

Life

Emotion

Move
2024

Exercise

당신을 위한 Trend
Move가 제공하는 Relationship

Team Movement

010 . 4222 . 1358

fletcher_seth2335@naver.com

What is Move ?



인공지능 모델과 OpenAPI를 활용한
생활체육 통합 소셜미디어

Index.

1

Introduction

소개

2

Architecture & Tech & Tools

설계구조 & 기술 & 도구

3

Planning

기획

4

Development

개발

5

Test

테스트

6

Result

결과물

7

Issues

이슈

1. Introduction

(1) BM Trend

(2) Project Intention

(3) Market Analysis

(4) Distinction

1. Introduction

(1) BM Trend

트렌드 분석

3C 기반 BM 변화 및 확장

ex) 무신사, 당근마켓 등

재미/정보/취향 공유
(Community)



관련 콘텐츠 생성
(Contents)



전자상거래플랫폼
(Commerce) 확장

디지털 공간을 통한 수익창출 기회&니즈 상승

ex) 굿닥칼리지, 제페토 등

유튜브, SNS 등의
유명 크리에이터&인플루언서
인기 = 자산



적극적인 서비스 이용자에게
수익창출 기회 및
금전보상 제공



충성도 높은 이용자 유지, 확보

O2O 서비스로서의 커뮤니티 플랫폼 경쟁 확대

ex) 카카오, 네이버,
트레바리, 문토 등

1) 네이버 :

'오픈톡 커뮤니티 서비스'

2) 네이버 카페 + 메타버스
= 신규 비즈니스 플랫폼 전환

3) 카카오 : 오픈채팅 →

관심사 기반의 비지 커뮤니티
'오픈링크'로 확장

실제 사례

개발자 커뮤니티 '커리어리'



콘텐츠 플랫폼 '퍼블리'

1) '커리어리'에서 활동하는
직장인을 저자로 섭외하여
퍼블리에 글 발행 시
금전적 수익창출 기회 제공

2) '커리어리' 내

사이드 프로젝트 코너 통해
팀원 구하는 소통 창구 마련

1. Introduction

(2) Project Intention

기획 의도

(1) 문제 제기 : 생활체육을 위한 전용 소셜 미디어 서비스의 부재 & 기성 SNS의 한 카테고리에 불과

(2) 니즈 분석 : 생활 체육인들의 자랑과 소통을 위한 소셜미디어 & AI 기술로 편리성 증가

① AI 기능 도입 편리성 증대

몸자랑 사진 업로드시 얼굴 자동마스킹,
챗봇에게 식단, 운동 추천 질답
관심 있는 운동 분야의 피드 맞춤 추천

② 희귀한 분야 포함 폭 넓은 커버리지

축구, 헬스 등 메이저한 분야부터
격투, 수상 스포츠까지

③ 생활 체육 관련 통합 소셜미디어

생활 체육인들의 커뮤니케이션 공간



Main Identity ⇒ Platform / Integrity

1. Introduction

(3) Market Analysis

서비스 시장 분석



1. 국내 스포츠산업 시장 규모

약 78조원(22년 기준)
변동추이 : 지속적 우상향



2. 세대별 SNS 이용률 추이

우상향 그래프를 그리며 지속적으로 상승

스포츠 산업
& SNS 이용률
지속적 증가·성장



①
소셜미디어의
소비 잠재력 ↑

②
O2O 서비스로의
유연한
BM 확장 가능성

자료 출처

국내 스포츠산업 시장규모 <https://www.junggi.co.kr/article/articleView.html?no=32065>

세대별 SNS 이용률 추이 <https://www.yna.co.kr/view/GYH20220621000200044>

1. Introduction

(4) Distinction



AWS Rekognition으로 얼굴 인식 후 얼굴만 가리기

오늘의 운동 완료 후
멋진 내 몸을 자랑하고 싶은데
몸만 나오도록 사진 찍기 번거롭고,
얼굴이 나오면 내가 직접
편집해야 하는게 너무 귀찮다!



「얼굴 가리기」 선택시
자동으로 마스킹 처리!



ChatGPT OpenAPI를 이용해서 운동 꿀팁 얻기

운동은 해야되겠는데,
PT는 너무 비싸고
가볍게 시작하고 싶은데
운동도 식단도 모른다!



운린이 헬퍼 챗봇과 대화하며
즉시 간단한 궁금증 해결!



LightFM 추천 모델이 내 관심 운동분야 피드만 추천

나는 축구에만 또는 웨이트에만
관심이 있어서
다른 운동 말고
해당 분야 피드만 보고싶다!



인공지능 추천 모델이
내게 맞는 콘텐츠를 추천!

방향성 및 차별점

2. Architecture & Tech & Tools

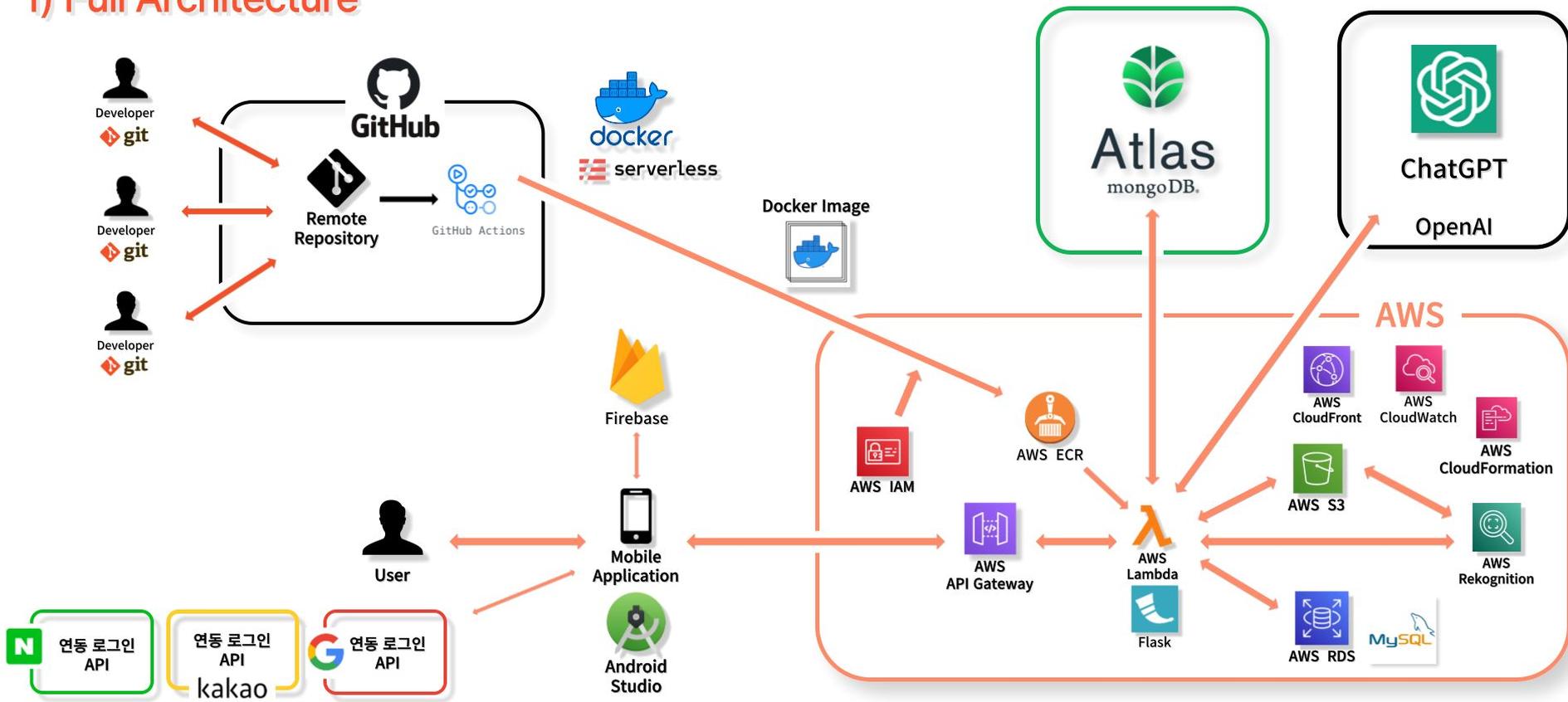
(1) Full Architecture

(2) Front & Back-Office Architecture

(3) Used Tech & Tools

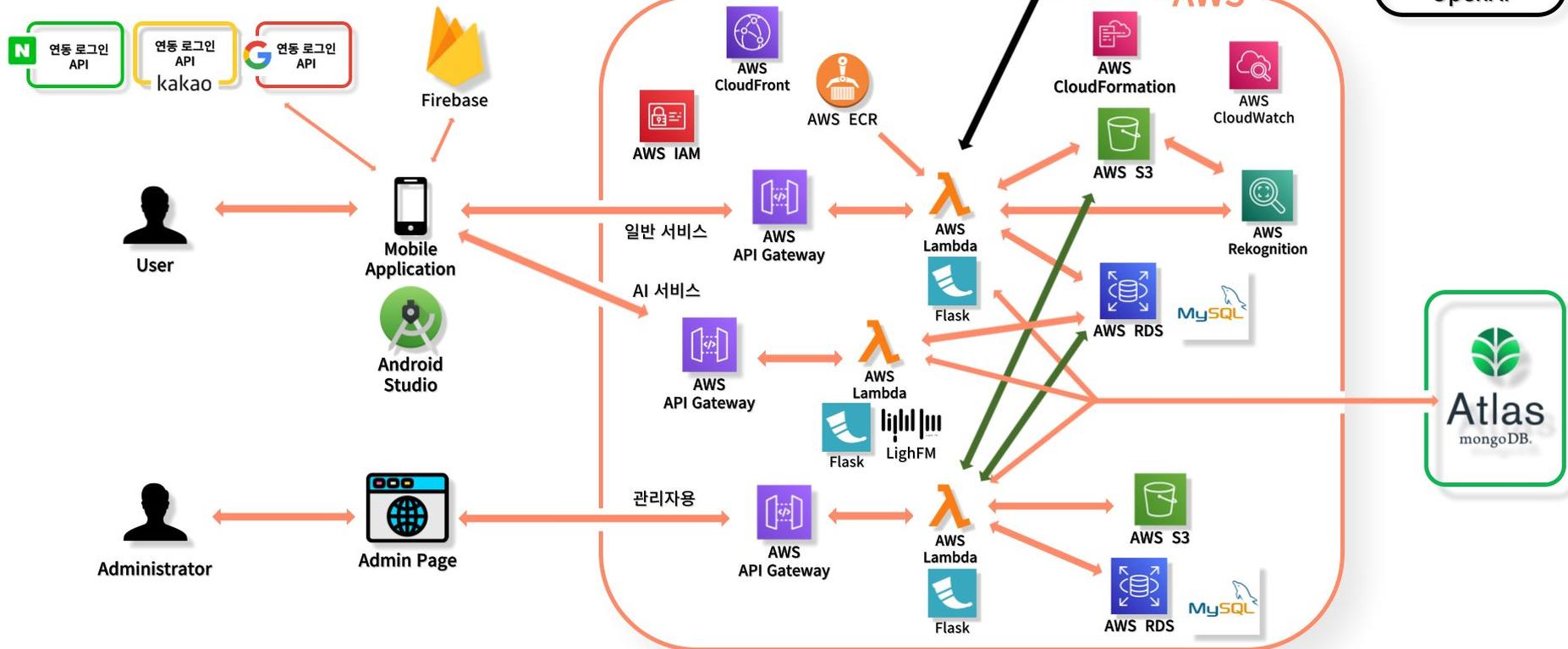
2. Architecture & Tech & Tools

1) Full Architecture



2. Architecture & Tech & Tools

2) Front & Back-Office Architecture



2. Architecture & Tech & Tools

3) Used Tech & Tools

Back-End(1)



Python



Flask

1. LightFM, Scikit-Learn 라이브러리 등 **AI 모델** 이용을 위해 Python 언어 선택
2. Flask 프레임워크의 **flask_restful** 라이브러리를 이용하여 **Restful API 서버** 개발
3. **flask-jwt-extended** 라이브러리 이용하여 **JWT 인증토큰** 사용해서 유저 인증 기능 구현
4. **passlib** 라이브러리의 **pbkdf2_sha256.hash()** 함수와 **salt** 값을 사용해서, 보안을 위해 단방향 해싱 함수 구현하여
패스워드 처리



AWS RDS



1. AWS RDS 인스턴스 생성하고 엔드포인트, 마스터 계정-암호 정보를 MySQL Workbench에 Connection 설정하여
클라우드 환경에서 데이터베이스 접근
2. **mysql-connector-python** 라이브러리 이용하여 Flask 서버와 데이터베이스 리소스 연결
3. **index 처리**를 통해 데이터베이스에서 특정 데이터 조회시 검색 연산 효율 증대
4. **foreign key 설정**을 통해 데이터의 중복을 막고 리소스 누수 방지
5. 파일 업로드시 S3에 저장 후 해당 **파일의 URL 정보**는 RDBMS에 저장

2. Architecture & Tech & Tools

3) Used Tech & Tools

Back-End(2)



AWS S3

1. **boto3 라이브러리**를 이용하여 IAM 사용자의 ACCESS KEY와 SECRET KEY 정보로 **Client 객체** 생성 후 Request의 이미지, 비디오 파일 고유하게 Rename 후 S3에 저장



Atlas
mongoDB.

