종합설계 프로젝트 수행 보고서

프로젝트명	웹 크롤링을 이용한 OTT 서비스 연동 어플리케이션
팀번호	S5-2
문서제목	수행계획서( ) 2차발표 중간보고서( ) 3차발표 중간보고서( ) 4차발표 중간보고서(O) 최종결과보고서( )

2022.06.03

팀원 : 박찬호 (팀장)
김재현 (팀원)
김진호 (팀원)

지도교수 : 전광일 교수

문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	비고
2022.02.04	박찬호 (팀장)	1.0	수행계획서	최초작성
2022.03.02	박찬호 (팀장)	2.0	2차 중간발표	2차 보고서 작성 서론 및 본론 ~2.3절 수정
2022.03.06	김재현 (팀원)	2.1	2차 중간발표	2.4.4 Backend 모듈 작성
2022.03.06	김진호 (팀원)	2.2	2차 중간발표	2.4.6 데이터베이스 설계 작성
2022.03.06	박찬호 (팀장)	2.3	2차 중간발표	~2.4.6절 설계내용 수정 및 추가
2022.04.16	박찬호 (팀장)	3.0	3차 중간발표	2.5절 프로토타입 구현 작성
2022.04.19	박찬호 (팀장)	3.1.1	3차 중간발표	2.4.3 설계내용 추가/수정
2022.04.20	김재현 (팀원)	3.1.2	3차 중간발표	2.4.4 설계내용 추가/수정
2022.04.21	김진호 (팀원)	3.1.3	3차 중간발표	2.4.3 설계내용 추가
2022.04.22	박찬호 (팀장)	3.2	3차 중간발표	~2.5절 내용 수정 및 검토
2022.06.03	박찬호 (팀장)	4.0	4차 보고서제출	~ 2.7절 내용 추가 및 수정
2022.06.03	김진호 (팀원)	4.1	4차 보고서제출	2.7.4절 내용 추가

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (3월)	중간발표2 (5월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~3) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~4) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~5) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~7) 참고자료	I II III

이 문서는 한국산업기술대학교 컴퓨터공학부의
“종합설계” 교과목에서 프로젝트
“웹 크롤링을 이용한 OTT 서비스 연동 어플리케이션” 을
수행하는 (S5-2, 박찬호,김재현,김진호)들이
작성한 것으로 사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

Ⅰ. 서론
1. 작품선정 배경 및 필요성 ……
2. 기존 연구/기술동향 분석 ……
3. 개발 목표 ……
4. 팀 역할 분담 ……
5. 개발 일정 ……
6. 개발 환경 ……

Ⅱ. 본론
1. 개발 내용 ……
2. 문제 및 해결방안 ……
3. 시험시나리오 ……
4. 상세 설계 ……
5. Prototype 구현 ……
6. 시험/ 테스트 결과 ……
7. Coding & DEMO ……

Ⅲ. 결론
1. 연구 결과 ……
2. 작품제작 소요재료 목록 ……

참고자료 ……

I. 서론

I 작품선정 배경 및 필요성

■ 배경과 시장성

[2014-2020 OTT 매출액]

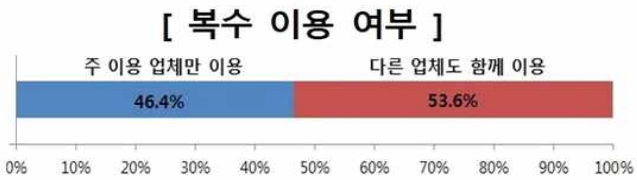
- 최근 OTT는 코로나19 팬데믹 상황에서 비대면 서비스 중 하나로 성장했다. OTT 시장이 급속도로 커지고 있는 가운데 기존 OTT 서비스의 개편과 신규 진출 OTT 사업자들의 진출이 계속해서 늘어날 전망이다. 한국수출입은행 보고서에 따르면 국내 OTT 산업은 2012년 이후 연평균 28% 성장을 거듭해 2020년 기준 7801억원 규모로 성장했다. 추가로 방송통신위원회는 2021년 국내 OTT 시장 규모를 1조원으로 예상했다.

[국내 OTT 사용자 현황]

- OTT 서비스의 등장 이후 '넷플릭스'의 독점을 선두로 OTT 시장이 활성화되었다. 하지만 현재 국내 OTT 기업도 기존 방송콘텐츠 위주의 OTT 서비스보다 자체 콘텐츠 경쟁력을 살릴 수 있는 전략을 통해 치열한 경쟁이 벌어져 OTT 서비스 시장의 활발함과 함께 꾸준한 성장세를 보인다.

■ 필요성

- OTT 시장의 활성화가 이뤄지면서 다양한 콘텐츠에 대한 소비자들의 갈증 해결도 기업들의 과제가 되었다. 이에 따라 다른 매체들에 머물던 기업들이 서로 만나 새로운 영화나 드라마를 제작하기도 하고, 통신사와 함께 혜택을 제공하기도 하며 OTT 선택이 폭을 늘렸다. 결과적으로는 각 플랫폼에 콘텐츠 경쟁력과 정체성이 분명해졌지만, 한편으로는 소비자들은 시청하고 싶은 콘텐츠의 서비스 분리에 실망감을 가질 수 있다.



[2021.06 한국소비자원 OTT 서비스 사용자 실태 조사]

- 한국소비자원에 따르면 2021년 6월 기준 OTT 플랫폼을 중복으로 구독하는 소비자는 전체의 53.6%이었다. 이 중 복수 이용자의 평균 구독 수는 2.69개였으며 다른 플랫폼으로 교체한다면 '시청하고 싶은 콘텐츠가 있다'이거나 '콘텐츠 종류가 다양해서'라고 답했다.

- 이처럼 점차 다양해지는 OTT 콘텐츠는 분리되어 있어 결국 소비자들은 복수의 플랫폼을 구독하게 된다. 이때 소비자들은 어떤 영화나 드라마가 어떤 플랫폼에서 제공하는지 수시로 궁금증을 갖게 되고, 계속해서 자신이 구독하는 OTT에서 이전에 찜했거나 현재 시청하고 있는 콘텐츠들을 일일이 나누어 관리하기에 불편함을 가질 수밖에 없다.

II 기존 연구/기술 동향 분석

■ 기존 서비스 기업

기업	소개	
키노라이츠 (KinoLights)	· 글로벌과 국내 주요 OTT 플랫폼의 메타 검색 엔진을 제공하는 영화 맞춤 추천 서비스	
	장점	유저 친화적, 유저 평가 기반 추천 시스템
	단점	인증 회원 제도를 통해 인증받은 유저들만 사용 가능
Watcha Pedia	· 이용자들의 평가 데이터를 통해 제공하는 영화, 드라마, 도서 통합 검색 및 추천 서비스	
	장점	Watcha(왓챠) 생태계 맞춤 서비스, 리뷰, 취향 분석 시스템 제공
	단점	일부 주요 기능이 왓챠 콘텐츠에 한정됨
JustWatch	· 온라인 스트리밍 전용 영화, TV 시리즈 및 기타 유형의 프로그램을 제공하는 온라인 가이드	
	장점	콘텐츠마다 제공 OTT 플랫폼을 제공, 구매 가격도 플랫폼별로 비교
	단점	OTT 플랫폼 자체 구독이 아닌 개별 영화/TV 시리즈 단위 구매 비교

III 개발 목표

■ 개발 목표

1. 주요 OTT 서비스에서 제공하는 콘텐츠 데이터를 수집하고 앱을 통해 사용자가 검색할 수 있게 하고, 해당 OTT로 연결한다.

2. 사용자의 OTT 서비스 이용 데이터를 통해 통합된 하나의 앱으로 관리하고 OTT의 찜한 목록이나 시청 중인 콘텐츠를 보여줄 수 있다.

3. 수집한 이용 데이터에 맞는 콘텐츠 추천 모델을 적용하여 정밀도 높은 추천 시스템을 구축한다.

4. 전체 시스템은 효율성과 가용성에 치중하여 개발하고 사용자 이용에 편리한 UI를 구현한다.

■ 차별성

〈기존 기업과 비교 OX 표〉

	키노라이츠	왓챠피디아	JustWatch	프로젝트
통합 검색	o	o	o	o
간편로그인	o	o	o	o
보고있는 항목 목록	o	o	o	o
랭킹	o	o	o	o
추천 알고리즘	o	o	o	o
요즘 인기 있는 영화/드라마 모음	o	o	o	o
리뷰/평가	o	o	o	o
이용 가이드	o	o	o	o
내가 좋아요 남긴 리뷰 나열	o	o	x	o
구독중인 서비스 선택	o	x	o	o
찜한 목록	o	o	o	o
검색 필터링	Δ	x	o	o
내가 쓴 댓글 관리	o	o	x	o
찜한 목록 연동	x	Δ	x	o
시청중인 콘텐츠 연동	x	x	x	o
게시판	x	x	x	o
UI 연령대별 추천	x	x	x	o
구독 서비스들 가격 제공	x	x	x	o
UI 갤럭시 앱 전환화면	x	x	x	o

* Δ는 일부 기능을 지원하지 않음.

