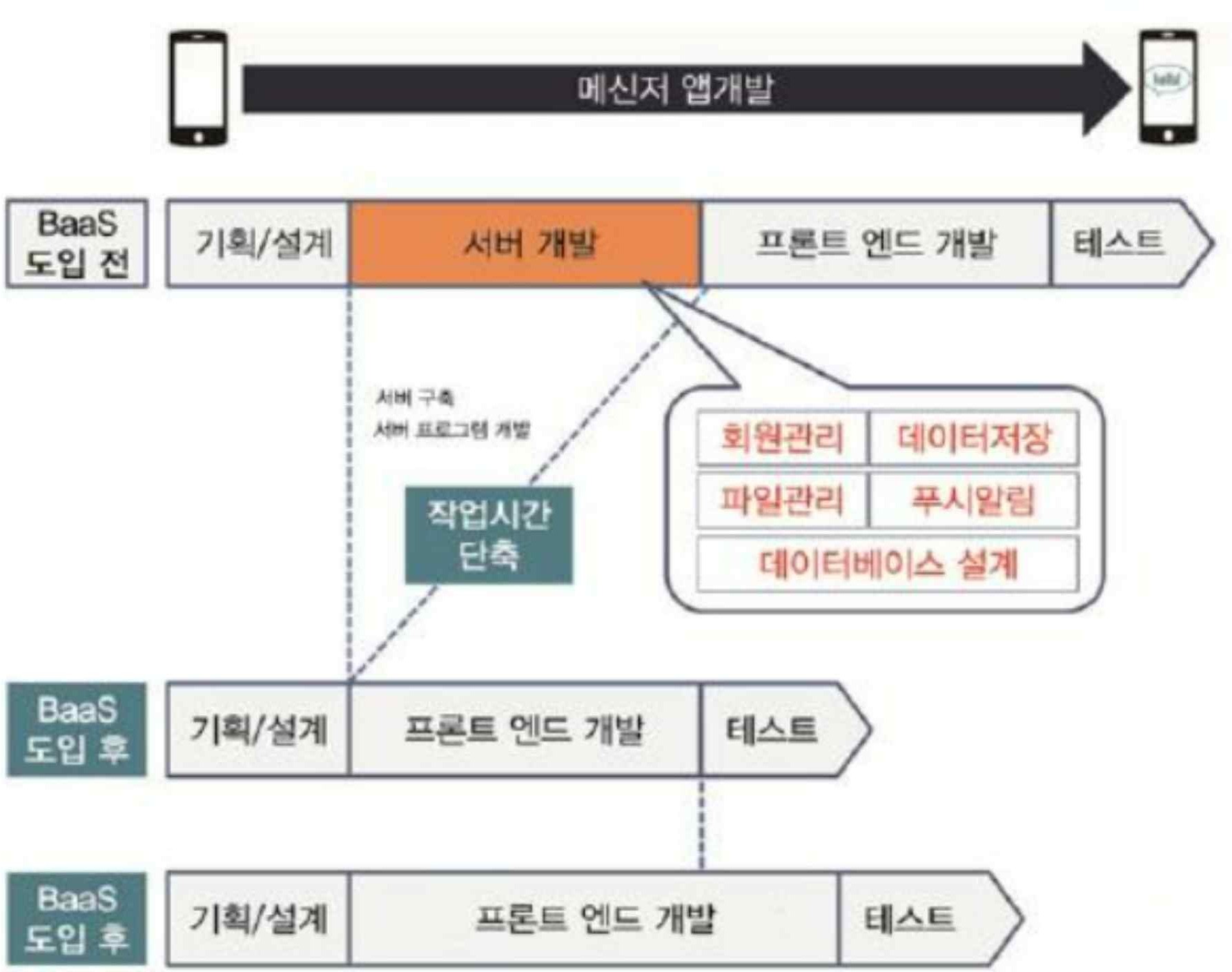
- 요구사항 명세서(SRS) 작성

테이블이(가) 표시된 사진

자동 생성된 설명서비스별 필요한 기능들과 그에 대한 세부적인 상세 기능을 작성하고, 중요도를 표시하여 개발 우선순위를 정했고, 특이사항 등을 작성할 수 있는 비고란으로 구분하여  
체계적인 개발을 하고자 하였고, 개발 과정 중 가장 많이 참조할 자료이자,  
추후 테스팅을 시행할 때의 기준점으로 삼아야 하는 자료기에 가능한 자세히 서술했다.

1) 요구사항 명세서를 토대로 구현한 부분에는 초록색 표시, 부분적으로 구현된 부분은  
주황색 표시를 통해 진행사항을 확인할 수 있게 하였다.

2) 개발 인원들에게 서비스 별로 구분하여 각자 개발할 파트를 배분하였고,  
추후 테스팅 과정도 모두 이 요구사항 명세서를 기반으로 하여 작성하였다.

- 프레임워크 선정

어플 개발은 개발하는데 있어, 사용자들이 시각적으로 확인할 수 있는, 아주 깔끔한 레이아웃을 가진 인터페이스를 프론트 엔드라고 부른다. 프론트 엔드 개발은 주로  
사용자 인터페이스(UI), 사용자 경험(UX)를 만드는데 초점을 맞추고 있고, UI를 디자인하는데 있어서는 실제 프로토타입을 통해 어플의 UX 안에 있는 이슈들을 확인하고,   
문제를 해결할 수 있다.

또한 사용자들의 정보를 담은 데이터베이스와 프로세스와 관련된 서버 측을 관리하는,  
사용자가 눈으로 확인할 수 없는 백 엔드 영역이 있다.

처음 우리가 프로젝트에 착수했을 때, 프론트 엔드 개발 언어로는 안드로이드와 iOS 두가지   
운영체제를 동시에 개발할 수 있는 크로스 플랫폼 환경이며, 브라우저 실행도 지원하는 네이티브 성능, 각종 확장 기능을 제공하는 Dart 언어 기반의 Flutter 프레임워크를 선정했으나,  
시간적 한계로 인해 외부 API들과 소스코드 들을 참조해야 하는 학생의 입장에서,  
Flutter 기반의 어플이 생각보다 수가 많지 않고, 개발 과정에서 참고한 어플 역시   
Flutter 기반이 아닌 Android Studio를 프레임워크로 하는 어플이고,   
Flutter 기반으로 개발을 진행할 경우 코드를 고칠 경우 새로운 버전을 배포해야 한다는 가장 큰 단점을 이유로, 이미 Stack Overflow 등과 같은 개발자 커뮤니티에서 아직까지도 활발히 사용되고 있고, 현재 수강중인 강의에서 도움을 가장 크게 받을 수 있는 “Kotlin” 언어를 기반으로 한 Android Studio 개발 환경에서 개발하기로 결정했다.

Android Studio는 Android의 공식 IDE로, Android용으로 제작되어 개발을 가속화하고,  
풍부한 코드 편집, 디버깅, 테스트 및 프로파일링 도구를 비롯한 맞춤형 도구를 제공한다.

Android Studio에서 어플의 프론트 엔드 개발에도 Java 언어와 Kotlin 언어를 사용할 수 있지만,

Kotlin언어가 Java에 비해 가진 다음과 같은 장점을 이유로 Kotlin을 선정하였다.

1. Java로 개발하는 것보다 비약적으로 코드의 양이 줄어들고,  
   Null-Pointer exception에 대한 안정성이 Java에 비해 매우 높다.
2. Kotlin은 인텔리제이(IntelliJ)와 강력하게 통합되어, 디버깅을 하는데 있어 불편함이 적다.
3. 문법이 간결하고, 풍부한 기능과 높은 안정성을 토대로 높은 생산성을 이끌어낼 수 있다.

C, C++, Java, Python과 같은 기초 프로그래밍 언어와 더불어

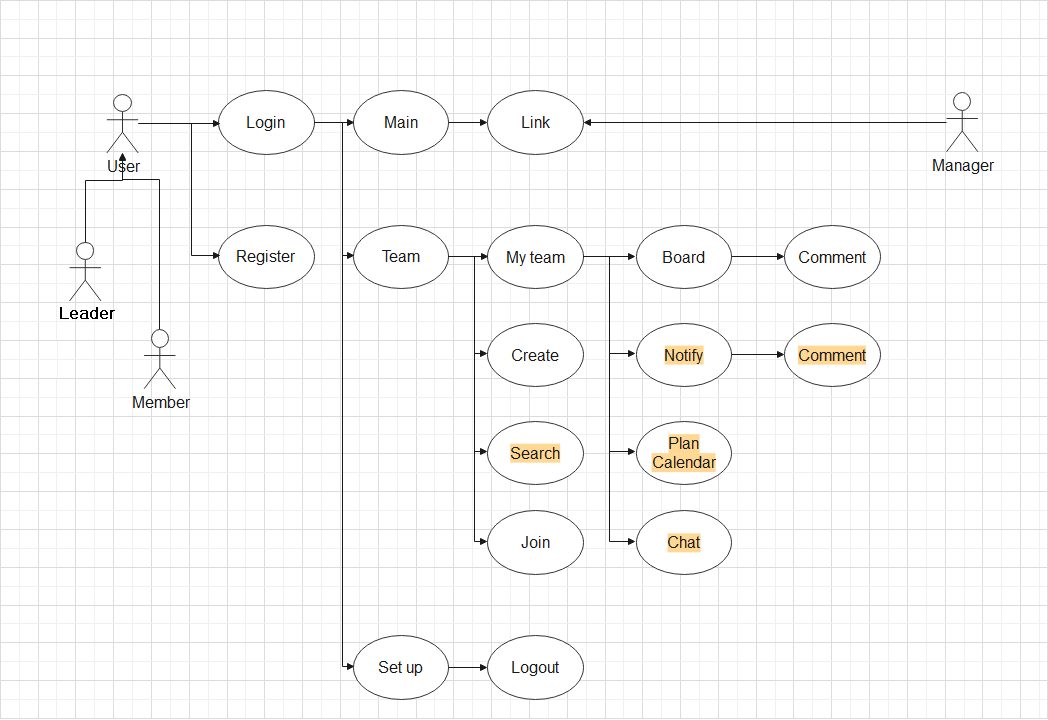
Data structure 와 같은 기초 전공 지식들만 습득하고 있었던 학부생 단위의 프로젝트 팀이기에, 처음 다뤄보는 어플리케이션 개발 도구는 익숙하지 않을 뿐 아니라, 해당 프로젝트에 사용되는 언어 역시 Android Studio에서 지원하는 언어들 중 Java가 아닌 Kotlin 언어이기 때문에, Kotlin 언어의 체계 또한 이해하고, 적용할 수 있을 정도의 수준이 필요하고. 또한, Android Studio 프레임워크의 구조와, 프로젝트 디렉토리의 파일들이 어떻게 동작하는지에 대한 동작 원리를 익힐 필요성 또한 존재, 이러한 학습은 시중에 나와있는 Kotlin 언어를 이용한 서적을 참고하며, 온라인 사이트 중 Stack Overflow, Github와 같은 개발자 사이트들 역시 이용했다.