1. **NoSQL** 중 Document 형식인 **MongoDB** 이용하여 게시물 조회기록, 검색 기록 등 불특정한 대량의 데이터 저장 및 조회 처리
2. **index 설정**을 통해 UQ 지정 및 검색 효율 증대
3. **동적 스키마**의 성격을 이용하여 유저의 활동 기록을 유연하게 저장, 조회 및 관리
4. 수집한 활동기록은 Back-Office에서 통계치로 가공

2. Architecture & Tech & Tools

3) Used Tech & Tools

Front-End



AndroidStudio



Firebase

1. **Yoyo** 라이브러리로 Video Pause/Play 아이콘에 애니메이션 효과 부여
2. **CircleImageview** 라이브러리를 사용하여 프로필 뷰 개발시 효율적으로 작업
3. **Glide** 라이브러리를 이용하여 이미지 로딩, 변환, 캐싱, 네트워크 에러 처리 등의 작업에서 효율성 증대
4. **exoPlayer** 라이브러리를 적용해서 HTTP 및 Cache 데이터 소스를 이용하여 **Video Streaming** 구현 및 Controller의 스타일을 **Customizing**
5. **Retrofit** 라이브러리를 사용하여 비동기식으로 API 서버와 네트워크 통신하여 데이터 **Rendering** 작업
6. Firebase의 **RealTime Database** 이용해서 실시간 데이터 동기화를 통해 서비스 내 **Realtime Chat** 구현
7. **Firebase Cloud Messaging** 이용해서 안드로이드의 **BroadCast Reciever**로 백그라운드 서비스에서 **Notification** 수신 처리

2. Architecture & Tech & Tools

3) Used Tech & Tools

DevOps(1)

CI/CD 파이프라인 구축



AWS IAM



AWS ECR

1. 작업시 Local에서 **Git** 소프트웨어를 이용하여 **소스코드 버전관리**
2. 각자의 작업물을 **GitHub Remote Repository**에 **Commit&Push**하며 **클라우드 환경**에서 **CI** 구축하여 협업
 - Main 브랜치 외 각자의 **Branch** 형성하고 소스코드를 Pull하여 개발 및 테스트 후 **Pull Request** 생성
 - Pull Request 생성 후 팀 코드 리뷰&검토 후 Main 브랜치에 **Merge**
3. **GitHub Actions** 이용하여, Main 브랜치에 Push 이벤트 발생 시 수행할 Action에 해당하는 **Workflow**를 .yaml 파일로 작성하고 지정하여 **자동 배포** 작업하여 **CD** 구축
4. 서버 배포시 개발 환경과 서버 환경 사이 효율적인 동기화 및 리소스 사용률 증대 위해 **Docker** 이용
5. GitHub Actions가 Workflow 수행시, **자동 배포**를 위해 **Serverless 프레임워크**를 이용
 - 1)먼저 해당 소스코드를 **컨테이너화** 하여 **Docker Image build**
 - 2)**AWS IAM 사용자 Credentials** 정보로 **AWS ECR** 서비스에 접근하여 해당 **Docker Image**를 **Push**하여 저장
 - 3)push 된 Docker Image 삽입하여 **Lambda Function** 생성

2. Architecture & Tech & Tools

3) Used Tech & Tools

DevOps(2)



AWS CloudFront



AWS CloudWatch



AWS CloudFormation



AWS IAM

6. Serverless 프레임워크를 이용하여 deploy할 때, build 된 Docker Image를 **AWS CloudFront** 이용하여 ECR에 효율적으로 Push

7. Serverless 프레임워크 이용하여 deploy 할 때, ECR에 Push 된 Docker Image를 삽입하여 AWS Lambda Function를 생성할 수 있도록 **AWS IAM 사용자 CloudFormation FullAccess 권한 설정** 및 **AWS CloudFormation 템플릿** 이용하여 **Stack** 작업하도록 Infra를 **프로비저닝**

AI & Open API



연동 로그인 API



연동 로그인 API

kakao



연동 로그인 API



ChatGPT



AWS IAM



AWS Rekognition

Front-End에서 **연동 로그인 Open API** 호출 및 네트워킹하여 유저 인증 과정 간결화해서 쾌적한 사용경험 제공

Open AI 社の **GPT Open API** 이용하여 **챗봇** 기능 구현

AWS IAM 사용자 권한 설정으로 Rekognition **Full Access 설정** 후 **boto3** 라이브러리로 Client 객체 생성 후 AWS Rekognition의 **detect_faces()** 함수 사용하여 사용자의 업로드 사진 속 얼굴 식별

2. Architecture & Tech & Tools

3) Used Tech & Tools

Tools(1)



Anaconda



Figma



Slack



GitHub Desktop



Visual Studio Code



ERD Cloud



Postman



Google Colab



Android Studio



MySQL Workbench

1. Slack 이용하여 팀 프로젝트 생성하고 팀원 초대 및 채널 분리하여 협업 환경 구축

-서드파티 앱 연결(GitHub, Figma, GoogleDrive, ..)하여 이벤트 발생시 해당 채널에 알림 메시지 발생

-일정관리 채널에 주요 일정을 공유하여 프로젝트 기한 고지&준수 등 관리

-채널에 화면기획서, 테이블명세서, API명세서 등 문서 링크 공유하여 작업중 언젠가 재확인 가능하도록 관리

-개발 중 이슈 발생시 이슈관리 채널에 공유하여 개발상황 파악 및 담당자 배정하여 디버깅

2. 아이템 선정 후 Figma 이용하여 화면기획서 작성 및 협업 → prototyping하여 Wireframe 설계 및 링크공유

3. GitHub Remote Repository 생성 후 GitHub Desktop으로 Clone하여 Local Repository 연동

→ Repository : (Android용), (Front-Office용), (Back-Office용), (AI Server용)으로 구분

4. Anaconda Prompt 이용해서 가상환경 생성하여 개발환경 구축

python==3.12, flask, flask_restful, scikit_learn, lightfm, pymongo==4.8,

flask_jwt_extended, pycoph-binary, mysql_connect_python, boto3 등

필요 언어 및 라이브러리 설치

2. Architecture & Tech & Tools

3) Used Tech & Tools

Tools(2)



Anaconda



Figma



Slack



GitHub
Desktop



Visual Studio
Code



ERD Cloud



Postman



Google Colab



Android
Studio



MySQL
Workbench

5. 화면기획서 바탕으로 DB 설계 후 ERD Cloud 이용하여 테이블 명세서 작성 및 공유

6. MySQL Workbench 이용하여 테이블 명세서 바탕으로 데이터베이스 스키마 구성

7. 화면기획서, 테이블 명세서 바탕으로 API 서버 개발

→ 각 개별 API 개발시마다 Postman으로 Request 발생시켜 Local 환경에서 Test

로컬 테스트에서 이상 없으면 서버 배포 후 End-point로 Request 발생시켜 Server 환경에서 Test

→ API 개발 후 Postman의 API Document 기능 이용하여 API 명세서 작성

8. AndroidStudio 이용하여 Java 언어로 Front-End 파트 개발

→ Minimum SDK Version : 23("Marshmallow", Android 6.0) : Coverage 98.8%

→ 안드로이드 환경에 최적화된 네이티브 앱 서비스 개발을 위해 AndroidStudio 이용

2. Architecture & Tech & Tools

3) Used Tech & Tools

Tools(3)



Anaconda



Figma



Slack



GitHub
Desktop



Visual Studio
Code



ERD Cloud



Postman



Google Colab



Android
Studio



MySQL
Workbench

9. Back-End 파트 중 API 서버 개발시 IDE 환경의 **VSC** 이용하여 API 개발 중, Postman으로 먼저

Local 환경에서 API 호출하여 단위 테스트 진행하며 터미널 로그 참고하여 **Debugging** 및 **소스코드 개발**

10. **Google Colab** 사용하여 **Scikit-Learn, LightFM** 라이브러리 이용해 협업 필터링 기반 **추천 AI 모델 개발**