● 주요 차별점

- 시청 중인 콘텐츠나 찜한 목록을 조회하고, 연동하여 각 OTT 플랫폼에 동기화
- UI 연령대별 추천 시스템을 적용하여 사용자 나이에 맞는 추천 콘텐츠 제공
- 게시판을 통해 유저들이 구독 서비스를 추천하고 각 플랫폼의 구독 가격 비교
- 기존 행렬형식의 UI에 더하여 추가적인 UI 디자인 제공

IV 팀 역할 분담

	박찬호	김재현	김진호
자료수집	· Android MVC · Android UI 디자인 · 추천 알고리즘 종류	· Android 앱 개발 · 3계층 RESTful API · Web Crawling · AWS Amplify	· Android 앱 개발 · AWS RDS · MySQL 통합검색
설계	· Android UI/UX · Android MVC · 추천 알고리즘	· REST Framework · Selenium 웹 크롤링 & 자동화 · AWS Cognito	· Kotlin을 통한 앱 개발 · MariaDB 구축 및 관리 · Fulltext Search 개발
구현	· Android MVC · 앱 디자인 및 애니메이션 · PySpark 추천 알고리즘	· AWS Server 구축 · Selenium 웹 크롤링 & 자동화 · AWS Cognito · Kakao 로그인 구현 · Surprise SVD 추천 알고리즘 구현 및 적용	· Android 앱 개발 · MariaDB 구축 및 관리 · Fulltext Search 구현 · Naver 로그인 구현 · 평가 및 리뷰 구현
테스트	· Application 작동 테스트 · Python Web Crawling & 자동화 작동 테스트 · 통합 테스트 / 유지보수		

V 개발 일정

스프린트 주요 내용	12월	1월	2월	3월	4월	5월	6월
개발 환경 구축 (DB-Server-App 연동)							
웹 크롤링 & 자동화 찜 & 시청중인 목록 연동							
로그인 기능 사용자별 OTT 서비스 연동							
통합 검색 & 필터링 간편 로그인 기능							
추천 알고리즘 구현 및 적용							
평가 & 리뷰 기능							
랭킹 & 인기있는 콘텐츠							
신작 업데이트 콘텐츠 연동							
게시판							
앱 UI 디자인 이용가이드							

VI 개발 환경

구분	항목	설명
사용자 어플리케이션	· Kotlin / XML	앱 개발 언어
	· Android Studio	앱 개발 Tool
서버	· Python3	기능 구현 언어
	· PyCharm	기능 구현
	· Selenium	웹 크롤링 라이브러리
	· Surprise SVD	콘텐츠 추천 알고리즘 패키지
	· Amazon Web Services (AWS) · Ubuntu	클라우드 서버 운영 목적
	· Django	앱 서버 Rest API
	· NGINX / Gunicorn	웹 서버 운영 / 프록시 목적 3계층 인터페이스
	· Docker	웹 / 앱 컨테이너 및 배포
데이터베이스	· MySQL Workbench CE	데이터베이스 관리 Tool
	· SQL	데이터베이스 관리 언어
	· Amazon RDS · MariaDB	데이터베이스 관리 시스템
인증	· AWS Cognito	사용자 인증 서비스(로그인)
협업 도구	· Agile (Scrum Framework)	점증적 개발 목적 소프트웨어 개발 방법론
	· Git / Github	VCS
	· Trello	프로젝트 일정 관리
	· Discord	팀원 간 커뮤니케이션 및 자료 공유

Ⅱ. 본론

Ⅰ 개발 내용

1.1 주요 개발 내용 * 현재까지 구현한 내용만 작성.

구분	내용
클라이언트(앱)	· Retrofit2으로 http프로토콜을 통해 서버에 데이터 요청(get/post) (URL-Encoded Form 형식으로 수신) · 요청된 데이터를 알맞은 UI에 출력 · 동적 데이터의 경우 Adapter를 통해 뷰에 데이터 바인딩 · 컨트롤러를 다수의 Fragment로 나누어 동작 경량화
서버	· AWS EC2 Ubuntu Server 20.04를 이용하여 서버 구축 · Docker를 이용하여 NGINX와 Gunicorn-Django를 연결한 3계층 구조 구현 · Django를 이용한 Python 웹 애플리케이션 개발 · Django REST Framework를 이용하여 REST API Server로 동작하도록 구현 · Middleware인 Gunicorn을 통해 NGINX로 웹 서버 배포 · 클라이언트의 REST API 요청(get/post)에 따른 값을 JSON 형식으로 전달 · 클라이언트는 DB에 직접 연결하지 않고 서버를 통해 DB 데이터를 요청하고 결과를 받아서 사용
데이터베이스	· MariaDB를 채택, MySQL WorkBench 툴을 이용하여 관리 · 서버 Django에 모듈 형태로 Amazon RDS를 통해 호스트 커넥션 · SQL 쿼리문을 사용하여 외부 API와 크롤링을 통해 수집된 데이터를 형식에 맞춰 추가/수정/삭제
웹 크롤링 & 웹 자동화	· Python3에서 지원하는 Selenium 라이브러리를 사용 · 선정한 OTT의 사이트 HTML 파일을 로드하여 개발에 필요한 데이터를 리턴 · Chrome Driver를 통해 태그나 CSS 속성으로 구분하여 자동화 구현 · 웹 데이터 로드 시간을 고려하여 로드 함수나 딜레이 함수 적절히 사용
통합 검색	· MariaDB에 내장된 텍스트 인덱싱 방식 Full-Text Indexing을 적용 · 정확도에 따른 정렬과 구분 검색을 위해 불린 모드로 검색 · WHERE문을 통해 서버에서 검색할 텍스트를 전달받으면 DB에서 컬럼을 지정하여 SELECT 후 출력
인증(로그인)	· AWS Amplify의 Cognito를 사용해 인증 체계 구축 · AWS Amplify를 안드로이드 앱과 연동시켜 회원가입, 로그인이 가능하도록 함 · e-mail 인증을 통해 e-mail 하나당 하나의 계정만 가입할 수 있도록 함 · 앱에서 한 번 로그인 하면 다음번 실행 때도 자동으로 로그인되도록 함 · e-mail, 비밀번호와 추가 사용자 정보를 저장하여 필요할 때 사용할 수 있도록 함

추천 알고리즘	· Surprise 라이브러리를 이용한 추천시스템 구현 · 행렬 분해(Matrix Factorization) 기법 중 특이값 분해(SVD : Singular Value Decomposition)를 이용한 협업 필터링(Collaborative Filtering) · MovieLens 영화 평점 데이터를 이용한 학습 · 추천을 요청하는 사용자가 보지 않은 영화 목록 중 예상 별점이 높은 200개의 작품 추천 제공

1.2 세부 기능

기능	설명	구현 여부
통합 검색	· OTT에서 제공하는 영화 및 드라마를 통합된 환경에서 검색	O
간편로그인	· 구글, 네이버, 카카오 계정을 통해 간편하게 가입하거나 로그인	O
시청중인 목록	· 등록된 OTT 계정에서 시청 중으로 표기되는 목록 통합 출력	O
랭킹	· 콘텐츠의 이용률 기준으로 사용자 정보 카테고리별 순위 제공	X
추천 알고리즘	· 하이브리드 필터링(Hybrid Filtering)에 따라 콘텐츠 추천	O
요즘 인기 있는 영화/드라마 모음	· OTT 플랫폼별로 인기 있는 영화 및 드라마를 제공	X
리뷰/평가	· 각 콘텐츠에 대한 사용자들의 리뷰와 평가를 저장 및 공유	O
이용 가이드	· 해당 프로젝트 앱의 이용 가이드를 제작 및 제공	X
내가 좋아요 남긴 리뷰	· 사용자가 좋아요 표시를 남긴 콘텐츠 출력	O
구독중인 OTT 관리	· 사용자의 OTT 플랫폼 계정을 추가/수정/삭제	O
찜한 목록	· 등록된 OTT 계정에서 찜하고 있는 콘텐츠 목록 통합 출력	O
검색 필터링	· 통합 검색에서 요구되는 영화 상세 정보에 따라 거름	O
내가 쓴 댓글 관리	· 게시판이나 리뷰에서 사용자가 남긴 글 관리	O
찜하기 연동	· OTT 제공 콘텐츠를 찜 목록에 추가, 삭제하고 실제 정보와 동기화	O
시청중인 콘텐츠 동기화	· 실시간으로 시청한 콘텐츠 목록을 지속해서 갱신	O
게시판	· OTT 플랫폼 또는 사용자 유형 카테고리별 커뮤니티	X
이용자 별 추천	· 사용자의 연동정보를 반영하여 추천 알고리즘에 따른 콘텐츠 추천	O
구독 서비스들 가격 제공	· 각 OTT 플랫폼의 가격 비교/제공/갱신	X

* Δ는 구현 중인 기능.

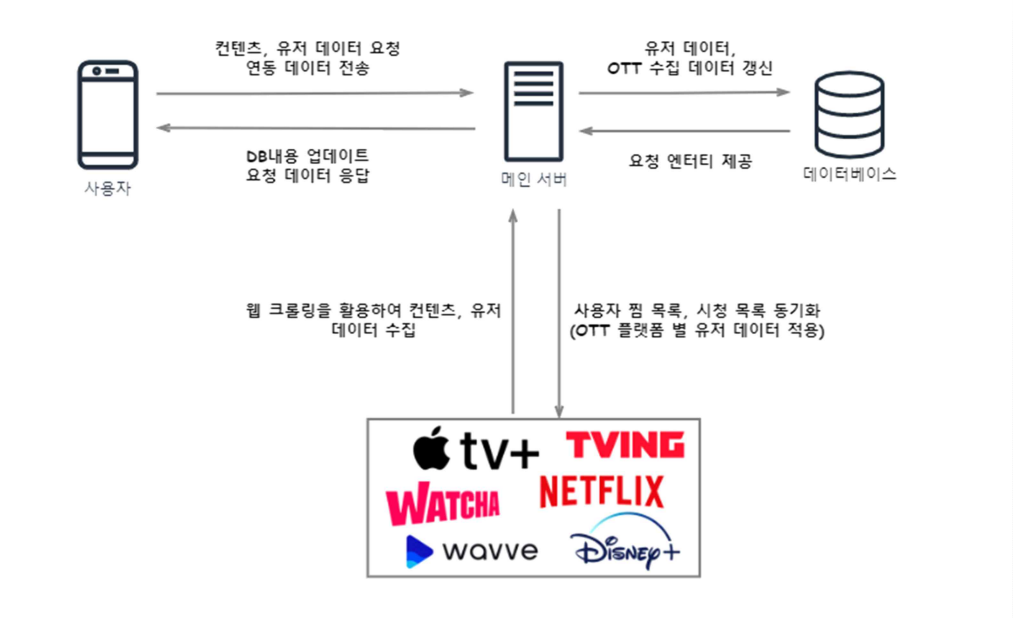
* 구현 비율 : 72.2% (13/18)