또한 해당 프로젝트는 단순히 어플의 프론트 엔드 디자인만 진행하는 것이 아닌,  
어플 내부에 저장될 정보들을 담아두고 관리해야 할 데이터베이스를 요구한다.  
이러한 데이터베이스를 관리함에 있어 우리는 Google에서 제공하는 Firebase를   
백 엔드 프레임워크로 선정하였다.  
Firebase는 자체적으로 Database, Storage, Auth, Analytics까지 제공해주기 때문에, 기본적으로 서비스를 구축하기 위한 서버 인프라, 서버 세팅을 고려하지 않고, 클라이언트단의 작업만 제작하여 서비스를 진행할 수 있다.

기존의 Realtime Database에서 사용하던 기능을 그대로 가져와 채팅과 같은 사용자 간   
실시간으로 데이터를 공유해야 하는 상황에 적합하며, NOSQL 데이터베이스를 지원하여  
서버 구조를 미리 구축할 필요 없이, 클라이언트 개발을 진행하면서 즉각적으로   
구조를 제작 및 변경할 수 있다는 장점이 있다.

- UML Diagram

UML Diagram은 통합 모델링 언어(UML)을 사용하여 시스템 간 상호 작용, 시스템의 구조 등을 시각적인 자료로 표현한 도면이다.  
어플 개발에 있어 시스템과 외부와의 상호 작용을 묘사하는 Use case Diagram을 통해  
사용자와 관리자 간 주고받는 상호작용을 묘사하는 게 적합하다고 판단,  
Use case Diagram을 통해 일련의 시퀀스를 이해하기로 하였다.



사용자는 크게 Leader(팀장) 과 Member(팀원}으로 구분되고,  
계정이 있는 사용자는 바로 로그인을 하여 서비스를 이용할 수 있고,  
계정이 없는 사용자는 회원가입을 통해 서비스를 이용할 권한을 부여 받는다.

어플의 초기 화면에선 관리자가 제공하는 유용한 정보들을 담은 Link를 이용할 수 있고,  
Team 화면을 통해 팀을 만들거나, 검색하거나, 참여할 수 있고,  
팀에 소속된 사용자들은 게시판과 공지사항, 채팅 시스템 등을 통하여 소통할 수 있다.

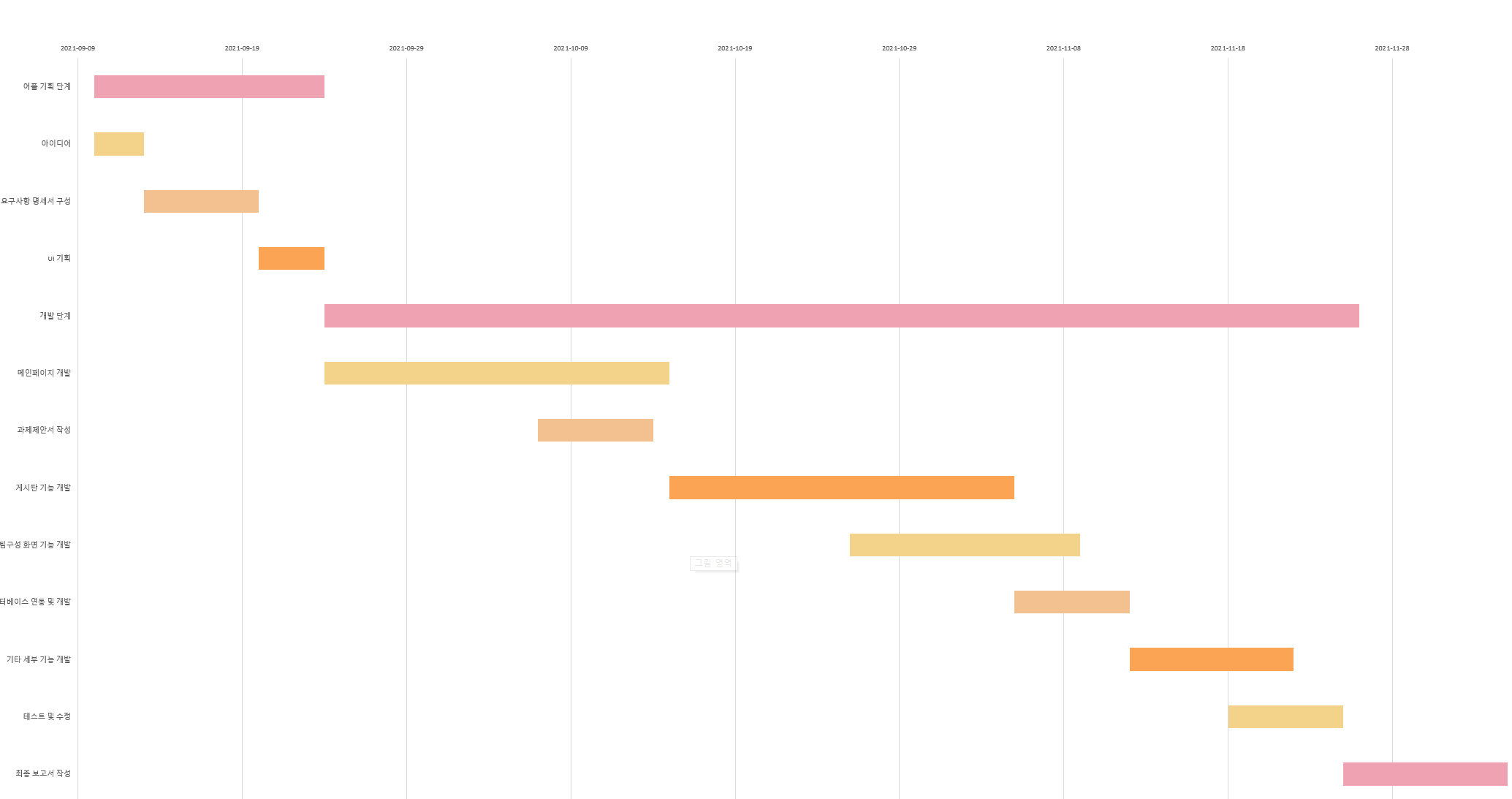
※ 개발 과정에서 요구사항 명세서에 작성된 우선순위를 기반으로 개발을 진행했으나,  
시간적 및 기술적 한계로 UML Diagram에 노란색으로 표시된 부분들은 구현하지 못했다.  
**3. 프로젝트 진행 전 논의사항**

- 프로젝트 일정 정립

일반적인 어플 개발은 총 개발 예산 = 개발 비용 + 유지 보수 비용으로 계산하고,   
이 외의 모든 비용을 포함하여 1,000만원 ~ 3,000 만원의 예산을 요구로 하고,  
개발 과정에서 기획자, 디자이너, 개발자 등 체계화된 조직과 프로세스를 통해 개발을  
진행하며, 일반적으로 2년 내외의 상당히 긴 수행기간을 통해 어플을 개발한다.

하지만 우리의 프로젝트는 0원에 가까운 예산과, 다섯 명의 소수 인원으로, 3개월 내지의  
짧은 프로젝트를 다른 학업들과 병행하며 진행해야 하는 시간적, 기술적 제약도 있었기 때문에  
최대한 탄력적인 일정을 계획하고 진행했다

결과적으로 12월 초, 기말고사 시험들을 대비하기 전까지 탄력적인 일정 진행이 가능했다.

간트 차트(Gantt Chart)를 통해 9월 팀 생성 이후, 어플 기획 단계부터 프로젝트가 종료되어  
최종 보고서 작성까지의 일정을 나타내면 다음과 같다.