3. Planning

(1) WireFrame

(2) Table Specification

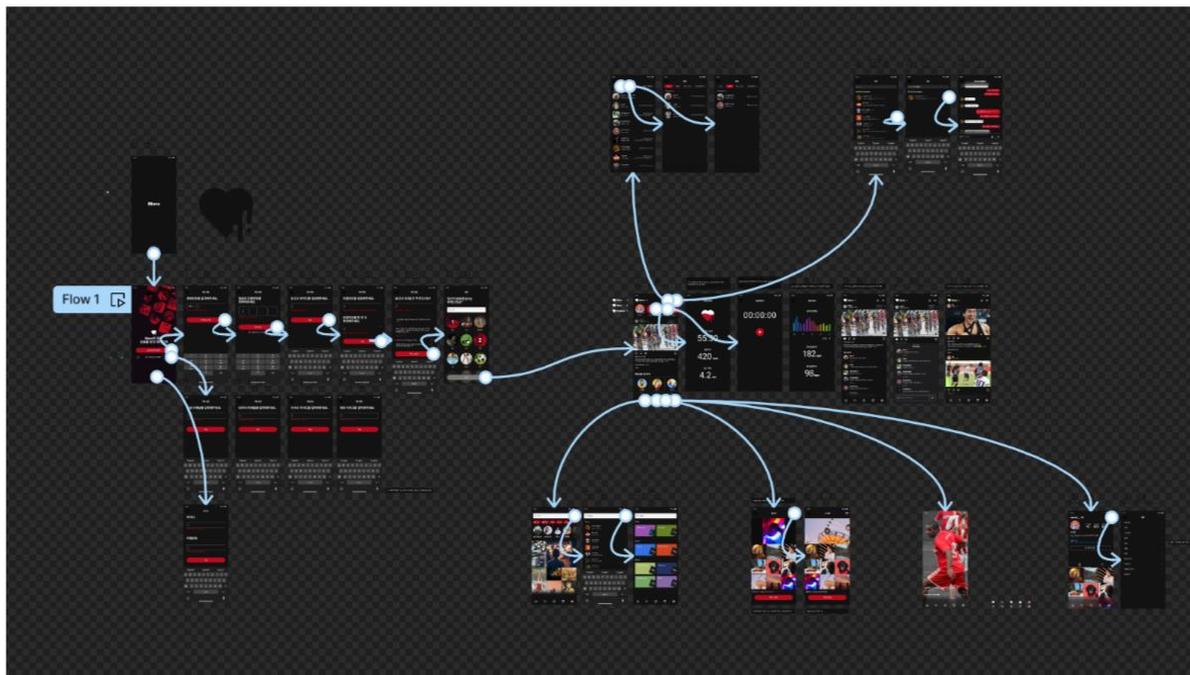
(3) API Document

3. Planning

(1) WireFrame

Figma 화면기획 자료

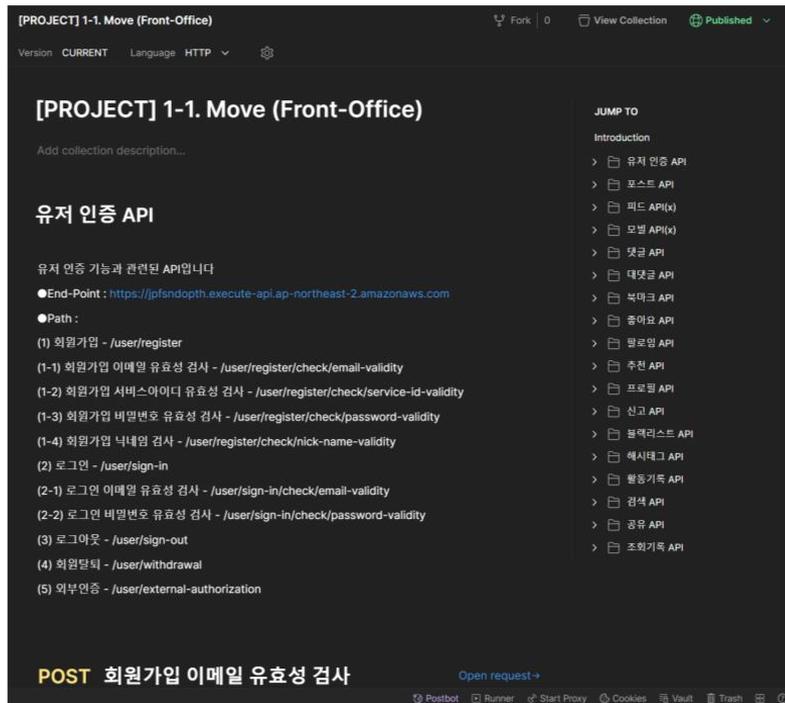
<https://www.figma.com/design/CF7wVH27RGYqj1dZzBB30B/Project-1.-Move?node-id=42-8&t=HfZA2HsYmwVr1wy4-1>



아이템 선정 후
Figma를 이용하여
Workspace 생성 뒤
팀 구성원 초대하여
edit 권한 부여 후
서비스의 **UI/UX 설계**하고
Prototyping 하여
WireFrame 작성하며 협업

3. Planning

(3) API Document



API 명세서 자료

<https://documenter.getpostman.com/view/35043583/2sA3dskDPn>

테이블 명세서를 바탕으로
API 개발하며
각 API 개발마다 POSTMAN으로
먼저 로컬에서 테스트 후
배포하여 서버 테스트 진행하고,
POSTMAN의 Document
기능 이용하여 API 명세서 작성

4. Development

(1) Project Management

(2) Development

(3) Continous Deployment

(4) Collaboration

4. Development

(1) Project Management

< Requirements >

시장 및 유사서비스 분석하여 서비스 **아이템 선정**



< Design >

아이템 선정 후 요구사항 분석하며 **화면기획서** 도출



< Front-End >

UI 설계



화면 개발



API 명세서 참고하여
바인딩



< Back-End >

테이블명세서



DB & 서버 & 인프라
아키텍처 구축

(CI/CD Pipe-Line 구축)



API 서버 개발



API 명세서 작성



Feedback



Review



Deploy



Test



개발 방식으로 **Agile Pattern**을 사용하여,
지속적으로 요구사항 분석 및 피드백 반영하여
개발 진행

4. Development

(2) Development

1. Firebase

DM기능 구현시

Firebase의 **RealTime DataBase** 사용



Front-End에서 Firebase와 연동하여
Instance 생성 후 **RealTime**으로 통신하면서,
데이터 변화 사항을 지속적으로 갱신하여,
UI와 연결하여 **Rendering**



데이터베이스에

각 채팅방의 메시지 기록을 저장하는 **chatRoom**,
각 해당 방에 대한 정보가 담겨 있는 **chatRooms** 존재
chatRoom의 중복 생성으로 인한 오류와 리소스 낭비를
방지하기 위해, 이미 chatRoom 존재시 방에 입장
API서버와 통신하며 **이미지 파일 정보는 URL**로 조회하여 Insert

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference chatRoomsRef = database.getReference(ChatRoomFragment.REFERENCE);

chatRoomsRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        boolean isFind = false;

        for (DataSnapshot roomSnapshot : dataSnapshot.getChildren()) {
            Integer userId1 = roomSnapshot.child("userId1").getValue(Integer.class);
            Integer userId2 = roomSnapshot.child("userId2").getValue(Integer.class);

            // 이미 방이 생성되어있는 경우
            if ((userId1 == mineUserId || userId1 == opponentUserId) || (userId2 == mineUserId || userId2 == opponentUserId)) {
                String roomId = "";

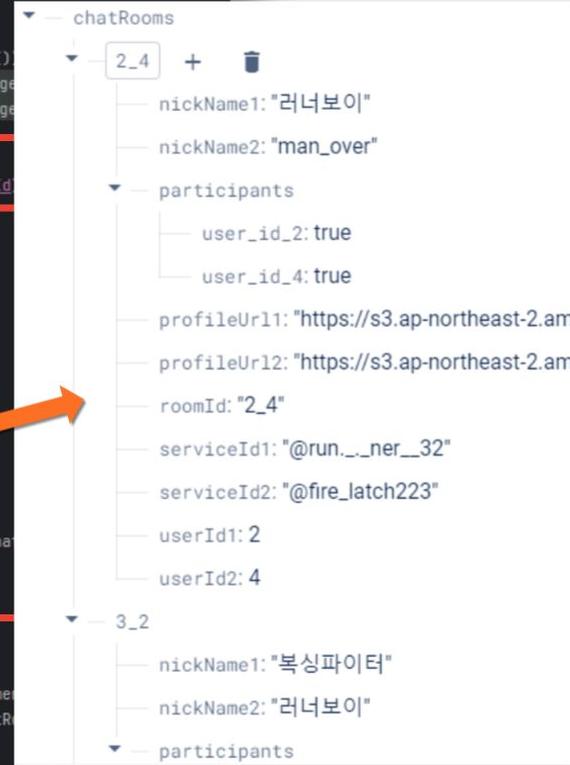
                if (userId1 == mineUserId) {
                    roomId = mineUserId + "_" + opponentUserId;
                }

                if (userId2 == mineUserId) {
                    roomId = opponentUserId + "_" + mineUserId;
                }

                isFind = true;
                ((MainActivity) context).navigateToFragment(new ChatRoomFragment(roomId, mineUserId));
            }

            // 방이 없다면 새로 생성
            if (!isFind) {
                ChatRoomFragment chatRoom = new ChatRoomFragment(mineUserId, opponentUserId);
                chatRoomsRef.child(chatRoom.getRoomId()).setValue(chatRoom);

                String roomId = mineUserId + "_" + opponentUserId;
                ((MainActivity) context).navigateToFragment(new ChatRoomFragment(roomId, mineUserId));
            }
        }
    }
});
```



4. Development

(2) Development

2. 연동 로그인

Client에서 Google 연동 로그인 API 호출시
로그인 URL로 Redirect



App 화면에서

Intent 이용하여 구글 로그인 화면으로 전환



Google 로그인 완료시
Client에게 Access Token 발급

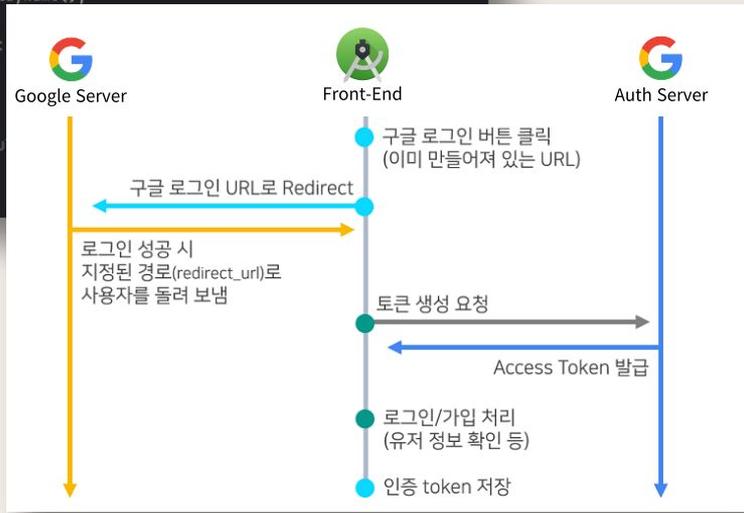


Google에서 해당 유저의
ID, E-mail, Name 정보 조회하여
Move API 서버로 외부 인증 API 호출해서
기존 회원 → 로그인
신규 회원 → 회원가입

```
no usages ± ph20531
public void initGoogleLogin(Activity activity) {
    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGNING_IN_OPTIONS)
        .signInFlow(GoogleSignInOptions.Flow.SilentForWeb)
        .requestEmail(true)
        .build();
    googleSignInClient = GoogleSignIn.getClient(activity, gso);
}

no usages ± ph20531
public void googleLogin(Activity activity) {
    Intent signInIntent = googleSignInClient.getSignInIntent();
    activity.startActivityForResult(signInIntent, RC_SIGN_IN);
}

no usages ± ph20531
public void handleGoogleSignInResult(Intent data) {
    Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
    try {
        GoogleSignInAccount account = task.getResult(ApiException.class);
        if (account != null) {
            id = account.getId();
            email = account.getEmail();
            displayName = account.getDisplayName();
            Log.d("GoogleLogin", "ID: " + id);
            Log.d("GoogleLogin", "Email: " + email);
            Log.d("GoogleLogin", "Name: " + displayName);
        }
    } catch (ApiException e) {
        Log.w("GoogleLogin", "signInResult: failed, " + e.getMessage());
    }
}
```



4. Development

(2) Development

3. Batch Processing

여러 API를 호출하여 하나의 화면에 대한 UI Rendering시

API마다 Response 시간이 상이하므로, 일괄처리하여 렌더링하기 위해서

Java의 `CountDownLatch` 클래스 사용



(1) 각 API가 Response될 때마다 countDown 누게

(2) 모든 API가 호출될 때까지 `await`으로 비동기 처리

(3) 모든 API 호출하여 데이터 Response 완료 후

일괄처리하여 `Rendering`하는 방식으로

Batch Processing 수행

```
new Thread() -> {
    try {
        latch.await();
        mainLatch.countDown();

        if (post.getProfileUrl() != null) {
            post.setType(Post.TYPE.THUMBNAIL);
        } else {
            if (DummyData.IS_PROFILE_DEFAULT TYPOGRAPHY)
                post.setType(Post.TYPE.TYPOGRAPHY);
            else {
                post.setType(Post.TYPE.ICON);
                post.setIconDrawable(R.drawable.default_icon);
            }
        }

        data.add(post);

        int index = data.size() - 1;
        if (index != 0 && index % RECOMMEND_FREQUENCY == 0) {
            getRecommendUser((totalCount1, itemTotalCount1,
                Post recommendData = new Post((ArrayList<Rec
                data.add(index, recommendData);

            postAdapter.notifyItemInserted(index);

            int size = itemTotalCount - 1;
            if (size != index)
                postAdapter.notifyItemRangeChanged(positio
                }, RECOMMEND_INIT_OFFSET, RECOMMEND_LIMIT);
        }

    } catch (InterruptedException e) {
        Toast.makeText(context, text: "Error: interrupted while waiting for re
    }
}).start();
}
```