II 문제 및 해결방안

문제	해결방안
OTT 플랫폼에서 크롤링을 금지할 때는 어떻게 할 것인가?	· 기본적인 영화나 TV 시리즈, 드라마 정보와 해당 콘텐츠의 제공 OTT 플랫폼은 TMDB 오픈 API를 통해 수집할 수 있다. 크롤링을 금지하는 정책이 있는 OTT의 경우, 해당 API의 제공 플랫폼 정보와 대조하여 서비스를 제공한다.
유저 별 서버 접속에 따른 트래픽과 연결 장애는 어떻게 해결하는가?	· Django – Gunicorn – Nginx 의 3계층을 통해 프록시 기능을 갖춘 웹 서버 기능 역할을 하는 인터페이스를 두어 설계한다. · 각 소프트웨어 계층의 가용성과 안전성, 속도를 고려하여 docker 컨테이너를 통해 운영하여 메모리 소모를 줄인다.
OTT 별 지속적인 접속에 따른 이용 제한은 생기지 않는가?	· 캐싱 : 저장된 유저의 계정정보를 이용하여 클라이언트와 별개로 서버가 일정 간격을 두며 주기적으로 동작하여 DB 정보를 업데이트한다. 이 때문에 OTT의 계속된 접속 방지와 함께 유저의 앱 사용감도 유연해질 수 있다. (유저가 서버에 직접 DB 정보를 수동으로 업데이트할 것을 요청할 수도 있게 기능을 별도 구현한다.)
데이터 크롤링에서 발생하는 딜레이에 있어서는 어떻게 할 것인가?	
콘텐츠 추천의 기준은 어떻게 되는가?	· 콘텐츠 추천 알고리즘의 기준은 사용자들이 남긴 리뷰의 평점에 따라 추천된다. 남긴 리뷰의 양이 많아질수록 추천결과에 신뢰도가 생기게 된다. 최초 사용자의 경우, 랜덤한 콘텐츠를 사용자가 임의로 선택하거나 평가하여 리뷰 데이터를 대신한다.

Ⅲ 시험시나리오

3.1 전체 시나리오

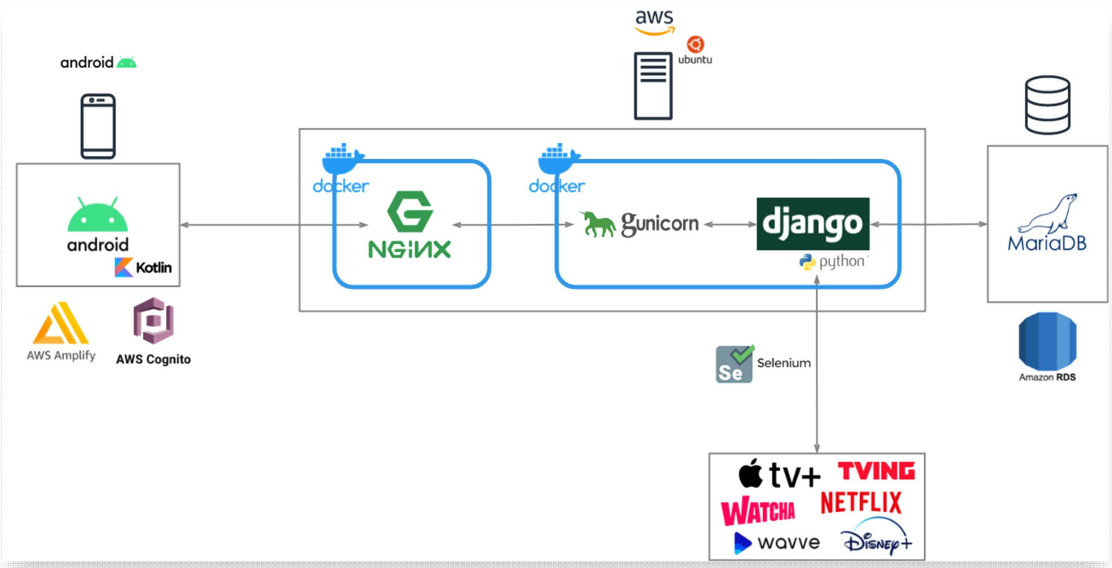


3.2 상세 시나리오 * 현재까지 구현한 내용만 작성.

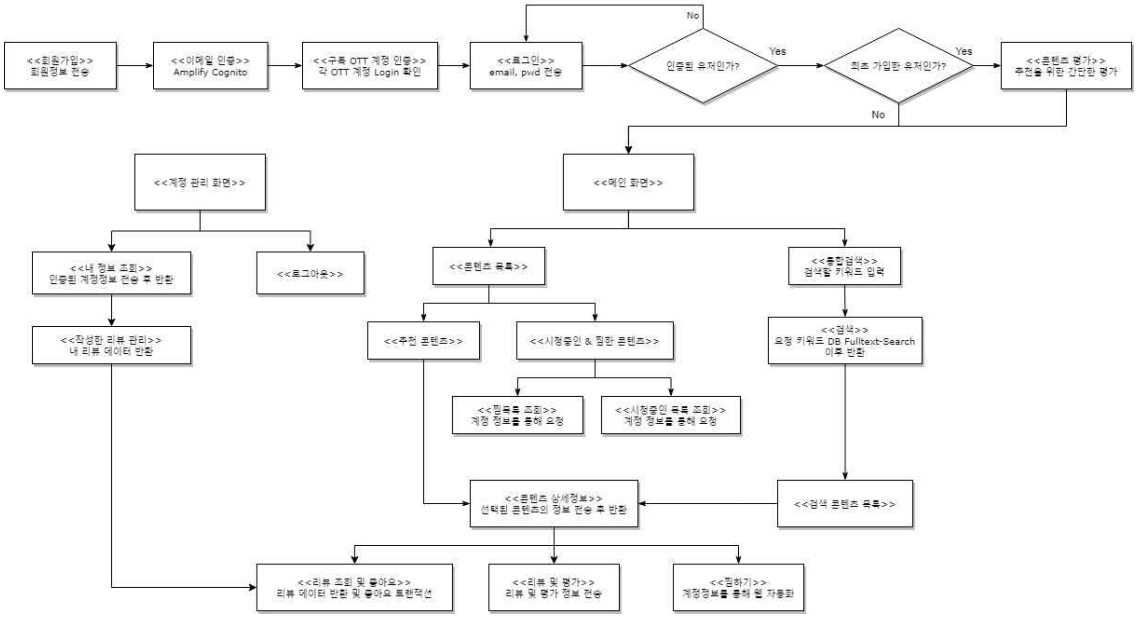
<table>
<tr><th rowspan="2">시나리오</th><th colspan="3">내용</th></tr>
<tr><th>클라이언트</th><th>서버</th><th>데이터베이스</th></tr>
<tr><td>로그인</td><td>· 계정 이메일과 비밀번호 전송
· AWS Cognito 인증 요청</td><td>· DB의 회원 정보와 일치하는지 확인 후 결과 반환
· Cognito를 통해 로그인 유지</td><td rowspan="12">서버 요청 SQL 반영/결과 반환</td></tr>
<tr><td>회원 가입</td><td>· 회원 가입 정보 전송
· AWS Cognito 인증 요청</td><td>· DB에 일맞은 컬럼에 맞춰 엔티티 추가
· Cognito를 통해 중복 가입 방지</td></tr>
<tr><td>OTT 계정 인증</td><td>· 구독한 OTT 정보와 계정 정보로 request</td><td>· response 데이터로 OTT 웹에 로그인 후 저장
· 유저에 성공 여부 반환</td></tr>
<tr><td>콘텐츠 정보 조회</td><td>· 전체 콘텐츠 정보 request
· 객체 List를 통해 UI 출력</td><td>· DB 저장된 콘텐츠 정보 모두 반환</td></tr>
<tr><td>찜한 목록 조회</td><td rowspan="2">· 인증된 유저가 목록 request</td><td rowspan="2">· 인증된 유저임을 확인
· 계정정보를 통해 DB정보 반환</td></tr>
<tr><td>시청중인 목록 조회</td></tr>
<tr><td>찜하기</td><td>· 찜하고 싶은 콘텐츠 정보와 제공하는 OTT로 request</td><td>· 인증된 유저임을 확인
· 계정정보와 콘텐츠 이름을 통해 웹 자동화 모듈 실행 후 성공 여부 반환</td></tr>
<tr><td>리뷰 작성</td><td>· 콘텐츠 번호, 계정 정보, 리뷰 내용, 평점으로 request</td><td>· 콘텐츠와 호출된 계정 확인
· 트랜잭션 상호배제를 고려하여 리뷰 저장 후 데이터화하여 반환</td></tr>
<tr><td>리뷰 조회 및 수정, 삭제</td><td>· 콘텐츠 번호와 계정 정보를 통해 request</td><td>· DB에 저장된 리뷰 정보를 요청된 기능에 맞게 선택적으로 반환
· 계정 정보는 자신의 리뷰 조회 때 사용됨</td></tr>
<tr><td>좋아요</td><td>· 콘텐츠 번호와 좋아요/싫어요 유무를 통해 request</td><td>· DB 리뷰 로그의 좋아요 컬럼에 상호배제 형태로 증가하거나 감소</td></tr>
<tr><td>콘텐츠 추천</td><td>· 계정 정보로 request</td><td>· 서버의 백그라운드에서 생성된 추천 모델을 통해 리스트 반환
· 중복 제거 작업 및 학습 모두 서버에서 정제</td></tr>
<tr><td>통합 검색</td><td>· 검색할 키워드로 request</td><td>· response 데이터로 DB에 SQL 문 형태로 검색 수행
· 결과 리스트를 유저에 반환</td><td>요청된 SQL 문의 매치 칼럼 지정 후 검색 결과 반환</td></tr>
</table>