테이블이(가) 표시된 사진

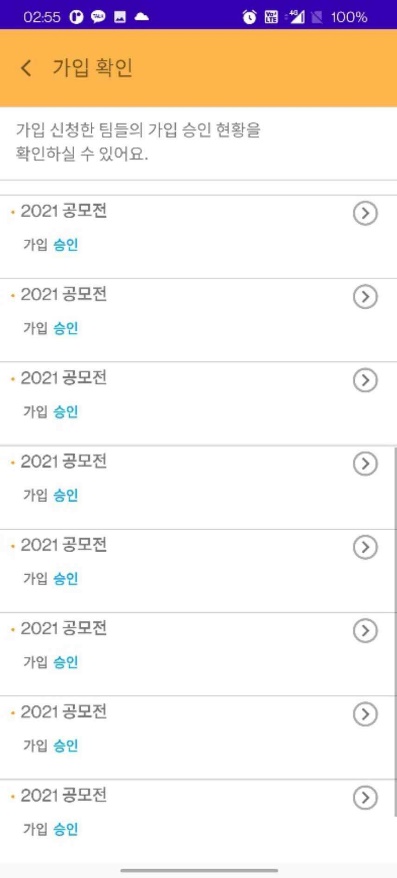
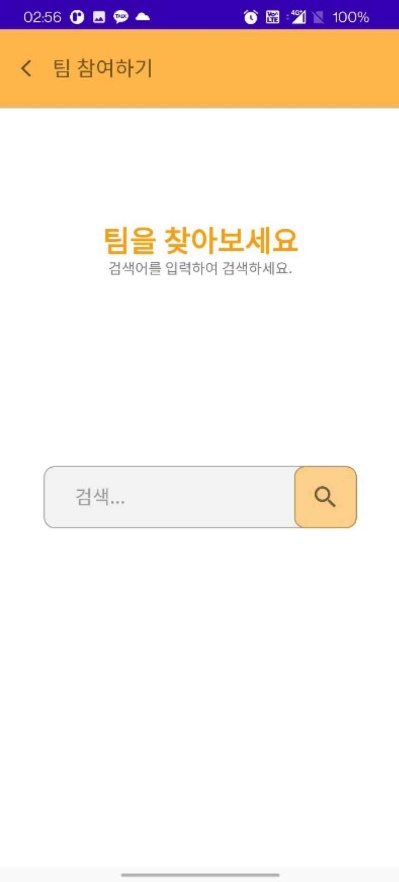
자동 생성된 설명

예상했던 대로 개발 단계에서 총 63일이라는 긴 시간을 소요했고, 그 중에서도  
많은 기능을 요구하는 메인 페이지와 게시판 기능을 구현하는 데 시간이 오래 걸렸다.

예상 외로 사용자들의 데이터를 관리하는 데이터베이스 프레임워크 Firebase가  
다양한 기능들을 제공하여, Android Studio와 Firebase를 연동만 시켜주면  
Firebase 내부에서 데이터들을 자동으로 처리해주는 프로세스 덕분에  
백 엔드 개발에서 시간을 아낄 수 있었다.

- 결과물 예시 디자인

해당 프로젝트는 실제 어플을 개발하는 프로젝트로, 요구사항 명세서에 작성된 내용들 뿐  
아니라 사용자가 어플을 실행했을 때 각 화면들 내부에 필요한 기능들과 위젯들, 레이아웃이  
적절한 위치에 배치되어 시각적으로 불편함이 없도록 실제 Android Studio를 통해 개발하기 전  
UI를 디자인하여, 추후 개발 과정에서 레이아웃들과 위젯 배치에 있어 보다 편리하고 빠르게  
개발할 수 있도록 하였다.



**예시 1 예시 2 예시 3 예시 4**

* + 예시 1: 사용자의 목적에 따라 팀 만들기/팀 참여하기/가입 확인하기

세 가지 기능 각각을 구현한 창으로 이동할 수 있다.

* + 예시 2: 자신이 만들고자 하는 팀의 이름, 해당 팀이 목표로 하는 활동 주제, 최대 활동 인원, 활동 기간 등을 설정하여 팀원을 모집할 수 있다.
  + 예시 3: 참여하고자 하는 팀을 검색어를 통해 탐색할 수 있다.
  + 예시 4: 사용자가 가입 신청한 팀들의 승인 여부를 확인할 수 있다.
* 해당 과정을 통해 위젯들을 화면 내에 배치하여 개발하는 Android Studio 개발 도구의 특성 상, 개발 과정에서 시간을 유의미하게 단축할 수 있었다.

- 예상한 한계와 해결 방안

상술한 문서 ‘프로젝트 일정 정립’ 부분에서도 일부 내용이 기술되어 있지만, 해당 항목에서는  
보다 상세한 한계와 그에 대해 논의한 구체적인 해결 방안, 결과적으로 어떻게 진행되었는지   
등을 서술한다.

**한계**

1. 시간적 한계  
   일반적인 어플 개발은 2~3년 정도의 긴 기간동안 개발을 진행하는데,  
   우리의 프로젝트는 3개월이라는 짧은 기간 동안 결과물을 도출해야 한다.  
   또한 해당 프로젝트만 진행하는 것이 아닌, 다른 학업 또한 병행해야 하므로  
   총괄적인 어플의 개발이 사실상 불가능했다.

또한 C, C++, Java, Python과 같은 기초적인 프로그래밍 언어를 익히고,  
자료구조, 알고리즘 등의 이론만을 학습하는 프로젝트가 아닌, 처음 다뤄보는 언어를 통해 개발해야 하는 문제에 맞닿아 있기 때문에 새로운 언어와 프레임워크를 익히고 학습하는 데에도 별도의 시간이 필요하므로, 우리의 예상보다도 시간이 촉박했다.

1. 기술적 한계  
   초기 개발에 있어 선정한 크로스 플랫폼 개발 도구인 Flutter의 경우  
   현재 네이티브 어플 개발자들이 마련한 토대가 부족하여, 외부 API와 오픈소스들을  
   참조해서 개발해야 하는 입장에선 매우 불리하다.

또한 프레임워크를 Flutter가 아닌 다른 프레임워크로 선정한다 하더라도,  
실제 어플을 만드는 과정은 처음 경험해 보는 것이기 때문에 다루는 언어, 함수,   
알고리즘 등에 대한 선수지식이 전무하다 해도 무방할 정도였기 때문에 이 부분에  
대해서도 추가적인 학습을 필요로 하였다.

1. 자원적 한계  
   처음으로 진행한 팀 단위의 프로젝트에서의 경험의 부재로 인해, 주제 선정 이후  
   개발 시작까지의 공백이 길었고, 일반적인 어플 개발의 프로세스와 달리  
   5인의 소규모 팀 단위로 결과물을 도출해내야 했기 때문에, 경험적 자원과  
   인적 자원의 부족으로 인해 우리가 바라는 실제 어플에 가까운 High-Quality의 결과물  
   을 도출해내기엔 한계가 있었다.