```
API.Method.getParentCommentCount(context, accessToken, feed.getPostId(), latch, new API
21 usages ph20531
@Override
public void onSuccess(Object response) {
    API.Comment.ParentCommentCountResponse parentCommentCountResponse = (API.Comme
    post.setCommentCount(parentCommentCountResponse.getCommentCount());
}

42 usages ph20531
@Override
public void onError(String errorMessage) {
}

ph20531
@Override
public void onFailure(String throwMessage) {
}

ph20531
API.Method.isFollow(context, accessToken, feed.getUserId(), latch, new API.ResponseCal
21 usages ph20531
@Override
public void onSuccess(Object response) {
    API.Follow.IsFollowResponse isFollowResponse = (API.Follow.IsFollowResponse) r
    post.setFollow(Utilities.integerToBoolean(isFollowResponse.getExistence()));
}
}
```

```
final CountDownLatch latch = new CountDownLatch(0);

ph20531 +1
API.Method.getUserInforma
21 usages ph20531
@Override
public void onSuccess
API.User.Informat
post.setProfileUr
post.setNickName(
post.setServiceId
}

ph20531
@Override
public void onResponse @NonNull Call<A
if (response.body() != null) {
    API.Bookmark.IsBookmarkRespons
    if (response.isSuccessful() &&
        String message = isBookmarkR
        Toast.makeText(context, "suc
        responseCallback.onSuccess
        latch.countDown();
    }
}
```

4. Development

(2) Development

4. UI Thread Synchronization

팔로우, 좋아요, 북마크 등 기능 동작시 Data Binding 기법
이용하여 각 데이터와 어댑터에 접근하여 데이터 수정 후
notifyItemChanged 메서드를 호출하여 UI 업데이트
하여 State 관리하면서 UI 동기화



UI 동기화 작업 미수행시 동일 데이터에 대한
컴포넌트가 서로 다른 아이템에서 상태를 공유하지 않음

ex) 피드 목록 조회 중

"복싱파이터" 유저를 언팔로우 했는데, 영상 목록

조회시 동일한 "복싱파이터" 유저에 대해

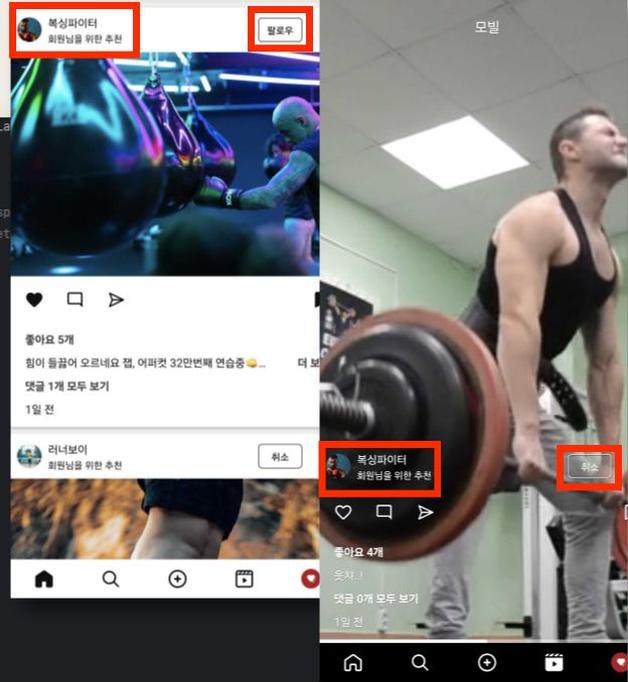
언팔로우 상태가 표시 되지 않음

```
API.Method.registerFollow(context, accessToken, item.getUserId(), new CountDownLa
30 usages · ph20531*
@Override
public void onSuccess(Object response) {
    API.Follow.RegisterResponse registerResponse = (API.Follow.RegisterResp
    Toast.makeText(context, item.getUserId() + " : " + registerResponse.get
    for (int i = 0; i < data.size(); i++) {
        Post target = data.get(i);
        if (item.getUserId() == target.getUserId()) {
            target.setFollow(true);
            postAdapter.notifyItemChanged(i);
        }
    }
}

for (int i = 0; i < feedData.size(); i++) {
    Post feed = feedData.get(i);
    if (feed.isRecommend()) {
        ArrayList<Recommend> recommendData = feed.getRecommendData();
        for (int j = 0; j < recommendData.size(); j++) {
            Recommend target = recommendData.get(j);
            if (item.getUserId() == target.getUserId()) {
                target.setFollow(true);
                postAdapter.notifyItemChanged(i);
            }
        }
    }
}
}
```

```
MobilFragment mobilFragment = (MobilFragment) ((MainActivity) context).getMainFragment().getFragmentByName(MobilFragment.class.getSimpleName());
if (mobilFragment != null) {
    ArrayList<MobilFragment.Mobil> mobilData = mobilFragment.getData();
    if (mobilData != null) {
        for (int i = 0; i < mobilData.size(); i++) {
            MobilFragment.Mobil target = mobilData.get(i);
            if (item.getUserId() == target.getUserId()) {
                target.setFollow(true);

                MobilFragment.MobilAdapter mobilAdapter = mobilFragment.getMobilAdapter();
                if (mobilAdapter != null)
                    mobilAdapter.notifyItemChanged(i);
            }
        }
    }
}
```



4. Development

(2) Development

6. Exception Handler

Exception 발생해도 프로세스가 종료되지 않도록
`Thread.setDefaultUncaughtExceptionHandler`를

사용



애플리케이션 레벨에서 기본 미처리 예외 핸들러를
설정하여 애플리케이션이 종료되지 않고,
발생하는 예외를 로깅하도록 설계

ex)

날짜 파싱 형식 불일치하여

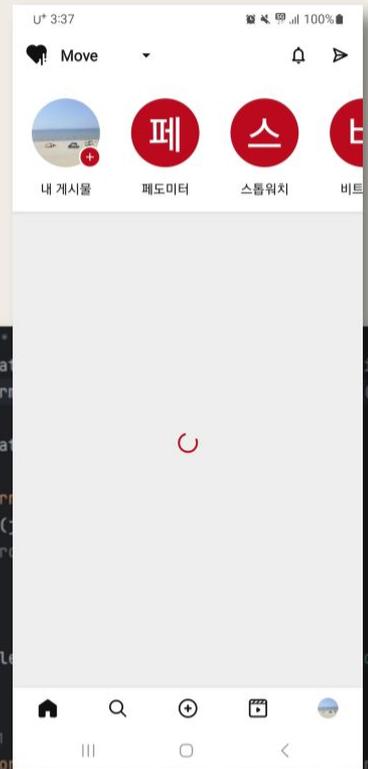
```
throw new IllegalArgumentException("Invalid  
date format: " + dateTimeString);
```

예외 발생으로 애플리케이션 종료돼야 하지만
→ 예외 무시하고 Log를 남긴 후 계속 로딩(동작)

```
@Override  
public void onCreate() {  
    super.onCreate();  
  
    FirebaseApp.initializeApp(context, this);  
    FirebaseDatabase.getInstance().setPersistenceEnabled(true);  
  
    // 캐시 설정 (16B)  
    cache = new SimpleCache(new File(getCacheDir(), child: "media"  
  
    // 어플리케이션에서 모든 익셉션 발생 시  
    // 어플리케이션이 kill process되지않고 로그만 찍고 넘어가는 코드  
    Thread.setDefaultUncaughtExceptionHandler((t, e) -> {  
        Log.d(tag: "Caught Exception", e.getMessage());  
    });  
}
```

LoginFragment	202
MainActivity	203
MainFragment	204
MobilFragment	205
ProfileFragment	207
SearchFragment	208
ShowHideNestedScrollListener	209
ShowHideScrollListener	210
SignUpEmailFragment	211

```
Logcat Logcat x com.eastwise1210.yh_finalproject_1_android (R5CX51BP... x  
samsung SM-A156L (R5CX51BP2DA) Android 14, API 3 Caught Exception  
2024-08-05 15:35:48.884 17239-17239 Caught Exception com...210.yh_finalproject_1_android D Illegal pattern
```



4. Development

(2) Development

7. Utility Class

Retrofit 라이브러리로 API 서버와 네트워킹시
사용 빈도가 높은 메서드들

→ API 클래스의 이너클래스인 **Method** class에
모아서 개발 및 관리 **효율성 ↑**



① 해당 API 사용 시 **API.Method**에서
호출하여 사용

② **ResponseCallback** 인터페이스를 통해
Front에서 별도로 임의의 success, error, failure
처리 가능토록 구현

```
Android
├── app
│   ├── manifests
│   │   └── AndroidManifest.xml
│   └── java
│       └── com.eastwise1210.yh_finalproject_1.android
│           ├── Account
│           ├── AccountFragment
│           └── API
│               ├── Authorization
│               ├── ChatFragment
│               ├── ChatRoomFragment
│               ├── CommentDialogFragment
│               ├── CreatePostFragment
│               ├── DummyData
│               ├── EditProfileFragment
│               ├── EmptableRecyclerView
│               ├── HashTagUtils
│               ├── HomeFragment
│               ├── MainActivity
│               ├── MainFragment
│               ├── MobilFragment
│               ├── MoveApplication
│               ├── ProfileFragment
│               ├── SearchFragment
│               ├── ShowHideNestedScrollListener
│               ├── ShowHideScrollListener
│               ├── SignUpEmailFragment
│               └── ...
└── ...

API.java
1543
1544
1545 public interface LoadedCallback {
1546     6 usages ─ kanghung95
1547     void onLoad(int totalCount, int itemTotalCount);
1548 }
1549
1550 public interface ChildLoadedCallback {
1551     2 usages ─ kanghung95
1552     void onLoad(int totalCount, int itemTotalCount, Object child);
1553 }
1554
1555 public static class Method {
1556     4 usages ─ cheeze +1
1557     public static void getSelfInformation(Context context, String access
1558         API.ApiService apiService = API.getInstance().create(API.ApiSer
1559 }
1560
1561 105 usages 75 implementations ─ ph20531
1562 public interface ResponseCallback {
1563     30 usages 75 implementations ─ ph20531
1564     void onSuccess(Object response);
1565 }
1566
1567 60 usages 75 implementations ─ ph20531
1568 void onError(String errorMessage);
1569 }
1570
1571 75 implementations ─ ph20531
1572 void onFailure(String throwMessage);
1573 }
1574
1575 6 usages ─ ph20531 +1
1576 public interface LoadedCallback {
1577     6 usages ─ kanghung95
1578     void onLoad(int totalCount, int itemTotalCount);
1579 }
1580
1581 2 usages ─ ph20531 +1
1582 public interface ChildLoadedCallback {
1583     2 usages ─ kanghung95
1584     void onLoad(int totalCount, int itemTotalCount, Object child);
1585 }