IV 상세 설계

4.1 시스템 구성도

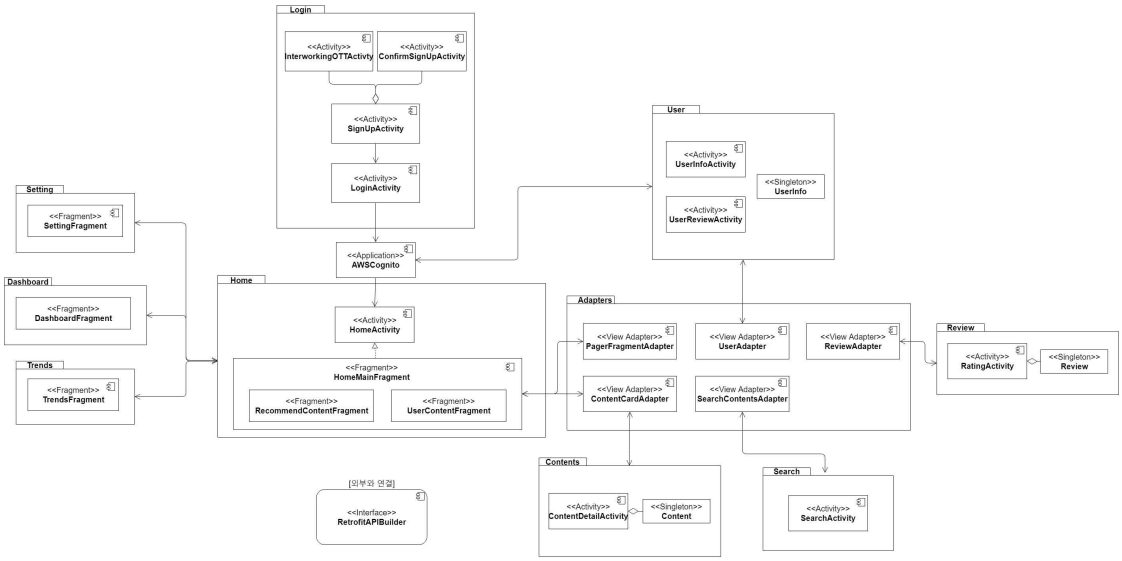


4.2 Flow Chart * 현재까지 구현한 내용만 작성.



4.3 S/W 모듈 구성도 – 클라이언트 * 현재까지 구현한 내용만 작성.

– FrontEnd 모듈 구성도



– 모듈 설명

<table>
<tr><th colspan="3">■ [데이터 전송 인터페이스]</th></tr>
<tr><td>기능</td><td colspan="2">■ [서버와 데이터 송수신을 목적으로 하는 API 정의 인터페이스]
– 사용자가 요청한 데이터를 전송하는 역할을 하고 DB의 정보를 읽어오는 기능 모두에서 호출된다.</td></tr>
<tr><td rowspan="4">모듈</td><td>이름</td><td>설명</td></tr>
<tr><td>okhttp3</td><td>http 기반의 통신을 할 수 있게 해주는 클라이언트 모듈</td></tr>
<tr><td>retrofit2</td><td>서버와 REST API 통신을 위해 구현되는 okhttp3 상위 구현체 모듈</td></tr>
<tr><td>TimeUnit</td><td>Http 클라이언트의 연결, 읽기, 쓰기 시간 설정 모듈</td></tr>
<tr><td rowspan="10">인터페이스
및
메소드</td><td>이름</td><td>설명</td></tr>
<tr><td>@GET()</td><td>지정한 URL에 GET 방식으로 통신하겠다는 것을 정의</td></tr>
<tr><td>@POST()</td><td>지정한 URL에 POST 방식으로 통신하겠다는 것을 정의</td></tr>
<tr><td>@FormUrlEncoded()</td><td>송수신 데이터의 Body 타입을 URL-Encoded Form으로 정의</td></tr>
<tr><td>.baseUrl()</td><td>retrofit 객체의 통신 희망 범위 최상위 URL 지정</td></tr>
<tr><td>.client()</td><td>build 한 okhttp 클라이언트 객체 지정</td></tr>
<tr><td>.addConverterFactory()</td><td>retrofit의 response 데이터 유형 변환에 사용(String, JSON 등….)</td></tr>
<tr><td>.create(API 클래스)</td><td>정의한 인터페이스를 통해 retrofit 통신 단위를 구축</td></tr>
<tr><td>.Call〈T〉</td><td>retrofit2에 미리 정의된 통신 요청 Builder</td></tr>
<tr><td>.enqueue()</td><td>동기적으로 서버에 request하고 callback 또는 error를 response 하는 메소드</td></tr>
<tr><td>추가설명</td><td colspan="2">각 요청 URL마다 별도의 API 정의가 필요(서버의 URL 개수만큼)</td></tr>
</table>

<table>
<tr><th colspan="3">■ [로그인 인증 모듈]</th></tr>
<tr><td>기능</td><td colspan="2">■ [로그인 및 회원 가입, 사용자 정보 인증 모듈]
– 이메일과 비밀번호를 통해 서비스에 로그인하는 역할을 하고 UI 요청에 따라 호출된다.
– 이메일과 비밀번호 및 기타 회원 정보를 통해 서비스에 회원 가입하는 역할을 하고 UI 요청에 따라 호출된다.
– 회원 가입이나 로그인 이후 기타 앱 기능에서 사용될 계정정보를 인증하는 역할을 하고 로그인 시 인증 절차가 수행된다.</td></tr>
<tr><td rowspan="6">모듈</td><td>이름</td><td>설명</td></tr>
<tr><td>Amplify</td><td>서버에 AWS Amplify.Auth로 Cognito 인증 기능을 사용할 모듈</td></tr>
<tr><td>AuthUserAttributeKey</td><td>Amplify.Auth에 알맞은 이메일(아이디)값으로 변환하는 모듈</td></tr>
<tr><td>AuthSignUpOptions</td><td>Amplify.Auth에 알맞은 회원 가입 조건을 설정하는 모듈</td></tr>
<tr><td>Pattern</td><td rowspan="2">회원 가입 시 정보입력의 유효성 검사 목적 정규식 모듈</td></tr>
<tr><td>Patterns</td></tr>
<tr><td rowspan="9">주요 메소드</td><td>이름</td><td>설명</td></tr>
<tr><td>.Auth.signIn()</td><td>AWS cognito를 통해 서버에 로그인 요청</td></tr>
<tr><td>.Auth.signUp()</td><td>AWS cognito를 통해 서버에 회원 가입 요청</td></tr>
<tr><td>.Auth.confirmSignUp()</td><td>AWS cognito를 통해 이메일 인증 및 중복 가입 방지</td></tr>
<tr><td>.Auth.fetchAuthSession()</td><td>AWS cognito를 통해 사용자가 서버에 연결 세션이 유효한지 조회</td></tr>
<tr><td>.isSignInComplete</td><td>Amplify에 정상적으로 로그인이 되었는지 확인하는 변수</td></tr>
<tr><td>.isSignUpComplete</td><td>Amplify에 정상적으로 회원 가입이 되었는지 확인하는 변수</td></tr>
<tr><td>.isSignedIn</td><td>Amplify에 정상적으로 로그인 상태로 연결되는지 확인하는 변수</td></tr>
<tr><td>.userAttribute()</td><td>사용자 정보에서 이메일을 회원의 단일 속성으로 설정</td></tr>
<tr><td>추가설명</td><td colspan="2">– 간편로그인의 추가 구현으로 인해 Amplify 기능 추가 구현 필요
– 카카오, 네이버 간편로그인은 Amplify에서 지원하지 않는 기능이므로 별도로 추가 구현 필요</td></tr>
</table>

<table>
<tr><th colspan="4">■ [동적 데이터 UI 어댑터]</th></tr>
<tr><td>기능</td><td colspan="3">■ [가변 크기의 데이터를 View에 표현을 돕는 지정 어댑터]
- RecyclerView, ViewPager2를 사용하는 UI에서 호출된다. 호출된 어댑터 클래스는 해당 View 컴포넌트에 어댑터로 재사용된다.</td></tr>
<tr><td rowspan="3">모듈</td><td colspan="2">이름</td><td>설명</td></tr>
<tr><td colspan="2">RecyclerView.Adapter<ViewHolder></td><td>RecyclerView에 사용되는 View 모듈</td></tr>
<tr><td colspan="2">FragmentStateAdapter</td><td>ViewPager2에 사용되는 View 모듈</td></tr>
<tr><td rowspan="6">구조</td><td>유형</td><td>이름</td><td>설명</td></tr>
<tr><td>매개변수</td><td>itemList</td><td>외부에서 전달할 객체 리스트</td></tr>
<tr><td>클래스(오버라이딩)</td><td>onCreateViewHolder()</td><td>View를 관리할 ViewHolder 생성 후 반환</td></tr>
<tr><td>클래스(오버라이딩)</td><td>onBindViewHolder()</td><td>itemList를 ViewHolder에 할당</td></tr>
<tr><td>클래스(오버라이딩)</td><td>getItemCount()</td><td>동적 리스트의 데이터 개수 반환</td></tr>
<tr><td>클래스</td><td>ViewHolder(view)</td><td>사용자 UI에 맞게 ViewHolder 직접 생성</td></tr>
<tr><td>추가설명</td><td colspan="3">사용자마다 데이터가 가변적이기 때문에 각 UI에 맞는 어댑터를 통해 동적으로 표현할 수 있다. 그러므로 가변적인 데이터의 개수가 늘어날 때마다 어댑터의 개수도 늘어날 것이다.</td></tr>
</table>