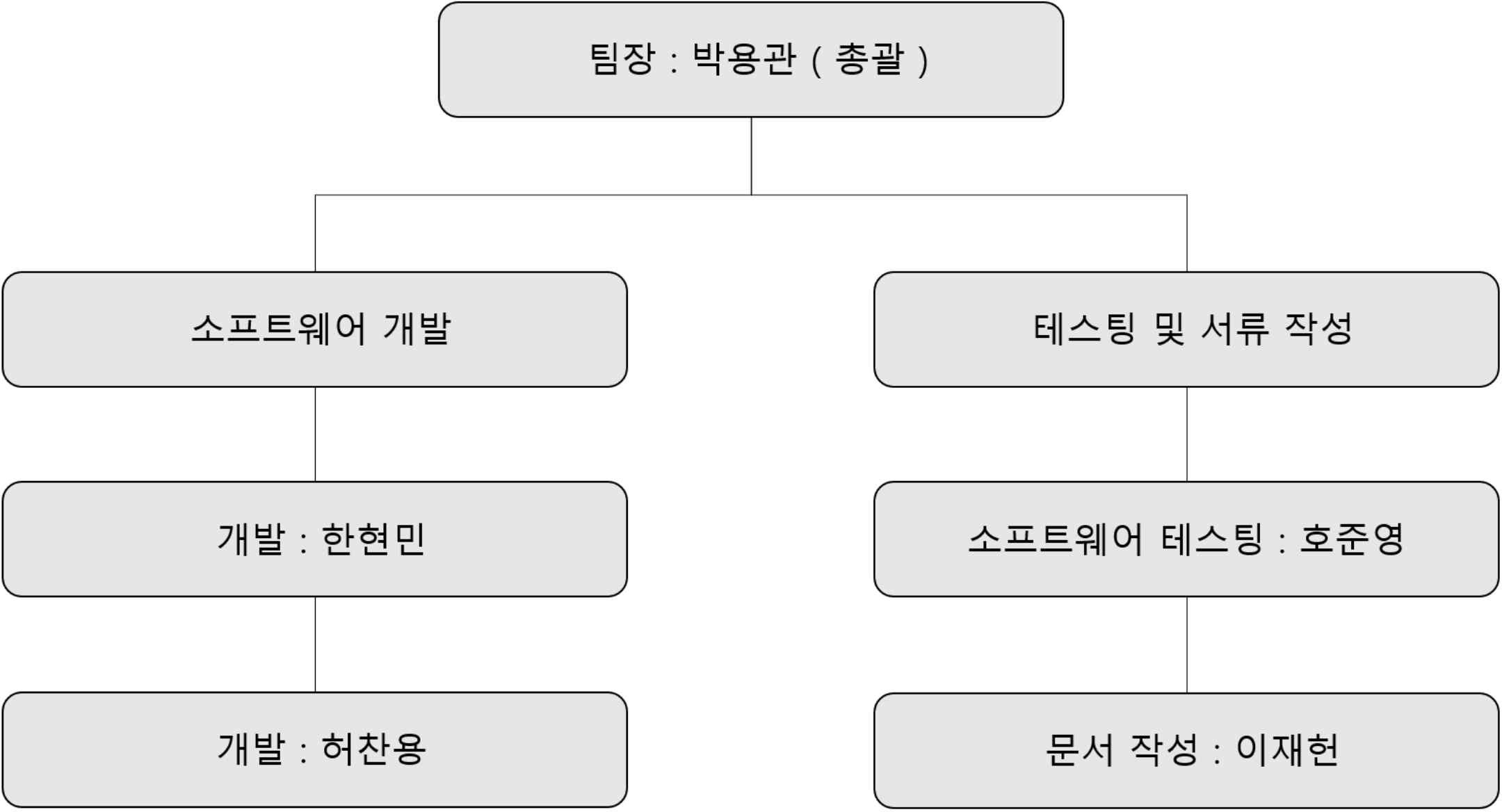
**해결방안**

1. 시간적 한계에 대한 해결 방안  
   탄력적인 일정 수립을 통해, 하루에 일정 시간을 개발에 꾸준히 투자함에 있어  
   부담감을 최소화하고, 시중에 나와있는 어플들과 같이 완벽한 결과물을 내기보다,  
   실제 기능들을 프로젝트를 통해 구현하는 연습을 하고, 이론적 배경을 접목시켜 학습을 하는데 우선순위를 둔다.  
   또한 개발 역할을 맡은 팀원들에 있어, 개발 과정에서 쓰이는 알고리즘들이나 문법 등을 서로 공유하며 같이 학습, 개발하는 것이 개발 속도를 촉진할 수 있을 것이라  
   판단하여, 학교와 지리적으로 가까운 위치에 사는 조원들을 개발 인원으로 선발하여,  
   일종의 스터디 그룹을 형성하여 같이 개발을 할 수 있도록 하였다.
2. 기술적 한계에 대한 해결 방안  
   Flutter와 같이 시장에 나온 지 오래되지 않은 프레임워크의 경우 초보 개발자들이  
   참고할 수 있는 API와 오픈소스들이 상대적으로 적고, Android Studio의 경우 이미  
   Java와 Kotlin 두 가지 언어로 제작된 어플 오픈소스 등이 널리 알려져 있는 경우가 많아, 개발 프레임워크를 Flutter에서 Android Studio로 전환하였고,  
   Android Studio에서 사용하는 Kotlin 언어와 Java 언어 중에서도  
   Kotlin의 경우 함수형 프로그래밍 언어로 Java와 유사한 특성을 가지고 있어 새로운 프로그래밍 언어임에도 적응하는데 어렵지 않을 것이라 판단하였고,  
   간단한 문법과 Kotlin 하나의 언어로 명령형, 객체지향형, 함수형 예제를 모두 다룰 수 있는 특성을 바탕으로 개발 코드의 수가 Java에 비해 비약적으로 줄어들어, 결과적으로 개발 피로도를 줄일 수 있다고 판단하였다.

또한 새로운 언어와 프레임워크를 다루는 과정에 있어,   
Stack Overflow와 같은 개발자 커뮤니티에서 개발 과정 막히는 부분들의 도움을 받고,  
GitHub에 제공된 오픈 소스들과, YouTube 공개 강의와 구글링을 통해 언어와   
프레임워크에 대한 선수지식들을 익히는 기간을 따로 설정하여 팀원들의 프로젝트  
이해도를 높이는 절차를 밟았다.

1. 자원적 한계에 대한 해결 방안  
   자원적 한계는 결국 시간적 한계와 일맥상통하는 부분이 있다고 판단하여,  
   소규모의 팀 단위인 만큼 High-Quality의 결과물을 도출하기엔 매우 어려운 상황임을 팀원들 스스로 인지하여, UI 디자인과 같은 부수적인 것들보다는 실제 기능들을 구현, 프레임워크에 대한 이해, 강의 시간에 학습한 이론적 배경을 바탕으로 프로젝트를   
   실제로 진행해보는 등의 중점적인 요소들에 초점을 맞추기로 했다.

이렇게 실제로 프로젝트 진행 중 있을 한계에 대한 논의를 진행하고, 해결방안을   
마련하여 진행한 결과, 훨씬 수월하고 탄력적으로 프로젝트를 진행할 수 있었다.

- 역할 분담

해당 프로젝트는 5인의 소규모로 진행되는 프로젝트이고, 대학 강의에 기반을 둔 점을 감안하여  
수행해야 하는 과제들을 크게 두 가지 분류로 나누어 진행하였다

1. 강의 시간에 배운 이론적 배경을 바탕으로 문서화 및 실습하는 과정 (테스팅, 서류 작성)
2. 실제 프로젝트 개발 (소프트웨어 개발)

프로젝트 진행과 보고서 작성이라는 하나의 큰 과제를 두 가지 분류로 나누어,  
팀원들 스스로의 역량과 비교하여 회의를 통해 역할을 배분한 결과,  
팀원 전원이 개발과 보고서 작성을 모두 참여하는 것은 비효율적이라 판단하여,  
실제로 팀 단위 프로젝트를 진행해 본 경험이 있고, 개발 쪽으로 진로 방향성을 잡아 개발의  
경험치를 쌓고자 하는 팀원들이 실제 소프트웨어를 개발하기로 하였고,

문서 작성 경험이 많고, 이론적 배경을 이해하고 프로젝트에 적용하는데 수월한 능력을 갖췄다고   
판단되는 팀원들이 개발한 내용에 대한 이해를 토대로 소프트웨어 테스팅과,   
과제 제안서 및 최종 보고서 등 서류 작성에 초점을 맞추기로 하였다.

또한 팀장의 경우 프로젝트 개발과 서류 작성 두 부분에 모두 경험이 다수 존재하여  
소프트웨어 개발과 테스팅 및 서류 작성 두 부분에 모두 피드백을 제공할 수 있는 사람을 선출,  
소프트웨어 개발과 서류 작성 두 분야에 모두 개입하여 프로젝트를 총괄하여 진행하였다.

- 테스팅 방안에 대한 논의

요구사항 검증(Requirements Validation) 에는 다섯 가지 검증 요소들이 존재한다

1. Validity: 시스템이 사용자의 수요를 뒷받침하는 기능을 제공하는가?
2. Consistency: 상충되는 요구사항들이 있는가?
3. Completeness: 소비자가 필요로 하는 모든 기능이 포함되었는가?
4. Realism: 요구사항들이 예산, 기술에 따라 구현이 가능한가?
5. Verifiability: 요구사항들이 검증될 수 있는가?

해당 다섯가지 항목들 중 4. Realism의 경우 배경적 한계가 있으므로 논외로 하여,  
요구사항 명세서에 작성된 소비자가 필요로 하는 모든 기능이 포함되었는지에 관한  
3. Completeness와, 어플 상에서 오류사항이 발생하지 않는지에 대한 5. Verifiability,  
두 가지 항목에 초점을 두고 테스팅을 시행하기로 하였다.

작성된 요구사항 명세서를 바탕으로 개발을 진행하기 때문에,  
Requirements review 검증 기법을 통해 시스템 매뉴얼을 분석하고, 개발한 어플 자체가  
완성된 프로젝트의 산출물이기 때문에 Prototyping을 통해 실행 과정에서  
오류사항이 있는지 또한 확인 가능하다.

또한, 어플은 그 특성 상 사용자가 직관적으로 결과물을 확인할 수 있다.  
원하는 기능의 위젯과 레이아웃, 버튼 등의 도구를 통해 즉각적으로 이용 및 확인 가능하다.

이러한 직관적으로 결과물을 확인할 수 있다는 특성은 곧 Prototyping 과정에서,  
실제로 어플을 실행하여 확인하는 절차를 거치기 용이하다는 장점을 갖는다.

**4. 실제 프로젝트 진행 후**

- 실제 결과물

해당 항목에서는 프로젝트를 진행하면서 실제로 개발한 항목들과, 개발하지 못한 부분들,  
추가적으로 개발했으면 좋았을 것 같은 전체적인 내용을 포함한다.

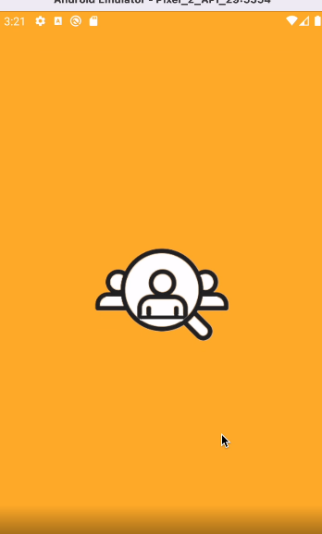
Android Studio 프레임워크를 완벽히 이해하기엔 턱없이 모자란 시간이었으나,  
프로젝트 진행 전 논의된 사항대로 외부 API들과 오픈소스들을 참고하여   
실제로 어플이라 부를 수 있는 정도의 결과물을 도출해냈다.

해당 어플은 사용자의 개인정보가 필요하기 때문에 회원가입을 통해 서비스를 이용할 수 있고,  
회원가입을 한 사용자들은 팀 구성, 팀 관리 등의 서비스를 이용할 수 있다.

또한 실제로 사용자들이 기입한 정보를 Firebase 내부의 데이터베이스에 저장하고,  
데이터베이스 내에서 UID를 통해 암호화된 식별 코드를 통해 특정 권한을 지닐 수 있다.

개발한 소프트웨어가 어플리케이션임을 감안하여 보고서는 최대한 실행화면에 초점을 맞췄으나,  
부차적인 설명이 필요하다 판단된 항목에 대해서는 Firebase의 화면과 실제 개발 코드의 일부도  
같이 첨부하였다.