API.Method.isFollow(context, accessToken, r...
30 usages ─ ph20531
@Override
public void onSuccess(Object response) {
    API.Follow.IsFollowResponse isFollow
    recommend.setFollow(Utilities.intege
}
```


4. Development

(3) Continuous Deployment

AWS, Serverless, Docker를
이용한 CD 구축

1. AWS IAM 사용자 생성

① 외부 app이 AWS 서비스에 접근할 수 있도록

Credential 정보 설정을 위해 **Access Key**,
Secret Key를 발급 받는다

② AWS 서비스 접근 및 이용을 위해

해당 **권한 정책**을 연결한다

2. AWS ECR 리포지토리 생성

- 어떤 IAM 사용자에게 연동할건지 **IAM 개체 지정**

- ECR 서비스 접근시 이용할 수 있는 권한에 대해

정책 JSON 편집

The screenshot displays the AWS IAM console interface. At the top, the user 'project-server' is selected, showing its ARN and console access status. Below this, a '리포지토리 (4)' (Repositories) table lists four ECR repositories: 'serverless-autumn-server', 'serverless-berrycake-server', and 'serverless-move-server'. A red arrow points from the 'serverless-move-server' repository to the '권한 정책 (10)' (Policies) section. This section shows a list of policies, with 'AmazonEC2ContainerRegistryFullAccess' highlighted. A second red arrow points from this policy to the 'JSON 편집' (JSON Editor) window, which displays the policy's JSON configuration. The JSON includes a 'Principal' block with the user's ARN and an 'Action' block listing various ECR actions like 'BatchCheckLayerAvailability', 'BatchDeleteImage', etc. At the bottom right, there are buttons for '재설정' (Reset) and '저장' (Save).

4. Development

(3) Continuous Deployment

AWS, Serverless, Docker를
이용한 CD 구축

3. Serverless 프레임워크 실행

① Flask API 선택하여 프로젝트 생성

4. Serverless ~ AWS Credential 정보 설정

① 해당 IAM 사용자 보안정보 중 **Access key**와

Secret key 이용하여, 배포시 AWS IAM

사용자에 접근 가능하도록 설정

5. serverless.yml 파일 설정

① provider 부분에서 **runtime**과 **region** 지정

② runtime은 본인 개발환경 중 AWS Lambda에서
지원하는 버전으로 지정

③ region 미 설정시 us-east-1으로 자동 지정되므로
한국 리전에 해당하는 ap-northeast-2로 지정

6. \$ serverless deploy

① Credential 설정 및 생성된 프로젝트 테스트용으로 시험 배포

```
serverless.yml X
serverless.yml
1  service: [redacted]
2
3  frameworkVersion: '3'
4
5  custom:
6    wsgi:
7      app: app.app
8
9  provider:
10   name: aws
11   runtime: python3.10
12   region: ap-northeast-2
13
14  functions:
15   api:
16     handler: wsgi_handler.handler
17     events:
18       - httpApi: '*'
19
20  plugins:
21   - serverless-wsgi
22   - serverless-python-requirements
23
```

```
C:\Users\Fletcher\Documents\GitHub> serverless
Running "serverless" from node_modules

Creating a new serverless project

? What do you want to make?
AWS - Node.js - Starter
AWS - Node.js - HTTP API
AWS - Node.js - Scheduled Task
AWS - Node.js - SQS Worker
AWS - Node.js - Express API
AWS - Node.js - Express API with DynamoDB
AWS - Python - Starter
AWS - Python - HTTP API
AWS - Python - Scheduled Task
AWS - Python - SQS Worker
> AWS - Python - Flask API
AWS - Python - Flask API with DynamoDB
Other
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(lambda_310) C:\Users\Fletcher\Documents\GitHub\test> serverless deploy
Running "serverless" from node_modules

Deploying test to stage dev (ap-northeast-2)
Python executable not found for "runtime": python3.10
Using default Python executable: python
Packaging Python WSGI handler...
Creating CloudFormation stack (10s)
```

```
C:\Users\Fletcher\Documents\GitHub> serverless config credentials --provider aws --key [redacted] --secret [redacted]
Running "serverless" from node_modules
✓ Profile "default" has been configured
C:\Users\Fletcher\Documents\GitHub>
```

4. Development

(3) Continuous Deployment

AWS, Serverless, Docker를
이용한 CD 구축

7. 파일 설정

- ① **dockerignore** 파일 설정을 통해 docker image 빌드시 프로젝트 내 파일 및 디렉토리 중 제외할 것들 지정
- ② **Dockerfile** 설정
- ③ **requirements.txt** 파일에 개발환경시 사용한 라이브러리 및 해당 버전 기재하여 docker image 빌드시 포함할 라이브러리 명시
- ④ docker를 이용한 배포에 맞게 **serverless.yml** 파일 설정
- **provider** 지정하여 AWS Lambda 생성시 **Runtime** 환경, 배포 **region**, AWS ECR 리포지토리 명시
- ECR에 Push 된 어떤 Docker Image로 Lambda Function 생성할지 명시
- ⑤ 해당 설정에 대해 배포 점검을 위해 Local에서 Docker 소프트웨어 실행 상태로 **\$ serverless deploy** 실행

```
serverless.yml
1 service: serverless-move-server
2
3 frameworkVersion: '3'
4
5 custom:
6   wsgi:
7     app: app.app
8
9 provider:
10  name: aws
11  runtime: python3.12
12  region: ap-northeast-2
13  ecr:
14    images:
15      appimage:
16        path: ./
17
18 functions:
19  app:
20    image:
21      name: appimage
22      timeout: 150
23    events:
24      - httpApi: '*'
25
```

```
Dockerfile
1 FROM public.ecr.aws/lambda/python:3.12
2 COPY . ${LAMBDA_TASK_ROOT}
3 COPY requirements.txt .
4
5 RUN yum -y install gcc
6 RUN pip3 install -r requirements.txt --target "${LAMBDA_TASK_ROOT}"
7
8 CMD ["app.handler"]
```

```
.dockerignore
1 # .dockerignore
2 __pycache__/
3 .git/
4 .serverless/
5 .dockerignore
6 serverless.yml
```

```
Running "serverless" from node_modules
Deploying serverless-autumn-server to stage dev (ap-northeast-2)
(lambda_310) C:\Users\Fletcher\Documents\GitHub\serverless-autumn-server> serverless deploy
Running "serverless" from node_modules
Deploying serverless-autumn-server to stage dev (ap-northeast-2)
Packaging (3s)
Building image "appimage"
```

4. Development

(3) Continuous Deployment

AWS, Serverless, Docker를
이용한 CD 구축

8. GitHub 원격 리포지토리 연동

- 1 Serverless 프레임워크로 생성한 Flask 프로젝트 폴더를
\$ git init으로 Local 리포지토리로 한 후, GitHub의 원격
리포지토리와 연동

9. GitHub Actions 설정

- 1 GitHub Actions를 이용한 자동 배포 액션 수행시 **serverless**
프레임워크와 AWS 서비스 접근 위해서 총 3개의 ACCESS KEY
정보 저장
- 2 해당 리포지토리에 특정 이벤트 발생시 GitHub Actions가
수행할 **Workflow**에 대해 **.yaml파일로 설정**
-트리거 : Push 이벤트 / **브랜치** : main
-스텝 & 잡 : actions checkout → install python==3.12
→ install serverless@3.38.0 → \$ serverless deploy 수행

The image shows a GitHub Actions workflow configuration and repository secrets. The workflow file is located at `serverless-move-server / github / workflows / main.yml`. The repository secrets are:

- SERVERLESS_ACCESS_KEY
- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY
- SERVERLESS_ACCESS_KEY

The workflow configuration is as follows:

```
name: Deploy sls app
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    env:
      SERVERLESS_ACCESS_KEY: ${ secrets.SERVERLESS_ACCESS_KEY }
      AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
      AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
    steps:
      - uses: actions/checkout@v3
      - name: install-python
        uses: actions/setup-python@v4
        with:
          python-version: '3.12'
      - name: install serverless
        run: npm i -g serverless@3.38.0
      - name: serverless deploy
        run: sls deploy --verbose --force
```

4. Development

(3) Continuous Deployment

AWS, Serverless, Docker를

이용한 CD 구축

10. 자동 배포 테스트

① GitHub Actions 트리거 설정한대로,

해당 리포지토리 Main 브랜치에 Push

이벤트 발생시켜서 Workflow 테스트 & 점검

② CD 구축 확인

The screenshot shows the GitHub Actions interface for a workflow named 'main.yml'. The workflow is triggered by a push to the 'main' branch. The 'deploy' job is highlighted with a red box, indicating it is the current focus. The job status is 'Success' and it took 3m 57s to complete. Below the job summary, a list of steps is shown, with the 'Post Run actions/checkout@v3' step expanded to show its logs. The logs include commands for setting up the environment, installing Python, and configuring Git for a serverless deployment.

Deploy sfs app

✓ Create main.yml #1

Summary

Jobs

✓ deploy

Run details

Usage

Workflow file

main.yml

on: push

✓ deploy 3m 57s

deploy

succeeded last month in 3m 57s

Search logs

- Set up job
- Run actions/checkout@v3
- install-python
- install serverless
- serverless deploy
- Post install-python
- Post Run actions/checkout@v3