■ [콘텐츠 정보 UI 모듈]		
기능	■ [콘텐츠 정보를 다양한 UI에 알맞게 출력하는 모듈] – 콘텐츠 정보를 나열하여 카드 형태 UI에 출력하는 역할을 하고 해당화면 실행 시 호출된다. – 콘텐츠 정보를 객체 형태로 불러와 UI에 출력하는 역할을 하고 콘텐츠가 요청될 때마다 호출된다.	
모듈	이름	설명
	Fragment	Activity 내부에서 탭이나 내비게이션 바를 이용하여 가볍게 화면전환을 목적으로 하는 컨테이너 모듈
	ViewPager2	화면 좌우나 상하로 슬라이드하여 여러 요소를 출력하는 View 모듈
	PagerFragmentStateAdapter	ViewPager2에 출력될 Fragment를 관리하는 어댑터 모듈
	TabLayoutMediator	Tab의 현 위치를 탐지하기 위해 사용되는 모듈
	LinearLayoutManager	동적 콘텐츠 데이터 리스트를 View에 출력할 때 배치와 재사용을 결정하는 레이아웃 매니저 모듈
	Glide	URL의 이미지 정보를 편리하게 출력하도록 돕는 모듈
	Parcelable	Fragment나 Activity 간의 데이터를 콘텐츠 정보 객체 형태로 전달하기 위한 인터페이스 모듈
	Parcelize	콘텐츠 정보 객체를 Parcelable로 변환해주는 기능 모듈
주요 메소드	이름	설명
	.addFragment()	ViewPager2에 Fragment 추가(동적)
	.attach()	Fragment의 위치를 찾아주는 Tab을 설정
	.adapter	RecyclerView의 표현 어댑터를 지정
	.layoutManager	RecyclerView의 레이아웃 매니저를 지정
	.notifyDataSetChanged()	RecyclerView 재사용이 이뤄졌을 때 새로고침
	.load()	출력할 이미지의 URL을 통해 이미지 로드
	.into()	출력할 이미지의 View 지정

■ [찜&시청중인 목록 UI 모듈]		
기능	■ [사용자의 찜한 목록과 시청중인 목록, 찜하기 기능을 수행하는 모듈] – 사용자의 인증된 계정정보를 통해 각 OTT에 찜한 목록과 시청중인 목록을 요청하는 역할을 하고 UI 요청에 따라 호출된다. – 사용자 계정정보와 찜하거나 취소하고 싶은 콘텐츠의 제목을 전송하는 역할을 하고 사용자의 UI 요청에 따라 호출된다.	
모듈	이름	설명
	LinearLayoutManager	동적 콘텐츠 데이터 리스트를 View에 출력할 때 배치와 재사용을 결정하는 레이아웃 매니저 모듈
주요 메소드	이름	설명
	.adapter	View의 표현 어댑터를 지정
	.layoutManager	동적 View의 레이아웃 매니저를 지정
	.notifyDataSetChanged()	사용자의 새로운 요청에 따라 View 재사용이 이뤄졌을 때 새로고침
추가설명	– 클라이언트 단계에서는 원하는 데이터만 요청 후 서버의 결과값을 반환하기 때문에 모듈의 크기가 작음 – OTT의 구별을 서버에서 이뤄지므로 단순 콘텐츠 데이터 전송	

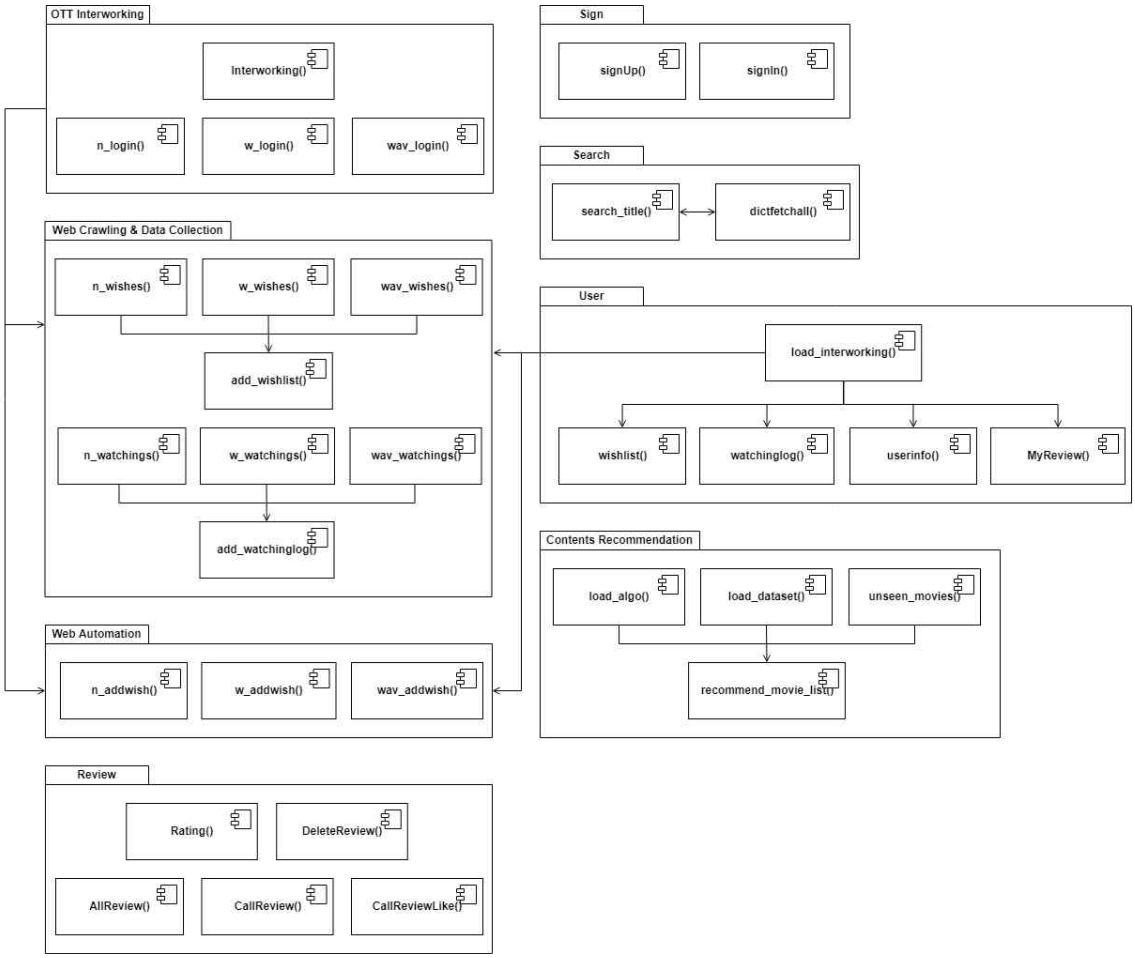
■ [유저 정보 UI 모듈]		
기능	■ [유저 계정 정보를 통해 다양한 내부 기능을 수행하는 데에 활용되는 모듈] – 내 계정 정보 조회하는 데에 데이터를 출력하는 역할을 하고 해당화면 실행 시 호출된다. – 계정 정보 수정, 타 모듈에서 계정 정보를 활용할 때 유저 정보를 제공하기 위해 호출된다.	
모듈	이름	설명
	Amplify	서버에 AWS Amplify.Auth로 Cognito 인증 기능을 사용할 모듈
	Serializable	Fragment나 Activity 간의 데이터를 콘텐츠 정보 객체 형태로 전달하기 위한 인터페이스 모듈
주요 메소드	이름	설명
	.Auth.currentUser	현재 어플에 접속 중인 인증 유저의 회원 정보를 제공
추가설명	– Amplify에서 지원하는 Auth인증 정보에서 제공하는 다양한 메소드를 활용하여 앞으로 게시판이나 리뷰 등 보안적인 접근요소에 필터링 용도로 활용될 예정	

■ [통합 검색 UI 모듈]		
기능	■ [콘텐츠 정보를 검색하고 카테고리에 맞춰 출력 리스트를 필터링하는 모듈] – 검색 키워드(콘텐츠 제목)를 통해 DB에 저장된 콘텐츠 정보를 검색하는 역할을 하고 사용자가 요청하는 때마다 수시로 호출된다.	
모듈	이름	설명
	RecyclerView	response된 콘텐츠 리스트의 동적인 크기에 따라 유동적으로 표현해주는 View 모듈
	ArrayAdapter	검색 필터링 카테고리 목록을 Spinner에 표현하기 위해 사용되는 Adapter 모듈
	AdapterView	검색 필터링 카테고리 List 데이터를 나열하는 View 모듈
	GridLayoutManager	동적 콘텐츠 데이터 리스트를 View에 출력할 때 배치와 재사용을 결정하는 레이아웃 매니저 모듈
주요 메소드	이름	설명
	.adapter	동적 View의 표현 어댑터를 지정
	.layoutManager	동적 View의 레이아웃 매니저를 지정
	.onItemSelectedListener()	Spinner의 선택된 항목에 따른 리스너 설정
	.contains()	검색 필터링 시 해당 카테고리가 콘텐츠 정보에 포함되었는지 확인하는 메소드
	.notifyDataSetChanged()	사용자의 새로운 요청에 따라 View 재사용이 이뤄졌을 때 새로고침