1. 초기 화면



↑ 스플래시 이미지 ↑ 어플 실행 초기 화면

어플 실행 시 가장 먼저 볼 수 있는 화면으로,  
어플의 실행 성공을 알리는 스플래시 이미지가 2초간 나타난 후, 오른쪽 메인 화면을 볼 수 있다.  
해당 화면에서 사용자는 로그인과 회원가입, 두 가지 기능 중 하나를 선택하여 클릭할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명2-1.회원 가입 화면

텍스트이(가) 표시된 사진

자동 생성된 설명

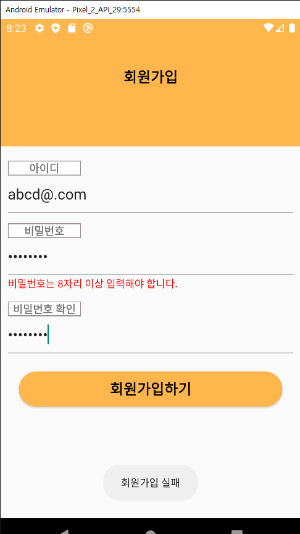
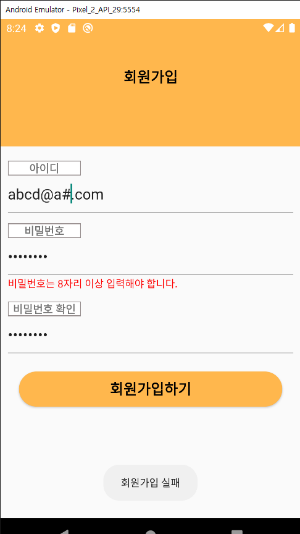
↑ 회원 가입 화면 ↑ 회원 가입 성공 시 화면

회원 가입 창에서는 사용자가 사용할 아이디를 이메일 형식으로,   
비밀번호를 8자 이상의 형식으로 기입하여 회원 가입을 진행할 수 있다.

이 때, 잘못된 비밀번호를 입력할 가능성이 있으니 한 번 더 자신의 비밀번호를 확인할 수 있는  
‘비밀번호 확인’ 칸을 통해 사용자가 사용하게 될 비밀번호를 재확인한다.

만약 ‘비밀번호’ 칸에 작성된 값과 ‘비밀번호 확인’ 칸에 작성된 값이 일치하지 않거나,  
비밀번호가 8자리 미만으로 구성되어 있다면 ‘회원가입 실패’ 라는 문구가 출력된다.

초기 목표는 다른 문자들까지 사용하여 사용하는 문자의 수를 늘리려 하였으나.  
시간 관계상 완성하지 못했다.



↑ 위와 같이, 비밀번호가 일치하지 않거나, 8자리 미만인 경우 ‘회원가입 실패’ 메시지 출력

회원가입이 완료되면, 사용자는 별도의 작업 없이도 바로 메인 페이지로 이동한다.

2-2. 회원 가입한 정보들과 Firebase 연동  
테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 회원가입에 성공하게 되면, 사용자의 이메일과 비밀번호는 Firebase 내부  
‘Authentication’ 영역에 저장된다. 이때 데이터베이스 상에서 사용자간 구별을 하기 위해,   
사용자 UID라는 일종의 암호 키를 생성하는데, 이후 해당 사용자가 게시판 글을 작성한 후   
추후 수정 및 삭제를 하는 경우 등 권한이 필요한 작업을 할 때, 사용자 UID를 통해 사용자를   
식별할 수 있게 한다.

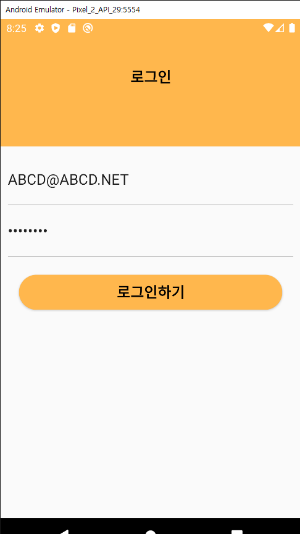
안드로이드 스튜디오와 Firebase를 연동하는 방법은, Android Studio 내부 Tools 바를 통해  
Firebase와 연동을 진행할 수 있었다.

이후 Firebase 관련 플러그인을 설치하고, google-services.json 파일이 app 폴더에 생성되었는지를 확인하는 간단한 절차만 거치면 된다.텍스트, 모니터, 스크린샷, 검은색이(가) 표시된 사진

자동 생성된 설명

위 코드와 같이, 메인 페이지로 이동 후, ‘뒤로 가기’ 버튼을 클릭하면 화면이 종료된다.

1. 로그인 화면



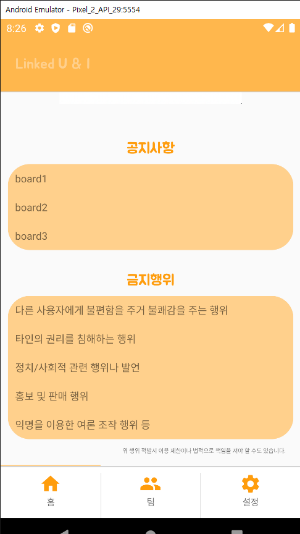
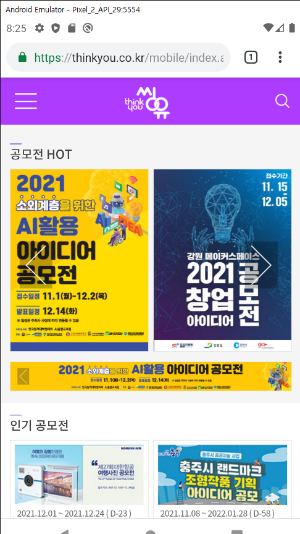
↑ 로그인 화면 ↑ 로그인 실패 화면 ↑ 로그인 성공 시

이미 회원 가입을 한 회원의 경우, 생성한 아이디를 통해 로그인을 진행할 수 있다.  
로그인을 진행할 때, 해당 회원의 아이디 및 비밀번호를 데이터베이스에 저장되어 있는  
UID와 비교한다. 비밀번호가 틀리거나, 가입한 이메일이 존재하지 않는다면 에러 메시지가  
나타난다.

로그인 실패 시, 아이디가 존재하지 않거나, 비밀번호가 옳지 않다는 두 가지 경우의 수가 있다.  
하지만 해당 프로젝트에선 자세하게 ‘아이디가 존재하지 않습니다’와 ‘비밀번호가 옳지 않습니다’  
같은 자세한 오류 메시지를 출력하는 것이 아닌,  
단순히 ‘로그인 실패’ 메시지만을 출력하게 제작하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

4. 메인 페이지

↑ 메인 페이지 ↑ 하이퍼링크 1 ↑ 하이퍼링크 2 ↑ 개발자 게시판

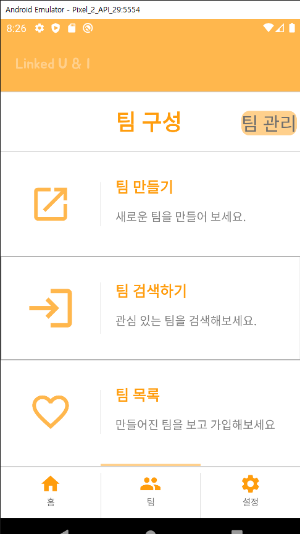
메인 페이지에서는 공모전 사이트에 대한 정보들을 배너 형식으로 제공한다.  
배너에 있는 공모전 사진을 클릭하면 해당 사이트로 넘어갈 수 있는 하이퍼링크를 설정했다.

프로젝트 진행 중 머신러닝을 이용하여 사용자들이 자주 검색하는 키워드들을 종합,   
사용자들에게 ‘인기 있는 키워드’의 형식으로 제공하고자 하였고,

또한 해당 공모전들을 단순히 나열하는 것이 아닌, 사용자의 데이터를 학습시켜  
사용자 개개인에 맞는 공모전과 취업 관련 정보를 웹데이터 크롤링을 통하여 나타내고 싶었으나,  
이 두 가지 기능은 시간과 기술적 한계로 구현하지 못했다.

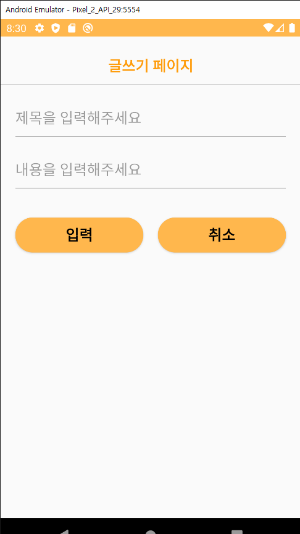
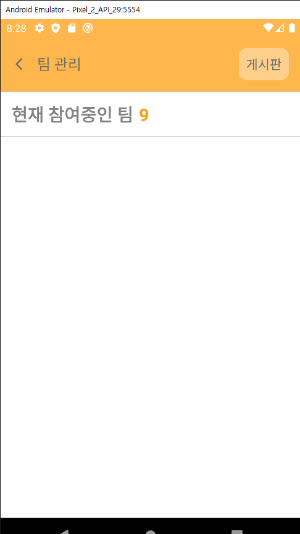
또한 개발자 게시판에는, 개발자가 사용자들에게 제시하는 ‘공지사항’ 과,   
사용자들로 하여금 어플을 사용할 때 지켜야 할 규범을 ‘금지행위’ 카테고리를 통해 제시한다.