```
1 Post_job cleanup.
2 /usr/bin/git version
3 git version 2.45.2
4 Temporarily overriding HOME="/home/runner/work/_temp/0a172fd1-3ac9-449c-9114-02f40c4202dc" before making global git config changes
5 Adding repository directory to the temporary git global config as a safe directory
6 /usr/bin/git config --global --add safe.directory /home/runner/work/serverless-move-server/serverless-move-server
7 /usr/bin/git config --local --name-only --get-regexp core.sshCommand
8 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core.sshCommand' && git config --local --unset-all 'core.sshCommand'
9 /usr/bin/git config --local --name-only --get-regexp http.https://github.com/.extraheader
10 http.https://github.com/.extraheader
11 /usr/bin/git config --local --unset-all http.https://github.com/.extraheader
12 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'http.https://github.com/.extraheader' && git config --local --un
"http.https://github.com/.extraheader" | | :"
```

✓ Complete job

- Cleaning up orphan processes

4. Development

(4) Collaboration

< Slack >

협업 Tool 중

커뮤니케이션 목적으로 **Slack** 사용

(1) **워크스페이스** 생성 후 팀원 초대

(2) **채널 분리**하여 관리 효율 ↑

(3) **서드파티 앱** 연결하여 해당 채널에 알림

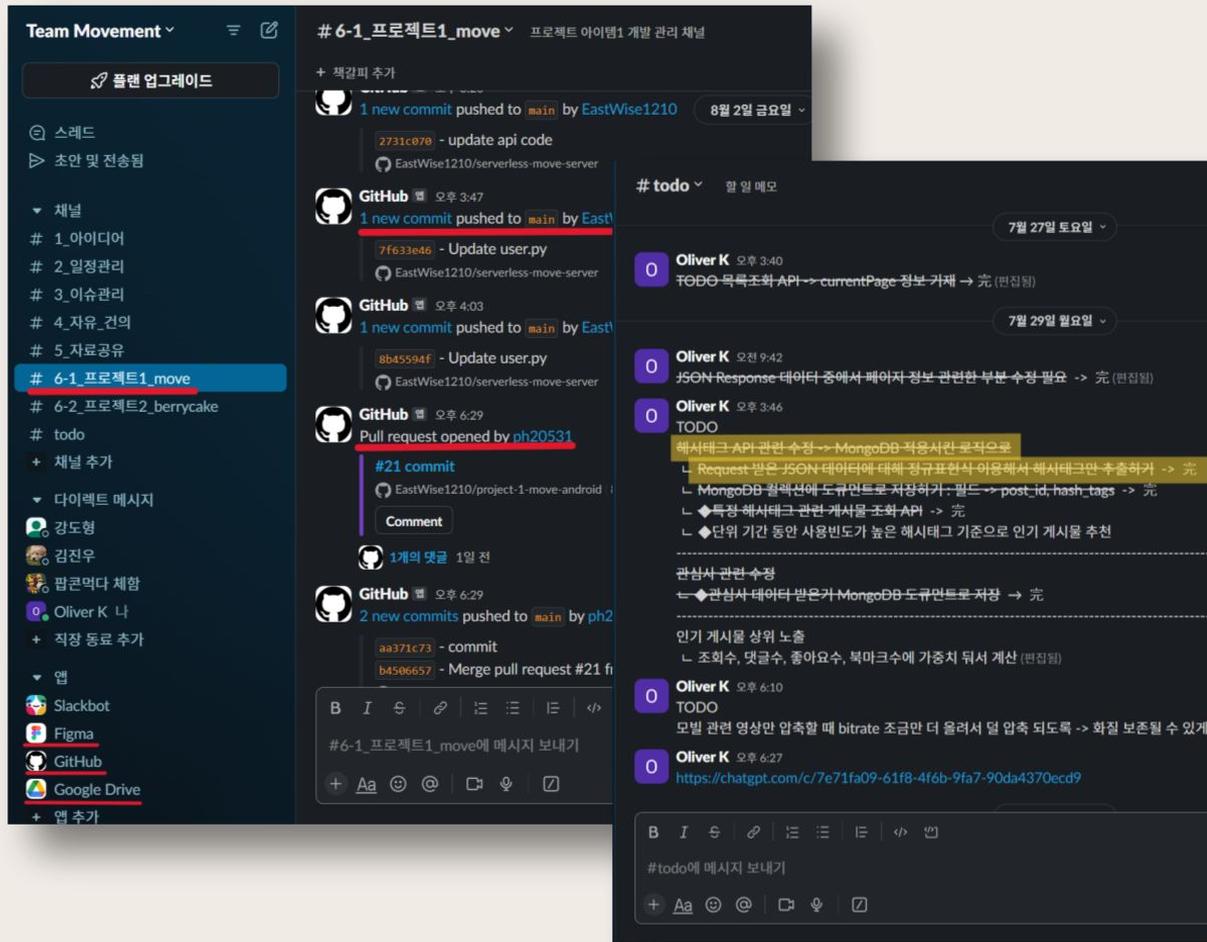
ex) **GitHub** 리포지토리에

Commit, Push, Pull Request, Merge 등

이벤트 발생시 알림 생성

(4) 팀원들과 논의하여 **TODO 사항** 체크 후

진행 및 **해결상황 공유**



4. Development

(4) Collaboration

< Figma >

협업 Tool 중

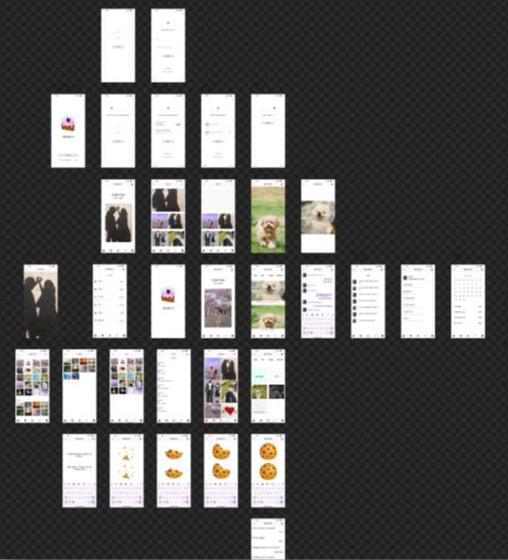
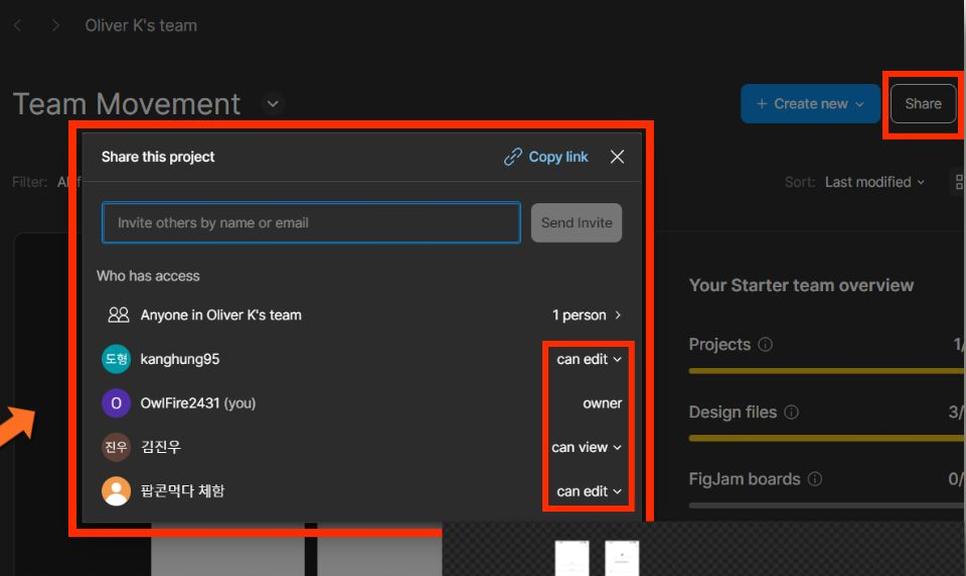
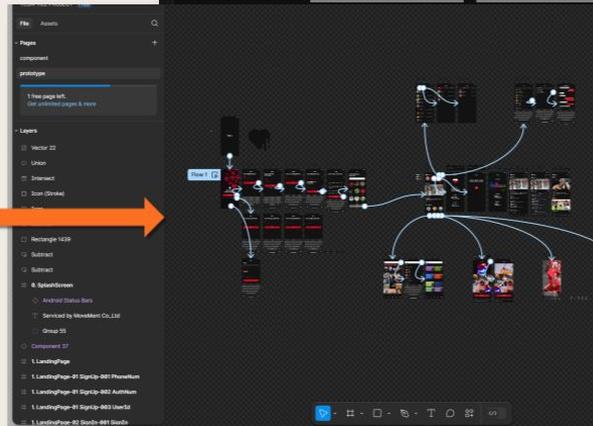
화면기획 및 Prototyping 목적으로 Figma 사용

- (1) Workspace 생성 후 팀원에게 Share
- (2) 역할에 맞게 권한(can edit / view) 부여
- (3) 화면 설계 후 Prototyping하여

WireFrame 작성

- (4) 해당 Design File에 대해 Share 허용해서

Slack에 링크로 공유



4. Development

(4) Collaboration

< GitHub >

협업 Tool 중

소스코드 관리 및 협업 목적으로 **GitHub** 사용

(1) 여러 **Repository** 생성 후 팀원 **Invite**

(2) 담당 파트에 맞게 해당 레포지토리에 작업

(3) 리포지토리의 소스코드 **Pull**한 다음

자신의 Branch를 만들어서 코드 개발

→ 단위 테스트 후 **Commit & Push**

→ **Pull Request** 생성하여 팀원 전체 공동

코드리뷰 및 검토 진행 후 메인 브랜치에 **Merge**

The screenshot displays the GitHub interface for a repository named 'project-1-move-android'. The 'Collaborators' tab is selected, showing a list of collaborators: agjin9613, kanghung95, and Cheeze. The 'Manage access' section is highlighted with a red box, showing the 'Add people' button and a search bar. Below this, a list of collaborators is shown with their commit counts and a red box highlights the '11 Pull request' count. The 'Overview' section shows 11 Active pull requests and a commit history with a red box highlighting '11 Pull request'. The bottom right shows a commit history with a red box highlighting '11 Pull request' and a commit by 'EastWise1210'.

4. Development

(4) Collaboration

< Postman >

협업 Tool 중 API 테스트 목적 외

API 작업시 협업 목적으로도 Postman 사용

(1) 팀 워크스페이스 생성 후 팀원 Invite

(2) Invite에 팀원 accept 후 역할에 따라

Editor & Viewer 권한 설정

(3) API 관련 수정사항 발생시 바로 공유 가능

(4) 해당 Folder에 대해 View Document

→ API Document 작성 → Publish 처리하여

링크 공유해서 공유 및 협업

※API 테스트시, 먼저 Local 환경에서 테스트 후

서버 배포하여 서버 환경에서 테스트

The screenshot displays the Postman interface with several key elements highlighted in red boxes:

- Invite Button:** Located in the top right corner of the main interface.
- Published Button:** Located in the top right corner of the workspace view.
- Workspaces Panel:** Located in the bottom left, showing a search bar and a list of workspaces including 'Movement Workspace' and 'My Workspace'.
- Invite Dialog:** A modal window titled 'Invite people to this workspace' is open, showing an input field with the email 'kanghung95@gmail...', a dropdown menu for 'Role' set to 'Admin', and a list of roles with their permissions.

The dialog content includes:

- Name, email, or group name: Add from file
- Role: Admin
- Admin: Can manage workspace details and members.
- Editor: Can create and edit workspace resources.
- Viewer: Can view, fork, and export workspace resources.

5. Testing

(1) Back-End Unit Test

(2) Front-End Unit Test

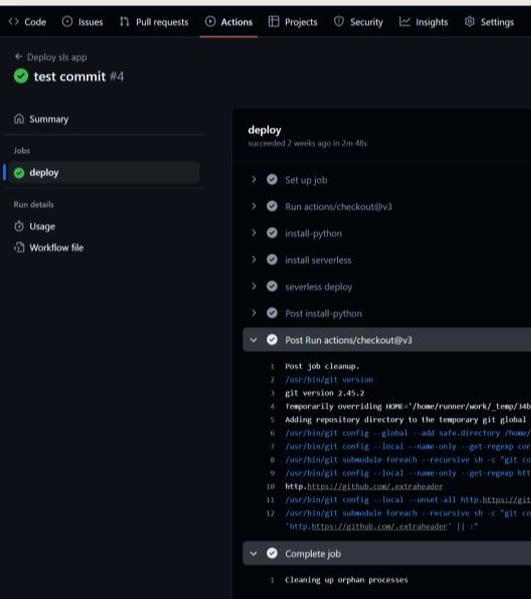
(3) Integration Test

(4) System Test

5. Testing

(1) Back-End Unit Test

(오류 발생시 Terminal 로그 조회하여 디버깅)

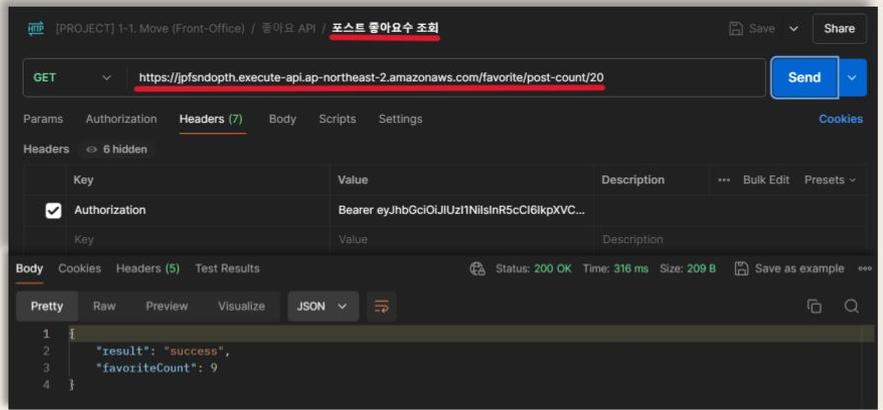
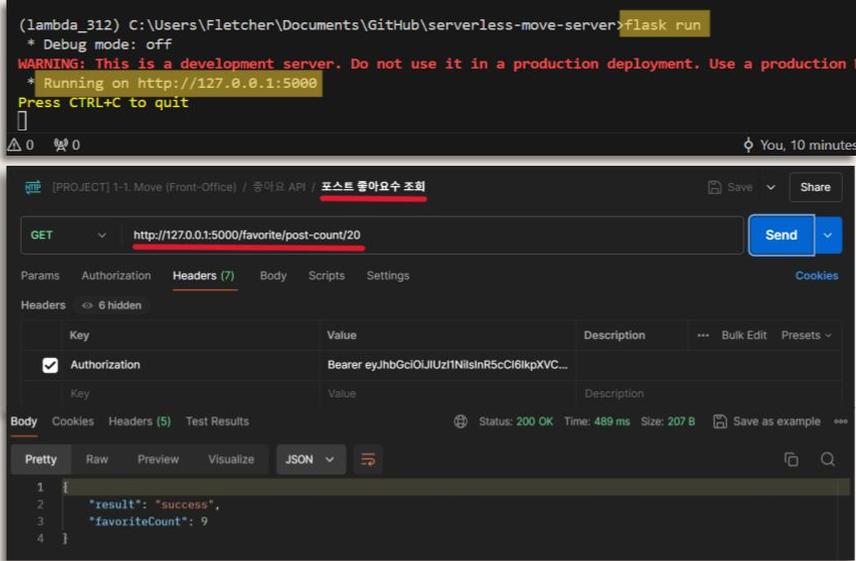


2. Local 환경에서
테스트 완료 후 GitHub에
Commit&Push하여
Push 이벤트 발생시켜서
GitHub Actions로
미리 설정한 자동배포
Workflow 실행하여
서버 배포

3. 배포 완료 확인 후
서버 환경에서
Request 발생시켜서
API 기능 테스트

(오류 발생시 AWS CloudWatch 로그 조회하여 디버깅)

1. Local 환경에서
터미널에 flask run 명령어로
서버 구동하여
Request 발생시켜서
API 기능 테스트

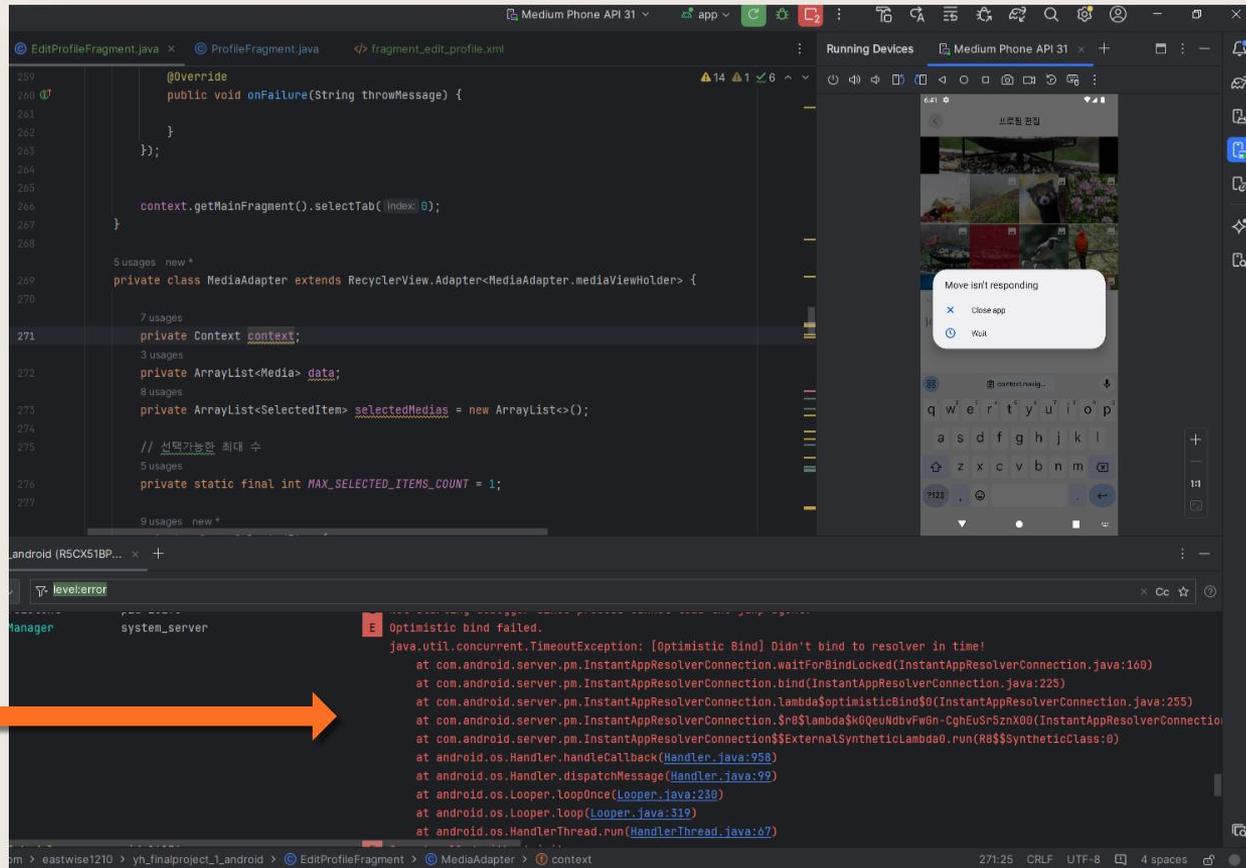


5. Testing

(2) Front-End Unit Test

Device Manager를 이용하여
Emulator를 생성한 다음
단위 기능 구현시마다
run 'app'으로 에뮬레이터를
실행시켜서 에뮬레이터 화면으로
코드 동작 상태 확인

버그 발생시 Terminal에서
로그 확인하여 디버깅



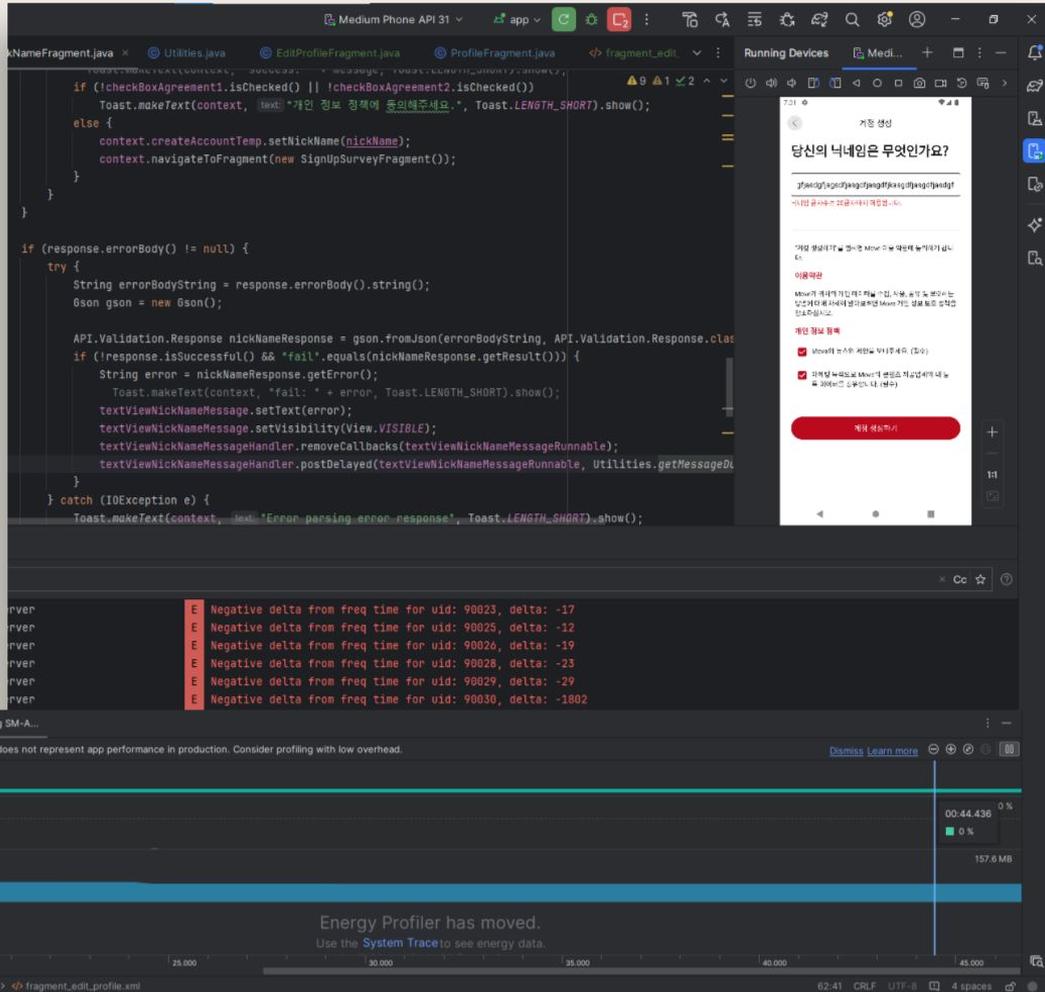
5. Testing

(3) Integration Test

Retrofit 라이브러리를 이용하여
AWS 배포중인 API 서버와 네트워킹하며
Front-End ~ Back-End
바인딩 작업 수행시

① API 명세서 참고하여 데이터 요청시 **엔드포인트**
및 **데이터 유효성, 에러핸들** 점검

② 안드로이드 스튜디오의 **Memory Profiler**
이용하여 **메모리 누수** 여부 점검



5. Testing

(4) System Test

통합테스트 후

다양한 하드웨어 환경에서

시스템 테스트 진행



Galaxy Note9 기종에서

로그인 화면 UI 비율이

깨지는 현상 발견 외 구동 이상X



Galaxy Z-Flip4

Galaxy S21

Galaxy Note9

Galaxy Buddy3

Galaxy Note8

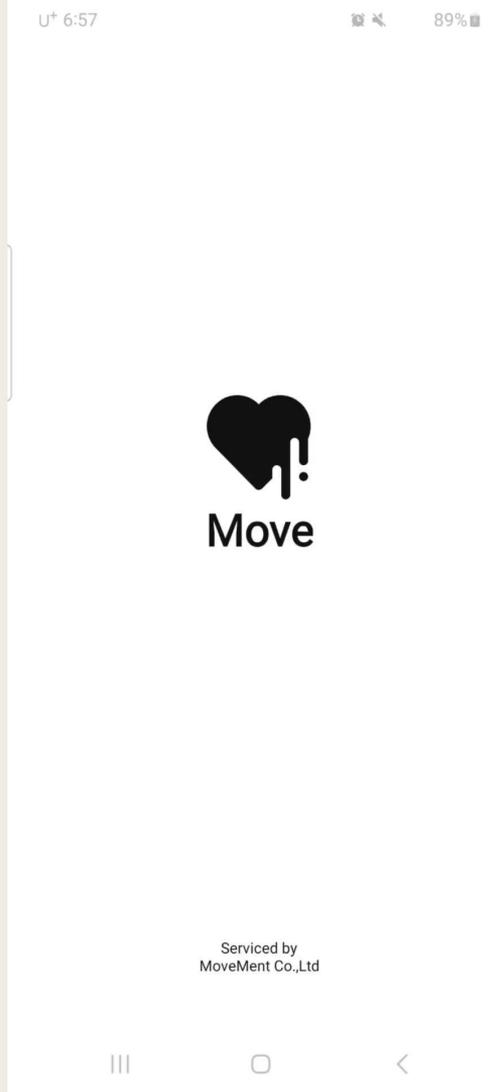
프로세서	Snapdragon 8+ Gen1	삼성 엑시노스 2000 Series (2100)SoC.	삼성 엑시노스 9 Series (9810)SoC.	미디어텍 Dimensity 6100+	삼성 엑시노스 9 Series (8895)SoC.
메모리	8GB LPDDR5 SDRAM/ 256GB내장메모리	8GB LPDDR5 SDRAM/ 256GBUFS 3.1 규격 내장 메모리	6GB LPDDR4XSDRAM/ 128GBUFS 2.1 규격 내장 메모리	6GB LPDDR4XSDRAM/ 128GBUFS 2.2 내장 메모리	6GB LPDDR4XSDRAM/ 64GBUFS 2.1 규격 내장 메모리
디스플레이	6.7인치 슈퍼 아몰레드 패널 FHD+ 커버 디스플레이 2인치	6.2인치다이내믹아몰레드 2X	6.4인치인피니티디스플레이	6.5인치 슈퍼 아몰레드 인피니티-U디스플레이	6.3인치인피니티디스플레이
운영체제	Android 12 (Samsung One UI 4.1.1)	Android 14 (Samsung One UI 6.1)	Android 10 (Samsung One UI 2.5)	Android 14 (Samsung One UI 6.1)	Android 9 (Samsung One UI 1.0)

6. Result

(1) Service Demo Video

6. Result

(1) Service Demo Video



7. Issues

(1) Front-End Issue 1

(2) Front-End Issue 2

(3) Back-End Issue 1

7. Issues

(1) Front-End Issue 1

Retrofit으로 한 번에 여러개의 API를 Request

→ API마다 Response 시간이 다르기 때문에

→ 일괄처리하여 렌더링 할 때 UI에 문제 발생



Java의 **CountDownLatch** 클래스 사용

→ 각각 API가 Response될 때 마다 countDown 누계

→ 모든 API가 호출되어 countDown이 끝날 때까지 **await**

→ 모든 API 호출 완료 후일괄처리하여 렌더링 방식으로 해결