■ [평가 및 리뷰 UI 모듈]		
기능	■ [콘텐츠에 대한 평가와 리뷰를 작성하고, 작성한 리뷰를 콘텐츠 정보에서 출력하는 모듈] – 콘텐츠에 대한 평점과 리뷰를 작성, 수정, 삭제할 수 있는 대화상자 형태 UI를 출력하는 역할을 하고 해당화면 실행 시 호출된다. – 사용자가 작성한 리뷰 리스트와 콘텐츠에 작성된 리뷰 리스트를 출력하는 역할을 하고 요청될 때마다 호출된다.	
모듈	이름	설명
	RecyclerView	response된 콘텐츠 리뷰 리스트의 동적인 크기에 따라 유동적으로 표현해주는 View 모듈
	AdapterView	콘텐츠 리뷰 List 데이터를 나열하는 View 모듈
	LinearLayoutManager	동적 콘텐츠 리뷰 리스트를 View에 출력할 때 배치와 재사용을 결정하는 레이아웃 매니저 모듈
주요 메소드	이름	설명
	.adapter	동적 View의 표현 어댑터를 지정
	.layoutManager	동적 View의 레이아웃 매니저를 지정
	callRating()	사용자가 콘텐츠에 대한 평가 및 리뷰 작성을 서버에 요청하는 메소드
	callAllReview()	콘텐츠 상세정보 화면을 요청할 때마다 해당 콘텐츠의 작성된 리뷰를 요청하는 메소드
	myReview()	사용자가 작성한 모든 리뷰를 요청하는 메소드
	likeReview()	콘텐츠 상세정보에 작성된 리뷰들에 좋아요를 할 수 있는 메소드

4.4 S/W 모듈 구성도 – 서버 * 현재까지 구현한 내용만 작성.

– BackEnd 모듈 구성도



– 모듈 설명

<table>
<tr><th colspan="3">■ [OTT 플랫폼 연동 모듈]</th></tr>
<tr><td>기능</td><td colspan="2">■ [OTT 플랫폼 회원 정보를 DB에 저장하고 웹 페이지에 로그인하는 모듈]
– 사용자가 입력한 OTT 서비스 회원 정보를 DB에 저장하는 역할을 하고 사용자의 요청이 있을 때 호출된다.
– OTT 플랫폼 웹 페이지에 로그인하는 역할을 하고 웹 크롤링이나 웹 자동화를 수행하기 전에 호출된다.</td></tr>
<tr><td rowspan="5">모듈</td><td>이름</td><td>설명</td></tr>
<tr><td>connection</td><td>Django에 연동된 DB에 접속하여 SQL 구문을 실행하기 위해 사용하는 Django 모듈</td></tr>
<tr><td>selenium</td><td>OTT 플랫폼의 웹 페이지에서 필요한 데이터들을 크롤링할 수 있도록 하는 모듈</td></tr>
<tr><td>pyvirtualdisplay</td><td>Ubuntu Server 상에서 selenium을 사용할 수 있도록 하는 가상 디스플레이 모듈</td></tr>
<tr><td>time</td><td>selenium 동작 중 딜레이를 주기 위해 사용하는 모듈</td></tr>
<tr><td rowspan="5">주요 메소드</td><td>이름</td><td>설명</td></tr>
<tr><td>Interworking()</td><td>사용자가 이용 중인 OTT 플랫폼의 ID, Password를 입력하여 연동을 요청하면 DB의 User_Interworking 테이블에 해당 정보를 입력하여 필요할 때 사용할 수 있도록 함</td></tr>
<tr><td>n_login()</td><td>사용자가 입력한 Netflix ID, Password와 프로필 명을 통해 selenium으로 Netflix 웹 페이지에 로그인</td></tr>
<tr><td>w_login()</td><td>사용자가 입력한 Watcha ID, Password와 프로필 명을 통해 selenium으로 Watcha 웹 페이지에 로그인</td></tr>
<tr><td>wav_login()</td><td>사용자가 입력한 Wavve ID, Password와 프로필 명을 통해 selenium으로 Wavve 웹 페이지에 로그인</td></tr>
</table>

■ [웹 크롤링 데이터 수집 모듈]

기능	■ [OTT 서비스의 웹 페이지 데이터를 크롤링하는 모듈] – 각각의 OTT 서비스에서 필요한 정보들을 크롤링하는 역할을 하고, 사용자의 요청이 있을 때 호출된다. – 사용자가 이용 중인 OTT 플랫폼의 찜 목록, 시청 기록을 크롤링하는 동작을 수행하고 사용자의 갱신 요청이 있을 때 호출된다.	
모듈	이름	설명
	selenium	OTT 플랫폼의 웹 페이지에서 필요한 데이터들을 크롤링할 수 있도록 하는 모듈
	pyvirtualdisplay	Ubuntu Server 상에서 selenium을 사용할 수 있도록 하는 가상 디스플레이 모듈
	time	selenium 동작 중 딜레이를 주기 위해 사용하는 모듈
	models	Django에서 DB와 연동하여 데이터를 불러오거나 삽입할 수 있도록 하는 Django모듈
주요 메소드	이름	설명
	n_watchings()	사용자가 Netflix에서 시청 중인 콘텐츠 리스트를 크롤링하여 반환함
	n_wishes()	사용자가 Netflix에서 찜 목록에 추가한 콘텐츠 리스트를 크롤링하여 반환함
	w_watchings()	사용자가 Watcha에서 시청 중인 콘텐츠 리스트를 크롤링하여 반환함
	w_wishes()	사용자가 Watcha에서 찜 목록에 추가한 콘텐츠 리스트를 크롤링하여 반환함
	wav_watchings()	사용자가 Wavve에서 시청 중인 콘텐츠 리스트를 크롤링하여 반환함
	wav_wishes()	사용자가 Wavve에서 찜 목록에 추가한 콘텐츠 리스트를 크롤링하여 반환함
	add_watchinglog()	크롤링한 시청 목록의 영화 제목 리스트를 DB의 Contents 데이터 리스트로 변환하여 DB에 저장함
	add_wishlist()	크롤링한 찜 목록의 영화 제목 리스트를 DB의 Contents 데이터 리스트로 변환하여 DB에 저장함
추가설명	– 사용자가 회원 가입하거나 해당 플랫폼 최초 연동 시, 새로고침 요청 시에만 동작하여 DB에 데이터를 추가하도록 할 때 해당 모듈의 메소드 사용 – 각각 OTT 플랫폼별 콘텐츠 제목과 DB의 콘텐츠 제목이 정확히 일치하지 않는 경우가 있으므로 일치하도록 수정 필요	

■ [웹 자동화 수행 모듈]		
기능	■ [OTT 서비스의 웹 페이지에서 자동화를 수행하는 모듈] – OTT 플랫폼에 웹 페이지에 접속하여 자동화를 수행하며 사용자가 찜 목록에 콘텐츠를 추가할 때 호출된다.	
모듈	이름	설명
	selenium	OTT 플랫폼의 웹 페이지에서 자동화를 수행할 수 있도록 하는 모듈
	pyvirtualdisplay	Ubuntu Server 상에서 selenium을 사용할 수 있도록 하는 가상 디스플레이 모듈
	time	selenium 동작 중 딜레이를 주기 위해 사용하는 모듈
주요 메소드	이름	설명
	n_addwish()	사용자가 찜 목록에 추가하고자 하는 콘텐츠의 Title을 이용해 Netflix 찜 목록에 해당 콘텐츠 추가
	w_addwish()	사용자가 찜 목록에 추가하고자 하는 콘텐츠의 Title을 이용해 Watcha 찜 목록에 해당 콘텐츠 추가
	wav_addwish()	사용자가 찜 목록에 추가하고자 하는 콘텐츠의 Title을 이용해 Wavve 찜 목록에 해당 콘텐츠 추가
추가설명	사용자가 Wishlist에 콘텐츠를 추가하면 DB에는 바로 반영되고 백그라운드에서 자동화가 수행되도록 수정 필요	

■ [사용자 관리 모듈]		
기능	■ [회원 가입, 로그인 기능을 수행하는 모듈] – 사용자의 정보를 입력받아 DB에 추가하는 역할을 하고 사용자가 회원 가입을 요청할 때 호출된다. – 사용자가 입력한 ID, Password를 DB의 정보와 비교하는 역할을 하고 사용자의 로그인 요청이 있을 때 호출된다.	
모듈	이름	설명
	connection	Django에 연동된 DB에 접속하여 SQL 구문을 실행하기 위해 사용하는 Django 모듈
주요 메소드	이름	설명
	signUp()	사용자가 입력한 회원 정보를 DB에 추가하고 성공 여부를 반환
	signIn()	사용자가 입력한 ID, Password가 DB에 저장된 정보와 일치하는지 비교하여 결과를 반환(True/False)

■ [콘텐츠 검색 모듈]		
기능	■ [DB에서 콘텐츠를 검색하는 모듈] – 콘텐츠 Title을 통해 콘텐츠를 검색하여 결과를 전달하는 역할을 하고 사용자의 콘텐츠 검색 요청이 있을 때 호출된다.	
모듈	이름	설명
	connection	Django에 연동된 DB에 접속하여 SQL 구문을 실행하기 위해 사용하는 Django 모듈
주요 메소드	이름	설명
	search_title()	DB의 Contents 테이블에서 콘텐츠 Title을 통해 콘텐츠를 검색하는 Fulltext Search SQL 구문을 수행하고 검색 결과를 json 형식으로 변환하여 전달
	dictfetchall()	search_title에서 검색한 결과를 json(Dictionary) 형식으로 변환하여 전달할 수 있도록 바꿔줌
추가설명	제목 외에 장르, 플랫폼 등등 다른 콘텐츠 정보로도 검색할 수 있도록 추가 필요	

■ [사용자 데이터 요청 모듈]		
기능	■ [DB에서 사용자 데이터를 받아오는 모듈] – 사용자의 찜 목록, 시청 기록을 받아와 전달하는 역할을 하고 사용자의 요청이 있을 때 호출된다.	
모듈	이름	설명
	models	Django에서 DB와 연동하여 데이터를 불러오거나 삽입할 수 있도록 하는 Django모듈
주요 메소드	이름	설명
	load_interworking()	DB의 User_Interworking 테이블에서 사용자의 OTT 플랫폼 연동 정보를 불러와 반환
	wishlist()	DB의 Wishlist 테이블에서 사용자의 찜 목록을 불러와 반환. load_interworking을 이용해 사용자의 OTT 연동 정보를 받아옴
	watchinglog()	DB의 Watching_Log 테이블에서 사용자의 시청 기록을 불러와 반환. load_interworking을 이용해 사용자의 OTT 연동 정보를 받아옴
	userinfo()	DB의 Users 테이블에서 사용자의 정보를 불러와 반환
	MyReview()	DB의 ContentsReview 테이블에서 사용자가 남긴 모든 리뷰 데이터를 불러와 반환