또한 메인 화면의 하단 위젯들을 통해 팀 구성 화면과, 설정 화면으로 이동할 수 있게 하였다.



↑ 팀 구성 화면 ↑ 설정 화면

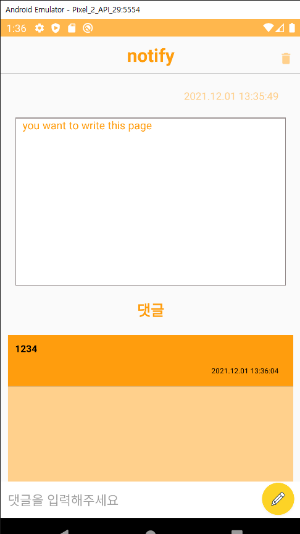
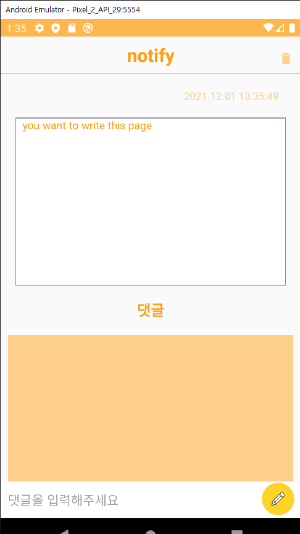
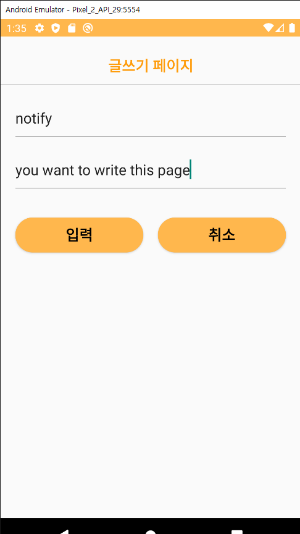
5-1. 게시판



↑ 게시판 항목 진입 ↑ 팀 관리 화면의 우측 상단의 버튼 ↑ 제목과 내용 입력

기본적으로 게시판은 팀 관리 화면에서 사용할 수 있는 기능이다.  
이미 구성된 팀에 가입되어 있는 경우, 팀 관리 화면의 우측 상단 ‘게시판’ 버튼을 클릭하여  
‘글쓰기 페이지’로 이동이 가능하다. ‘글쓰기 페이지’ 에서는 제목과 내용을 작성하여 게시할 수   
있고, 게시판에 작성된 내용은 전부 Firebase 내부의 데이터 베이스에 저장이 된다.  
게시글 밑에 작성할 수 있는 댓글들의 경우도 전부 데이터 베이스에 저장이 되며,  
게시글이나 댓글의 삭제는 사용자의 UID값과 데이터베이스 내부의 UID 값을 비교하여  
삭제 버튼을 클릭한 사용자의 UID와 게시글 및 댓글을 작성한 사용자의 UID가 일치하는  
경우에만 삭제가 가능하도록 구현하였다.

또한 댓글을 삭제하는 경우, 사용자에게 정상적으로 삭제가 되었다는 메시지를 보여주는  
Dialog 창까지 구현하였다.

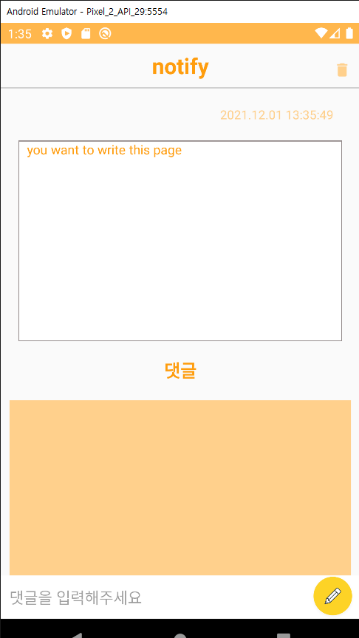


↑ 게시글 작성 예시 ↑ 게시글이 작성된 모습 ↑ 댓글이 작성된 모습

5-2. 게시판과 Firebase 연동

텍스트이(가) 표시된 사진

자동 생성된 설명게시판에 사용자가 글을 작성하면, 글의 제목과 내용, 작성자의 UID 코드들은 자동으로  
Firebase 내부에 저장된다. 이 저장된 데이터를 이용하여   
추후 사용자가 게시글을 삭제하려고 할 경우, 작성자의 UID와 삭제를 시도하는 UID를 비교하여  
일치하는 경우에만 글 삭제 권한을 부여한다.



↑ 위와 같이 게시글이 작성된 경우, Firebase 내부 Realtime Database에 게시글과 관련된 데이터가 저장된다.

텍스트이(가) 표시된 사진

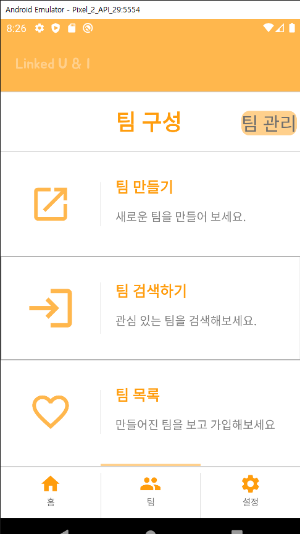
자동 생성된 설명

↑ 사용자가 삭제 버튼을 클릭했을 때, 삭제 관련 Dialog창을 출력하는 알고리즘 코드

또한 게시글이나 댓글이 삭제되었을 때, 사용자에게 정상적으로 삭제되었음을 알리는  
Dialog 창까지 구현하여, 사용자가 게시글을 관리하는 데 좀 더 직관적인 이해를 돕고자 하였다.

6. 팀 구성 화면

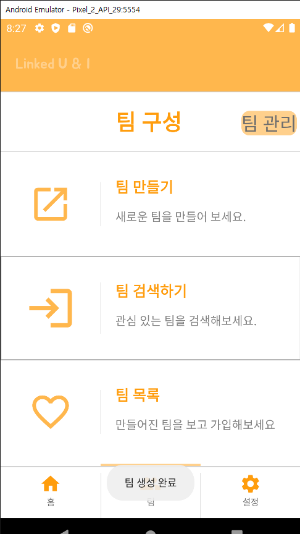
텍스트이(가) 표시된 사진

자동 생성된 설명

↑ 팀 구성 화면 ↑ 팀 찾기 화면 ↑ 팀 만들기 화면

팀 구성 기능에서 팀을 만들 사용자는 팀 만들기 기능을 통해,   
팀에 참여할 사용자는 팀 찾기 기능을 통해 현재 생성되어 있는 팀들의 목록을 조회하고,  
가입하고자 하는 팀에 대한 가입 여부를 확인할 수 있게 하였다.  
목록에 있는 팀들 중 하나를 선택하면 해당 팀의 세션으로 페이지가 이동하여, 팀원 간   
협업을 가능하게 한다.

팀 구성 기능은 크게 팀 만들기, 팀 찾기, 팀 목록 세 가지의 기능을 구현하였다.

1. 팀 만들기

↑ 팀 만들기 화면 ↑ 팀 생성 완료 확인문구

팀 만들기 기능에는 팀 이름, 활동 주제, 팀장 이름, 팀의 생성일을 사용자로부터  
입력 받는다. 사용자가 ‘생성하기’ 버튼을 누르면 Firebase에 입력한 정보를 전달하여  
팀이 생성되며, 팀 찾기 기능에서 다른 사용자가 해당 팀을 검색할 수 있다.   
또한 팀을 생성한 사용자로 하여금 팀 생성이 완료되었다는 문구를 표시하여 보여준다.

1. 팀 찾기 및 팀 목록 화면

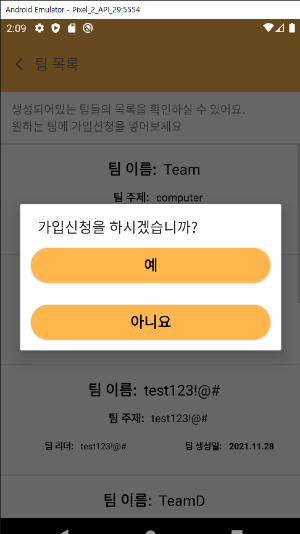
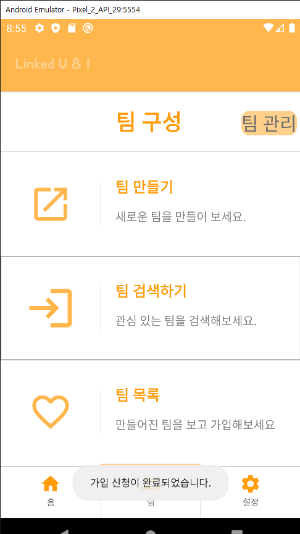
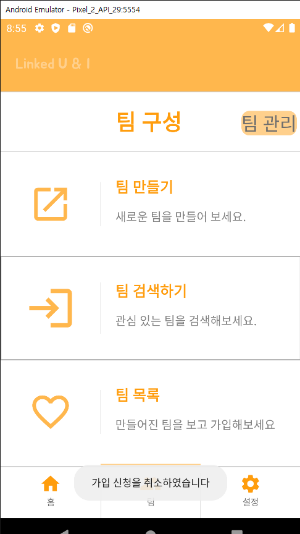
텍스트이(가) 표시된 사진

자동 생성된 설명

↑ 팀 검색 화면 ↑ 팀 목록 화면

팀 찾기 기능에는 대응하는 검색어를 사용자로부터 입력을 받아,   
검색어와 매칭되는 팀 목록을 화면에 보여준다. 목록 중 가입하고자 하는 팀을 사용자가   
선택하면, 해당 팀의 팀장에게 가입신청을 요청한다.   
이때 팀장은 가입 승인 여부를 결정하며, 승인 시 팀 가입이 완료되는 절차를 구상하였으나,  
가입 요청을 처리하는 부분을 구현하는 데 있어 시간상, 기술상의 문제로 구현하지 못하였다.