```
final CountDownLatch latch = new CountDownLatch(6);

API.Method.getUserInformation(context, accessToken, feed.getUserId());
21 usages ± ph20531
@Override
public void onSuccess(Object response) {
    API.User.InformationResponse response = (API.User.InformationResponse) response;
    post.setProfileUrl(informationResponse.getProfileUrl());
    post.setNickName(informationResponse.getNickName());
    post.setServiceId(informationResponse.getServiceId());
}
42 usages ± ph20531
@Override
public void onError(String errorMessage) {}

API.Method.getParentCommentCount(context, accessToken, feed.getPostId(), latch, new API.Callback() {
    21 usages ± ph20531
    @Override
    public void onSuccess(Object response) {
        API.Comment.ParentCommentCountResponse parentCommentCountResponse = (API.Comment.ParentCommentCountResponse) response;
        post.setCommentCount(parentCommentCountResponse.getCommentCount());
    }
}
42 usages ± ph20531
@Override
public void onError(String errorMessage) {}

API.Method.getBookmark(context, accessToken, feed.getUserId(), latch, new API.Callback() {
    21 usages ± ph20531
    @Override
    public void onSuccess(Object response) {}
    @Override
    public void onFailure(String throwMessage) {}
});
```

```
new Thread() -> {
    try {
        latch.await();
        mainLatch.countDown();
    } catch (InterruptedException e) {}
}

if (post.getProfileUrl() != null) {
    post.setType(Post.TYPE_THUMBNAIL);
} else {
    if (DummyData.IS_PROFILE_IMAGE_AVAILABLE) {
        post.setType(Post.TYPE_IMAGE);
    } else {
        post.setType(Post.TYPE_VIDEO);
        post.setIconDrawable(R.drawable.ic_image);
    }
}

public static void isBookmark(Context context, String accessToken, int postId, API.ApiService apiService = API.getInstance().create(API.ApiService.class)) {
    Call<API.Bookmark.IsBookmarkResponse> call = apiService.isBookmark(accessToken, postId);
    call.enqueue(new Callback<API.Bookmark.IsBookmarkResponse>() {
        21 usages ± ph20531
        @Override
        public void onResponse(@NonNull Call<API.Bookmark.IsBookmarkResponse> call, Response<API.Bookmark.IsBookmarkResponse> response) {
            if (response.body() != null) {
                API.Bookmark.IsBookmarkResponse isBookmarkResponse = response.body();
                if (response.isSuccessful() && "success".equals(isBookmarkResponse.getMessage())) {
                    String message = isBookmarkResponse.getMessage();
                    Toast.makeText(context, "success: " + message, Toast.LENGTH_SHORT).show();
                    responseCallback.onSuccess(isBookmarkResponse);
                } else {
                    Toast.makeText(context, "error: " + message, Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
    latch.countDown();
}
```

7. Issues

(2) Front-End Issue 2

Exoplayer 라이브러리 사용시 동영상 처리 중
Recyclerview의 각 item에 Exoplayer를 적용하면
어플리케이션의 프로세스가 종료될 정도

심각한 메모리 누수 문제 발생



① 각 동영상 URL(Media Item) 데이터 파일 캐시처리

(파일 캐시 허용크기는 1gb로 설정)

② 리사이클러뷰의 어댑터에서

-Item detach시 Exoplayer를 Release

-Item attach시 Exoplayer를 다시 인스턴스화

※ 안드로이드 스튜디오의 Memory Profiler를 사용하여

메모리 상태 관측하며 디버깅

※ glide 라이브러리는 자동 캐시처리가 되므로 별도의 작업X

```
4 usages 1 ph20531
public void release() {
    if (exoPlayer != null) {
        styledPlayerViewMobil.setPlayer(null);
        exoPlayer.setPlayWhenReady(false);
        exoPlayer.stop();
        exoPlayer.clearMediaItems();
        exoPlayer.release();
        exoPlayer = null;
    }

    if (handlerProgress != null && runnableProgress != null) {
        handlerProgress.removeCallbacks(runnableProgress);
        handlerProgress = null;
        runnableProgress = null;
    }
}

styledPlayerViewMobil.getVideoSurfaceView().setOnClickListener(v -> {
    if (exoPlayer == null) return;

    if (isPlaying) {
        imageViewState.setVisibility(View.VISIBLE);
        imageViewState.setImageResource(R.drawable.round_pause_circle);
        imageViewState.post(() -> {
            pulse.playOn(imageViewState).stop();
            fadeOut.playOn(imageViewState).stop();
            pulse.playOn(imageViewState);
            fadeOut.playOn(imageViewState);
        });
        handlerState.removeCallbacks(runnableState);
        handlerState.postDelayed(runnableState, STATE_VISIBLE_TIME);
        exoPlayer.setPlayWhenReady(false);
        exoPlayer.stop();
        isPlaying = false;
    } else {
        imageViewState.setVisibility(View.VISIBLE);
        imageViewState.setImageResource(R.drawable.round_play_circle_f);
        imageViewState.post(() -> {
            pulse.playOn(imageViewState).stop();
            fadeOut.playOn(imageViewState).stop();
            pulse.playOn(imageViewState);
            fadeOut.playOn(imageViewState);
        });
        handlerState.removeCallbacks(runnableState);
        handlerState.postDelayed(runnableState, STATE_VISIBLE_TIME);
        exoPlayer.prepare();
        exoPlayer.play();
        isPlaying = true;
    }
});

1 ph20531
@Override
public void onViewAttachedToWindow(@NonNull ViewHolder holder) {
    super.onViewAttachedToWindow(holder);
    int position = holder.getAbsoluteAdapterPosition();
    holder.styledPlayerViewMobil.post(() -> {
        if (holder.exoPlayer == null) {
            Mobil item = data.get(position);
            holder.bind(context, item);
        }
    });
}

1 ph20531+1
@Override
public void onBind(@NonNull ViewHolder holder) {
    Mobil item = data.get(position);
    holder.bind(context, item);
}

1 ph20531
@Override
public void onViewDetachedFromWindow(@NonNull ViewHolder holder) {
    super.onViewDetachedFromWindow(holder);
    holder.release();
}

1 ph20531
@Override
public void onViewRecycled(@NonNull ViewHolder holder) {
    super.onViewRecycled(holder);
    holder.release();
}
```

7. Issues

(3) Back-End Issue 1

① 배포시 AWS Lambda RunTime 환경 고려하여

가상환경 생성시 Python==3.10으로 설치

② MongoDB Atlas 이용하기 위해 파이썬 3.10 버전을 지원하는

pymongo==3.6 설치

③ 다른 라이브러리들과 지속적인 Dependency Conflict 문제 발생



① AWS Lambda RunTime 지원버전 재확인 후,

Python==3.12로 가상환경 재생성

② pymongo==4.8(최신버전)로 재설치

③ 가상환경 재생성 및 Activate 후

GitHub Actions, Docker, Serverless 이용한

자동배포 상태 테스트하여 CD 재구축

```
PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

(lambda_310) C:\Users\Fletcher\Documents\GitHub\serverless-move-server>pip install "pymongo[srv]"==3.6
Collecting pymongo==3.6 (from pymongo[srv]==3.6)
  Downloading pymongo-3.6.0.tar.gz (581 kB)
    581.3/581.3 kB 2.2 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
Collecting dnspython<2.0.0,>=1.13.0 (from pymongo[srv]==3.6)
  Downloading dnspython-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
  Downloading dnspython-1.16.0-py2.py3-none-any.whl (188 kB)
    188.4/188.4 kB 2.8 MB/s eta 0:00:00
Building wheels for collected packages: pymongo
  Building wheel for pymongo (setup.py) ... done
  Created wheel for pymongo: filename=pymongo-3.6.0-cp310-cp310-win_amd64.whl size=286203 sha256=7542666bfe620af828a5e8937a
  Stored in directory: c:\users\fletcher\appdata\local\pip\cache\wheels\88\57\56\3225f60abe36f24a9f9ea413891771b7e9ab7fac82e9
Successfully built pymongo
Installing collected packages: pymongo, dnspython
  Attempting uninstall: dnspython
    Found existing installation: dnspython 2.6.1
  Successfully uninstalled dnspython-2.6.1
Successfully installed pymongo-3.6.0 dnspython-1.16.0

inchedpad 4 6 0 You, 27 minutes ago

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

  Downloading dnspython-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
  Downloading dnspython-1.16.0-py2.py3-none-any.whl (188 kB)
    188.4/188.4 kB 2.8 MB/s eta 0:00:00
Building wheels for collected packages: pymongo
  Building wheel for pymongo (setup.py) ... done
  Created wheel for pymongo: filename=pymongo-3.6.0-cp310-cp310-win_amd64.whl size=286203 sha256=7542666bfe620af828a5e8937a
  Stored in directory: c:\users\fletcher\appdata\local\pip\cache\wheels\88\57\56\3225f60abe36f24a9f9ea413891771b7e9ab7fac82e9
Successfully built pymongo
Installing collected packages: pymongo, dnspython
  Attempting uninstall: dnspython
    Found existing installation: dnspython 2.6.1
  Uninstalling dnspython-2.6.1:
    Successfully uninstalled dnspython-2.6.1
  Successfully installed pymongo-3.6.0 dnspython-1.16.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is
  email-validator 2.1.1 requires dnspython>=2.0.0, but you have dnspython 1.16.0 which is incompatible.
  Successfully installed dnspython-1.16.0 pymongo-3.6.0

(lambda_310) C:\Users\Fletcher\Documents\GitHub\serverless-move-server>
aunchedpad 4 6 0 You, 27 minutes ago Ln 20, Col 16 Spaces
```

Members.



강동현 Team Leader +

Move 프로젝트 총괄 및 백엔드 전담
 -DB 아키텍처 설계 & 스키마 구성
 -서버 아키텍처 설계 & 구축
 -API 서버 개발
 -CI/CD 파이프라인 구축
 -백엔드 이슈 디버깅

#AWS #MySQL #MongoDB
 #Flask #Python #Java
 #AndroidStudio #DataAnalysis
 #Streamlit #Docker #GitHub
 #GitHub Actions #Serverless
 #디버깅사랑 #성실



강도형 Collaborator +

Move 프로젝트 프론트엔드 담당
 -UI/UX 프로토타이핑
 -커스텀 UI 스타일링
 -프론트엔드 이슈 디버깅

#AndroidStudio #Java
 #UI Design #Figma
 #Domain Knowledge
 #분위기메이커 #유머지존
 #행보관스타일



장범근 Collaborator +

Move 프로젝트 프론트엔드 담당
 -UI/UX 설계
 -MVC 패턴, Singleton 활용
 -프론트 전체 기능 구현 및 바인딩

#AndroidStudio #Java #UI/UX
 #Front-End #JavaScript
 #Realtime Database #FCM
 #C# #.NET #Design #WPF
 #중고신입 #끈기왕



김진우 Collaborator +

Move 프로젝트 인공지능 담당
 -LightFM 라이브러리를 이용하여
 협업 필터링 기반 추천 모델 개발

#AI #Machine-Learning
 #Deep-Learning #Python
 #Numpy #Pandas #Scikit-Learn
 #Matplotlib #Seaborn
 #인공지능 러버 #열리어답탈

Move

Team Movement

Appreciate
it.

Mobile 010.4222.1358

E.Mail fletcher_seth2335@naver.com

GitHub