■ [웹 크롤링 & 웹 자동화 모듈]

기능	■ [웹 크롤링 또는 웹 자동화를 수행하는 모듈] – OTT 플랫폼 웹 페이지에서 데이터를 크롤링하는 역할을 함 – OTT 플랫폼 웹 페이지에서 자동화를 수행하는 역할을 함	
모듈	이름	설명
	selenium	OTT 플랫폼의 웹 페이지에서 필요한 데이터들을 크롤링하거나 자동화를 수행할 수 있도록 하는 모듈
	pyvirtualdisplay	Ubuntu Server 상에서 selenium을 사용할 수 있도록 하는 가상 디스플레이 모듈
	time	selenium 동작 중 딜레이를 주기 위해 사용하는 모듈
주요 메소드	이름	설명
	.get()	입력한 URL의 웹 페이지를 로드해줌
	.find_element()	웹 페이지에서 class명, xpath 등으로 요소를 찾아줌
	.implicitly_wait()	웹 페이지가 로드될 때까지 대기하도록 함
	.sleep()	입력한 시간 동안 딜레이를 줌

■ [평점(Review) 모듈]

기능	■ [DB에 평점 데이터를 추가하거나 DB에서 평점 데이터를 불러오는 모듈] – 특정 컨텐츠에 리뷰 데이터를 추가하거나 해당 콘텐츠의 전체 리뷰 데이터, 사용자가 남긴 데이터를 불러온다.	
모듈	이름	설명
	models	Django에서 DB와 연동하여 데이터를 불러오거나 삽입할 수 있도록 하는 Django모듈
	connection	Django에 연동된 DB에 접속하여 SQL 구문을 실행하기 위해 사용하는 Django 모듈
주요 메소드	이름	설명
	Rating()	사용자가 입력한 리뷰 데이터를 DB에 추가하거나 기존의 리뷰 데이터를 새롭게 입력된 내용으로 수정
	CallReview()	해당 사용자의 특정 콘텐츠에 대한 리뷰 데이터를 DB에서 불러와 반환
	AllReview()	DB의 ContentsReview 테이블에서 에서 특정 콘텐츠의 전체 리뷰를 불러와 반환
	DeleteReview()	해당 사용자의 특정 콘텐츠에 대한 리뷰 데이터를 DB에서 삭제
	CallReviewLike()	해당 리뷰의 '좋아요' 카운트를 1 증가시키거나 감소시킴
추가설명		

<table>
<tr><th colspan="3">■ [콘텐츠 추천 모듈]</th></tr>
<tr><td>기능</td><td colspan="2">■ [Surprise 추천 알고리즘을 이용해 영화를 추천해주는 모듈]
– 따로 학습하여 저장한 SurpriseSVD 파일에서 학습된 알고리즘을 불러와서 영화 추천에 사용할 수 있도록 한다.
– 사용자의 contents_review 정보를 불러와 사용자가 아직 평점을 남기지 않은 영화의 예상 평점을 매기고, 예상 평점이 높은 200편의 영화를 추천해준다.</td></tr>
<tr><td rowspan="5">모듈</td><td>이름</td><td>설명</td></tr>
<tr><td>os</td><td>SurpriseSVD 파일이 저장된 위치를 불러오기 위해 사용하는 모듈</td></tr>
<tr><td>dump</td><td>surprise 라이브러리의 모듈 중 하나로, 이미 학습된 알고리즘 파일을 불러와 사용할 수 있도록 하는 모듈</td></tr>
<tr><td>pandas</td><td>불러온 사용자의 contents_review 데이터를 가공하거나 MovieLens Dataset을 불러오기 위해 사용하는 모듈</td></tr>
<tr><td>models</td><td>Django에서 DB와 연동하여 데이터를 불러오거나 삽입할 수 있도록 하는 Django모듈</td></tr>
<tr><td rowspan="5">주요 메소드</td><td>이름</td><td>설명</td></tr>
<tr><td>load_algo()</td><td>SurpriseSVD 파일을 불러와 파일에 저장되어있는 추천 알고리즘을 반환</td></tr>
<tr><td>load_dataset()</td><td>MovieLens의 영화, 평점 Dataset을 불러와 pandas DataFrame으로 반환</td></tr>
<tr><td>unseen_movies()</td><td>DB에서 사용자의 contents_review 목록을 불러와 전체 영화 목록 중 평점을 남긴 영화를 제외한 영화의 TMDB ID 리스트를 반환</td></tr>
<tr><td>recommend_movie_list()</td><td>위 메소드에서 불러온 알고리즘, 영화 데이터셋, TMDB ID 리스트를 이용해 예상 평점을 계산하고 그것을 기반으로 200편의 영화 정보와 예상 평점 리스트를 반환</td></tr>
<tr><td>추가설명</td><td colspan="2">– 주기적으로 사용자 평점 데이터를 수집하여 추천 알고리즘을 재학습시켜 저장하도록 수정 필요</td></tr>
</table>

■ [OTT 컨텐츠 링크 모듈]

기능	■ [Selenium 자동화를 이용해 컨텐츠의 OTT별 URL링크를 불러오는 모듈] – 컨텐츠의 정보를 받아서 OTT 플랫폼별로 컨텐츠 제목 검색을 통해 컨텐츠 시청 URL을 찾아오는 역할을 함	
모듈	이름	설명
	selenium	OTT 플랫폼의 웹 페이지에서 필요한 데이터들을 크롤링하거나 자동화를 수행할 수 있도록 하는 모듈
	time	Selenium 동작 중 find_element 메소드를 사용할 때 요소를 찾을 수 있도록 딜레이를 걸기 위해 사용하는 모듈
주요 메소드	이름	설명
	find_element()	웹 페이지에서 요소를 찾아줌
	netflix_url()	Netflix 웹페이지에 접속해서 컨텐츠 제목과 타입으로 컨텐츠 검색을 한 후 해당 웹 URL을 반환
	disney_url()	Disney Plus 웹페이지에 접속해서 컨텐츠 제목과 타입으로 컨텐츠 검색을 한 후 해당 웹 URL을 반환
	wavve_url()	Wavve 웹페이지에 접속해서 컨텐츠 제목과 타입으로 컨텐츠 검색을 한 후 해당 웹 URL을 반환
	watcha_url()	Watcha 웹페이지에 접속해서 컨텐츠 제목과 타입으로 컨텐츠 검색을 한 후 해당 웹 URL을 반환
	tving_url()	Tving 웹페이지에 접속해서 컨텐츠 제목과 타입으로 컨텐츠 검색을 한 후 해당 웹 URL을 반환

4.5 데이터베이스 설계 * 현재까지 구현한 내용만 작성.

■ [테이블 명] Contents	
기본키	[id]
설명	■ TV 프로그램이나 영화의 상세 정보를 저장해 놓는 테이블이다.
구성	Contents

id	varchar(100)	not null	PRIMARY_KEY	고유번호
type	varchar(10)	not null		콘텐츠타입(TV or Movie)
title	varchar(100)	not null		제목
genre	varchar(100)	not null		장르(리스트)
production_countries	varchar(200)	not null		제작 나라
vote_count	int	not null		평점 수
vote_average	float(3,1)	not null		평점
number_of_seasons	int			시즌 수
number_of_episodes	int			에피소드 수
release_date	date			첫 방영일(개봉일)
adult	boolean			청불
poster_path	varchar(200)			포스터 주소
runtime	int			런타임(분)
overview	varchar(2000)			설명
now_status	varchar(100)			방영중인 상태(TV용)
flatrate	varchar(100)			구독 서비스(리스트)
rent	varchar(100)			대여 서비스(리스트)
buy	varchar(100)			구매 서비스(리스트)

■ [테이블 명] users	
기본키	[email]
설명	■ OOSOOSOO 서비스를 이용하는 사용자들의 정보를 저장해 놓는 테이블이다.
구성	users email varchar(50) not null PRIMARY_KEY 유저 이메일 passwd varchar(100) not null 비밀번호 name varchar(20) not null 실명 phone_number varchar(20) not null 전화번호 nickname varchar(20) not null 별명 gender boolean not null 성별 birthday varchar(10) not null 생년월일 job varchar(20) 직업 overview varchar(200) 간단한 소개 coin int 코인

■ [테이블 명] user_interworking	
기본키	[i_id]
외부 참조키	[u_email] users 테이블의 email을 참조
설명	■ OTT 서비스와 연동을 위한 정보들을 저장해 놓는 테이블이다.
구성	user_interworking i_id varchar(50) not null PRIMARY_KEY 유저 연동 고유번호 u_email varchar(50) not null FK 유저 이메일 platform varchar(20) not null 연동할 플랫폼 id varchar(50) not null 연동 아이디 passwd varchar(20) not null 연동 비밀번호 profile_name varchar(20) not null 연동 프로필 이름 expiration_date date 연동 만료 날짜 simple_login varchar(50) 간편로그인

■ [테이블 명] contents_seasons	
기본키	[_id]
외부 참조키	[c_id] contents 테이블의 id를 참조
설명	■ TV 프로그램의 시즌에 대한 상세 정보들을 저장해 놓는 테이블이다.
구성	contents_seasons _id varchar(100) not null PRIMARY_KEY 시즌 고유번호 c_id varchar(100) not null FK 콘텐츠 고유번호 air_date date not null 첫 방영일 title varchar(100) not null 시즌 명 number int not null 시즌 번호 overview varchar(2000) 설명 poster_path varchar(200) 포스터 주소