초기 구상에서는 가입 신청 및 승인 여부를 사용자가 직접 확인할 수 있도록 구상했으나,  
Firebase와의 연동에서 어려움이 있어, 이미 가입이 승인되었다는 가정 하에 가입이 완료된  
팀들의 목록을 보여주는 기능으로 구현하였다.



↑ 가입 신청하기 ↑ 가입 신청완료 ↑ 가입 취소하기

가입 신청을 한 사용자의 경우, 가입 신청 및 취소에 대한 기능은 구현해내는데 성공했다.

- 테스트 결과

해당 보고서 3, 프로젝트 진행 전 논의사항 중 ‘테스팅 방안에 대한 논의’에서 언급한 두 항목

1. Completeness: 소비자가 필요로 하는 모든 기능이 포함되었는가?
2. Verifiability: 요구사항들이 검증될 수 있는가?

두 가지 항목에 초점을 맞추어, Prototyping을 통해 검증을 실시하였다.

1. Completeness테이블이(가) 표시된 사진

자동 생성된 설명  
해당 항목은 개발 전 작성한 ‘요구사항 명세서’를 개발 중 실시간으로 수정하면서 검증하였다.

구현을 성공한 부분에 대해서는 초록색 영역과 기호 표시(O)를 통해 완료되었음을,  
개발이 되었으나 미심쩍은 부분이 있거나, 큰 틀은 잡혔으나 세부적인 항목에서 모자람이 있는   
경우 주황색 영역과 기호 표시(△)를 통해 표시하였고,

시간적, 기술적 제약으로 구현하지 못한 부분은 별도의 색깔 영역 없이 기호 표시(X)로 구분했다.

위 항목을 기준으로 체크리스트를 작성해 테스트를 하였다.



- Verifiability  
해당 항목은 사용자의 관점에서 가능한 많은 시도를 통해 어플 내부의 세부적인 오류 사항과  
보완해야 할 사항들을 문서화하였다.

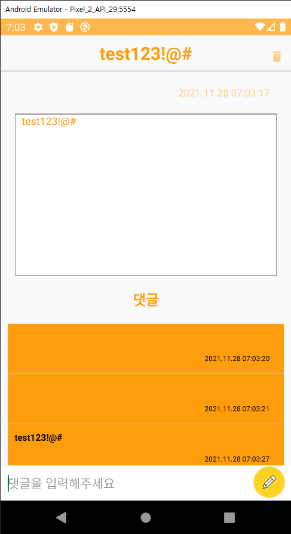
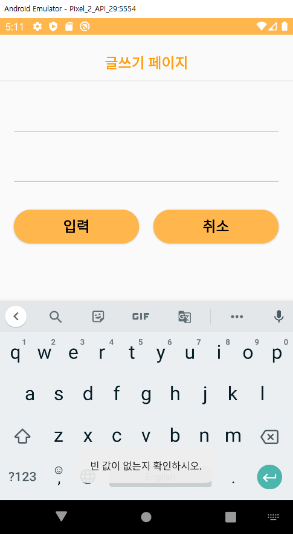
검출된 오류들 및 보완해야 할 사항

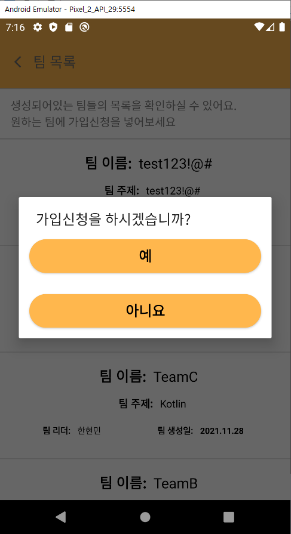
1. 로그인 시, 아이디의 대,소문자 구별 없이 내용이 일치하면 로그인이 가능함.  
ex) 비밀번호가 동일하다는 가정 하에,  
[afd@dankook.ac.kr](mailto:afd@dankook.ac.kr)의 아이디와 [Afd@dankook.ac.kr](mailto:Afd@dankook.ac.kr) 두 가지 모두 로그인이 가능

2. 회원가입 시 아이디에 특수문자 입력가능함

3. ‘팀 목록’ 중 자신이 생성한 팀에 자신이 직접 가입 신청을 보낼 수 있음.

4. ‘팀 가입’ 신청 시 ‘예’ 또는 ‘아니오’를 선택하는 화면에서 ‘아니오’를 선택하면   
팀 목록이 유지되지 않고 메인 페이지로 이동함.

텍스트이(가) 표시된 사진

자동 생성된 설명5번 항목 6번 항목 7번 항목

5. 오른쪽 화면이 뜨자마자 마우스를 ‘예’ 또는 ‘아니오’ 위에서 스크롤을 위 혹은 아래로 돌리면, 간헐적으로 임의의 버튼이 클릭이 되는 현상이 발생함.

→ Android Studio 내부의 Emulator로 실행했을 때 발생한 오류로,  
모바일 디바이스로 실행했을 땐 오류가 발생하지 않음.

6. 신규 작성시 빈칸이나 띄어쓰기만으로는 작성이 안되지만, 띄어쓰기 후에 작성된 내용은  
띄어쓰기가 생략되지 않고 그대로 입력됨

→ ‘6번 항목’ 오른쪽 예시와 같이, ‘ 1’ 과 같은 형식의 작성이 가능함.

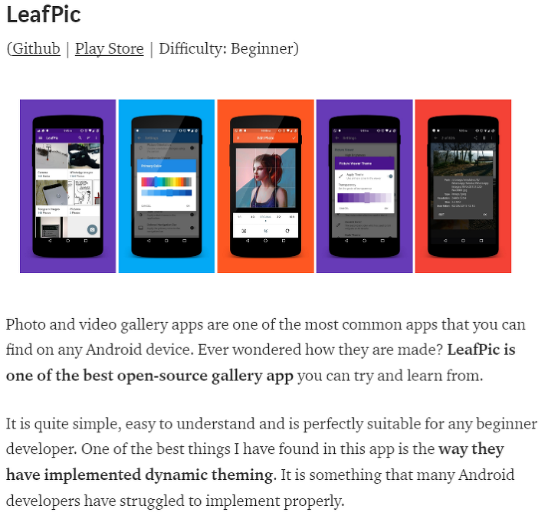
7. ‘팀 관리’ 항목 중 게시판 댓글에, 띄어쓰기와 같은 공백 문자만으로 댓글 작성이 가능함.

8. 어플 내에서 ‘뒤로 가기’ 버튼을 누르면 상위 화면으로 넘어가는 다른 어플들과 달리,  
실행 바로 직전의 화면으로 넘어가는 불편함이 있음.

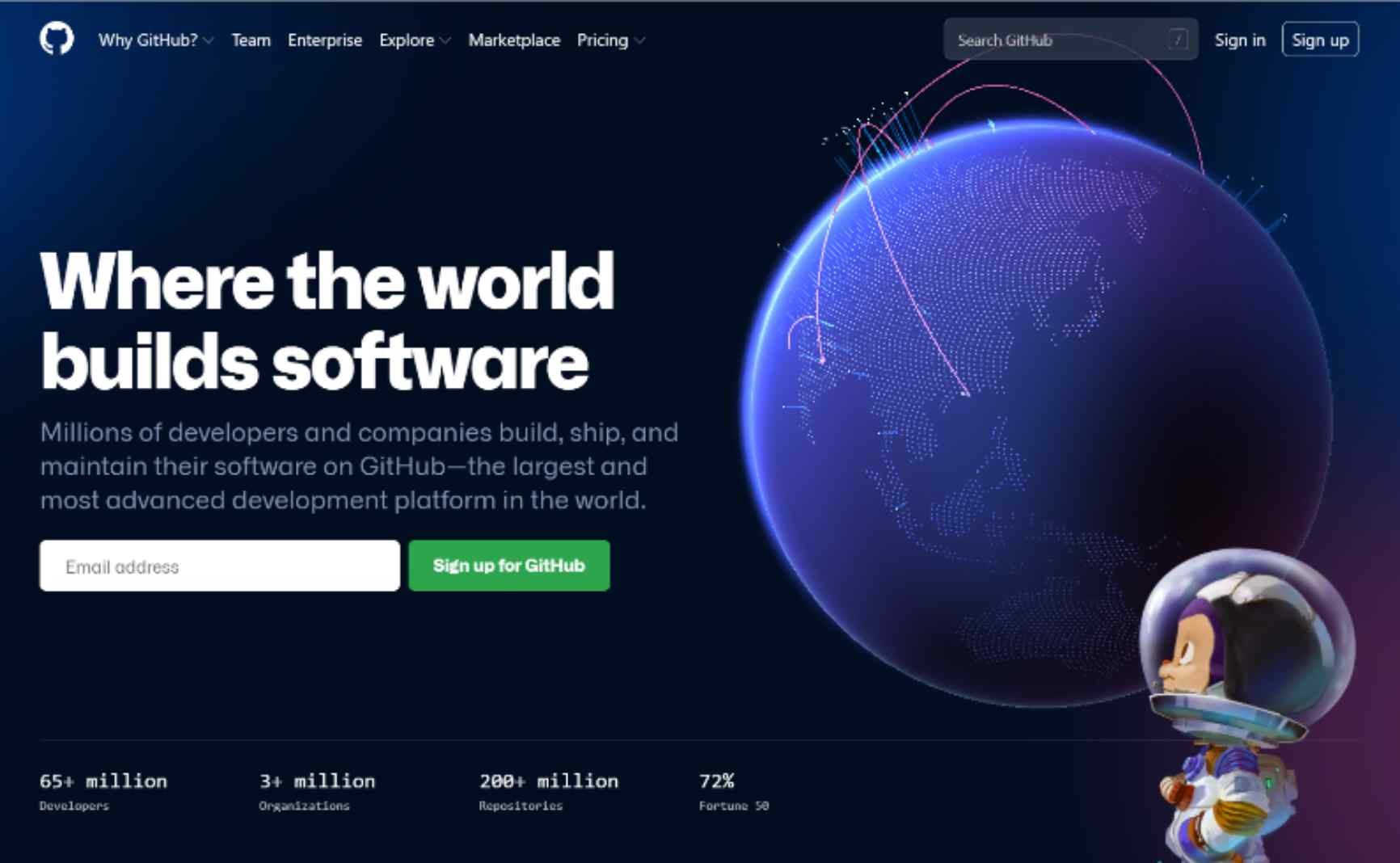
테스트 결론

요구명세서 내의 기획 대비 구현된 비율은 65%로 미구현된 내용의 테스트는 제외했다. 구현된 54개의 기능중 7개의 기능에서 오류가 발견되었다. 7개의 오류들은 조건을 추가하면 수정할 수 있는 것들이다. 구현이 완료 되었다는 가정하에 테스트를 한다면 사용자의 각종 정보와 기능들이 충돌하지는 않는지, 팀 가입 과정에서 소통이 매끄럽게 진행 될 수 있는 구조인지, 텍스트 입력 부분에서 각각의 기능들이 빈칸이나 띄어쓰기, 특수문자 같은 특수한 경우에도 실행이 되는지, 사용자 입장에서 편의성을 더해야 하는 부분이 있는지 등의 테스트를 할 필요가 있다고 생각된다.

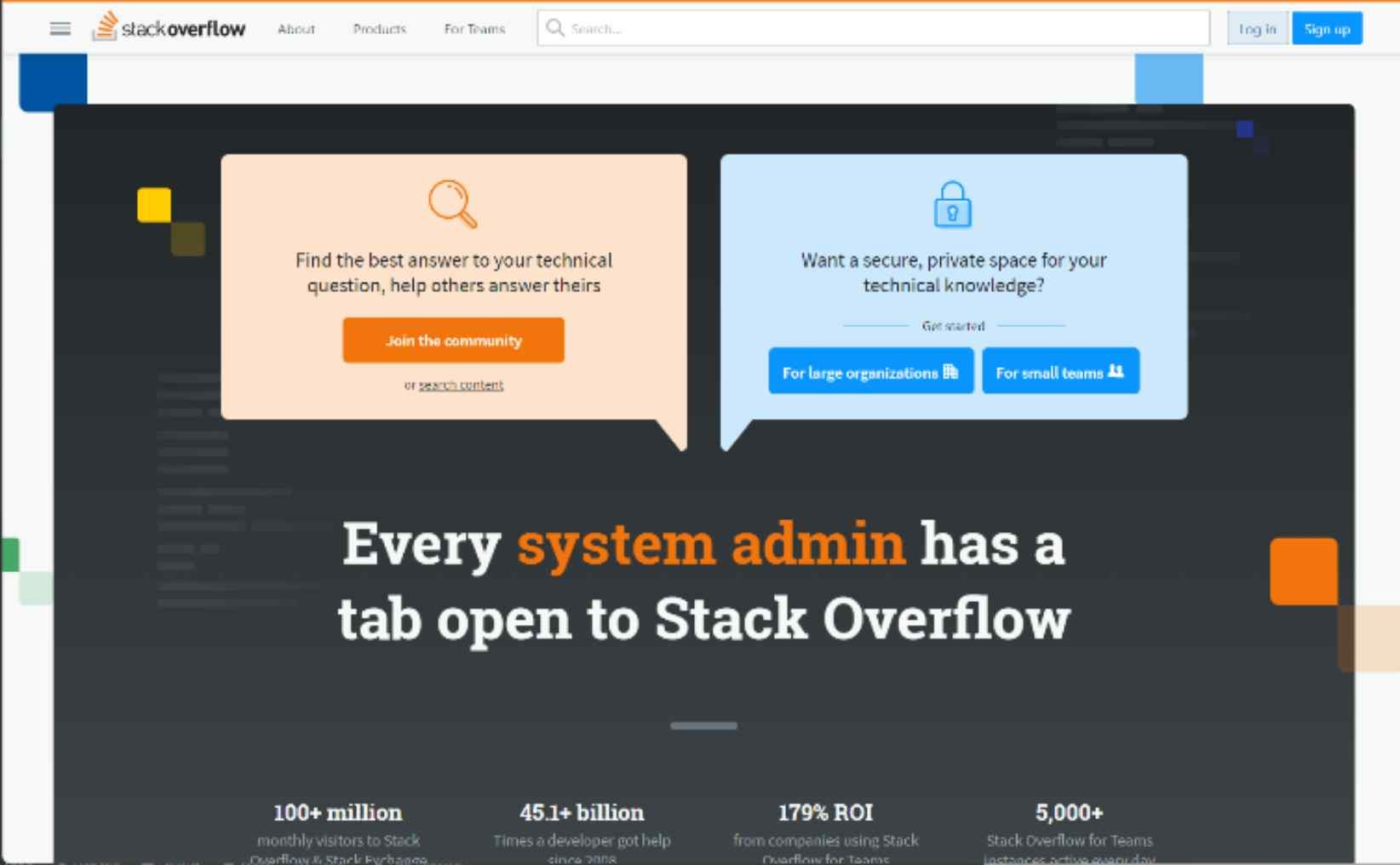
- 참고 자료  
짧은 기간동안 이론적 배경을 바탕으로 실제 프로젝트를 진행하면서,  
우리 팀은 ‘초보 개발자들’로서 오픈소스를 사용하는 게 필수불가결했다.  
Kotlin 언어의 경우 2011년에 출시된 언어로 10년이 넘는 긴 시간동안  
수많은 개발자들이 개발한 어플의 데모버전들이 많아 UI 디자인 및  
실제 소프트웨어 개발에도 상당수 도움이 많이 되었다.

1. LeafPic  
  
안드로이드 어플을 개발할 때 참고할 만한 다른 어플들을 모아 놓은 사이트로,  
개발을 공부하는데 있어 이론과 실기를 공부하는 것보다 이미 완성된 프로젝트의  
소스를 보는 것이 더 도움이 될 때가 있다는 조언을 바탕으로 참고하였다.

2. Github

   
개발자들이 자신이 개발한 소프트웨어들을 업로드하여 저장하는 일종의 저장소로,  
다른 사용자들이 유용하게 사용할 수 있도록 공유해 놓은 양질의 오픈소스들과 API를  
참조할 수 있었다.

해당 프로젝트를 진행하면서 팀원들 간의 코드 공유와 피드백에도 유용하게 사용하였다.

3. Stack Overflow  


개발자들 간 질의응답을 할 수 있는 최대 규모의 개발자 커뮤니티로,  
개발을 하는데 있어 맞닥뜨린 어려움들과 오류의 원인들에 대한 해결책을 얻을 때  
유용하게 사용하였다.

이 외에도 Kotlin 언어 학습을 위해 인터넷 강의 사이트에서 유료 결제를 하고,  
다른 여러 사이트와 블로그, 서적들도 참고하였으나, 실제 프로젝트를 진행하면서 느낀 점은  
‘프로그래밍 언어에 대한 이해도는 프로젝트를 진행하면 자연스레 높아진다’는 점이었다.

Kotlin 언어에 대한 이해가 조금 부족한 상황에서도 실제 동작되는 결과물을 제작해 내는  
프로젝트를 진행하면서 가장 유용하게 사용했던 참고 자료들은 상술한 세 인터넷 사이트이다.

- 프로젝트를 마치며  
3개월이라는 길다면 길고, 짧다면 짧은 시간 속에서 다른 전공 수업들과 각자가 준비하는 분야에 대한 공부를 하면서도 이 프로젝트에서 각자 맡은 바에 최선을 다해 하나의 멋진 결과물을   
만들어 냈다는 것에 그동안 느꼈던 피로와 고단함이 씻겨 내려가는 듯한 느낌이 들었다.

처음으로 팀 규모의 프로젝트를 진행하며 애로사항도, 시행착오도 많았지만  
프로젝트에 필요한 이론적 배경들을 단순히 이론으로만 익히는 것이 아닌,  
실제로 프로젝트를 진행하며 체계화하고 구체화한 이론들이 개발에 있어 큰 도움이 되고,  
이론을 공부하여 적용한 시간이 결코 헛되지 않았음을 몸소 체험할 수 있었다.

아직 소프트웨어 공학에 있어 요구되는 이론적 배경을 전부 익혔다고는 말할 수 없겠지만,  
일부의 이론이라도 실제로 적용하는 연습을 한 학기동한 진행한 값진 경험은

추후에 우리가 각자의 전공 필드에 나갔을 때 큰 도움이 될 것이라 자신할 수 있다.