■ [테이블 명] contents_episodes	
기본키	[_id]
외부 참조키	[c_id] contents 테이블의 id를 참조
설명	■ TV 프로그램의 에피소드에 대한 상세 정보들을 저장해 놓는 테이블이다.
구성	contents_episodes _id varchar(100) not null PRIMARY_KEY 에피소드 고유번호 c_id varchar(100) not null FK 콘텐츠 고유번호 season_num int not null 시즌 번호 title varchar(100) not null 에피소드 명 number int not null 에피소드 번호 air_date date 첫 방영일 vote_count int 평점 수 vote_average float(3, 1) 평점 overview varchar(2000) 설명 still_path varchar(200) 스틸패스 crew varchar(1000) 크루들 이름(리스트) guests varchar(1000) 게스트들 이름(리스트)

■ [테이블 명] contents_review	
기본키	[id]
외부 참조키	[c_id] contents 테이블의 id를 참조 [u_email] users 테이블의 email을 참조
설명	■ 어떤 콘텐츠에 어떤 사용자가 리뷰를 남긴 지에 대한 정보가 있는 테이블이다.
구성	contents_review id int not null PRIMARY_KEY AUTO_INCREMENT 리뷰 고유번호 c_id varchar(100) not null FK 콘텐츠 고유번호 u_email varchar(50) not null FK 유저 이메일 _like boolean not null 좋아요 rating float(3,1) not null 작성자의 평점 review varchar(1000) not null 리뷰 _datetilme datetime not null 리뷰를 남긴 날짜와 시간

■ [테이블 명] wish_list	
기본키	[id]
외부 참조키	[c_id] contents 테이블의 id를 참조 [i_id] user_interworking 테이블의 id를 참조
설명	■ OTT 서비스에 있는 찜 목록을 OOSOOSOO 서비스와 연동하기 위한 정보들을 저장해 놓은 테이블이다.
구성	wish_list id int not null PRIMARY_KEY AUTO-INCREMENT 찜 고유번호 c_id varchar(100) not null FK 콘텐츠 고유번호 i_id varchar(50) not null FK 연동 고유번호

■ [테이블 명] watching_log	
기본키	[id]
외부 참조키	[c_id] contents 테이블의 id를 참조 [i_id] user_interworking 테이블의 id를 참조
설명	■ OTT 서비스에 있는 시청중인 목록을 OOSOOSOO 서비스와 연동하기 위한 정보들을 저장해 놓은 테이블이다.
구성	watching_log id int not null PRIMARY_KEY AUTO-INCREMENT 시청 기록 고유번호 c_id varchar(100) not null FK 콘텐츠 고유번호 i_id varchar(50) not null FK 연동 고유번호

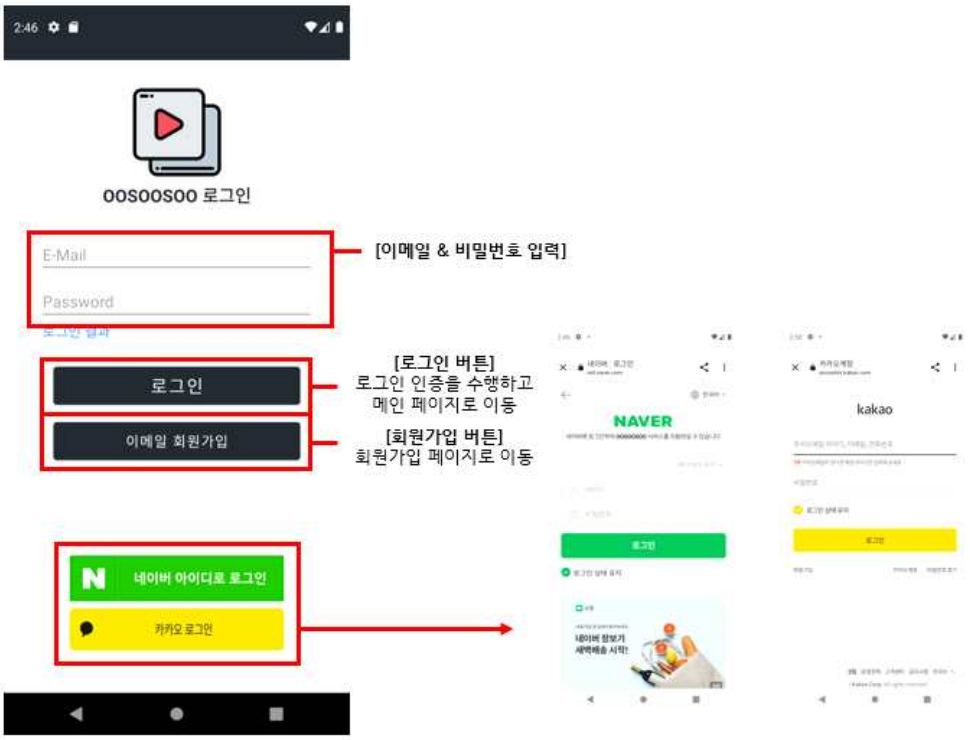
■ [테이블 명] contents_credits	
기본키	[id]
외부 참조키	[c_id] contents 테이블의 id를 참조 [p_id] people 테이블의 id를 참조
설명	■ 어떤 콘텐츠에 어떤 인물이 참여하였는지에 대한 정보가 있는 테이블이다.
구성	contents_credits id int not null PRIMARY_KEY 크레딧 고유번호 c_id varchar(100) not null FK 콘텐츠 고유번호 p_id int not null FK People 고유번호 job varchar(20) not null 직업 (배우 or 감독) role varchar(20) 극중 역할

■ [테이블 명] people	
기본키	[id]
설명	■ 배우, 감독 등 인물의 정보가 있는 테이블이다.
구성	people id int not null PRIMARY_KEY People 고유번호 name varchar(50) not null 이름 birthday varchar(20) not null 생년월일 department varchar(20) not null 주 직업 (연기 or 감독) popularity float not null 유명도 profile_path varchar(50) 인물 사진 URL

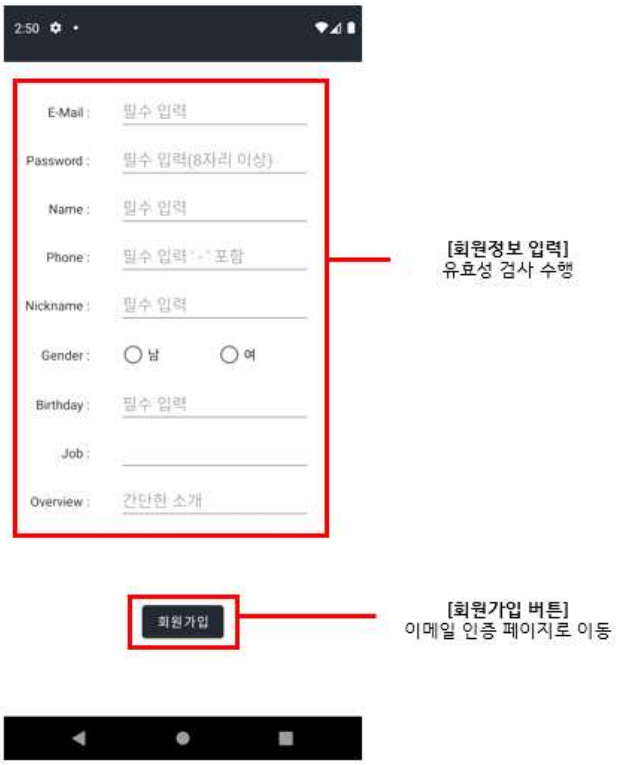
V Prototype 구현

■ 프로토타입 세부 구성

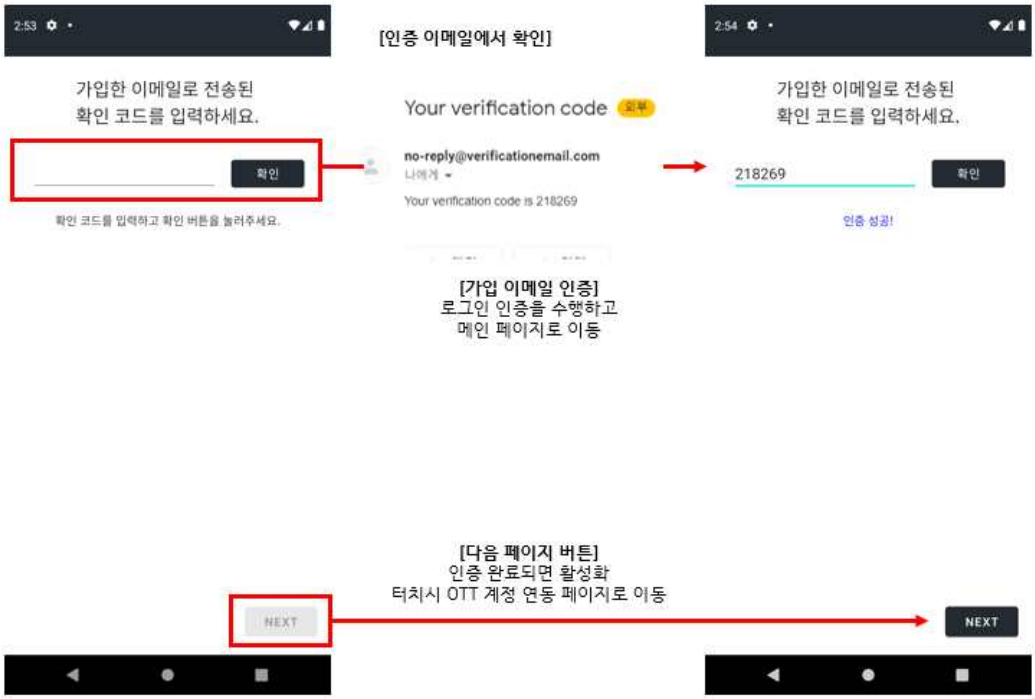
① 로그인 페이지 프로토타입



② A. 회원가입 페이지 프로토타입

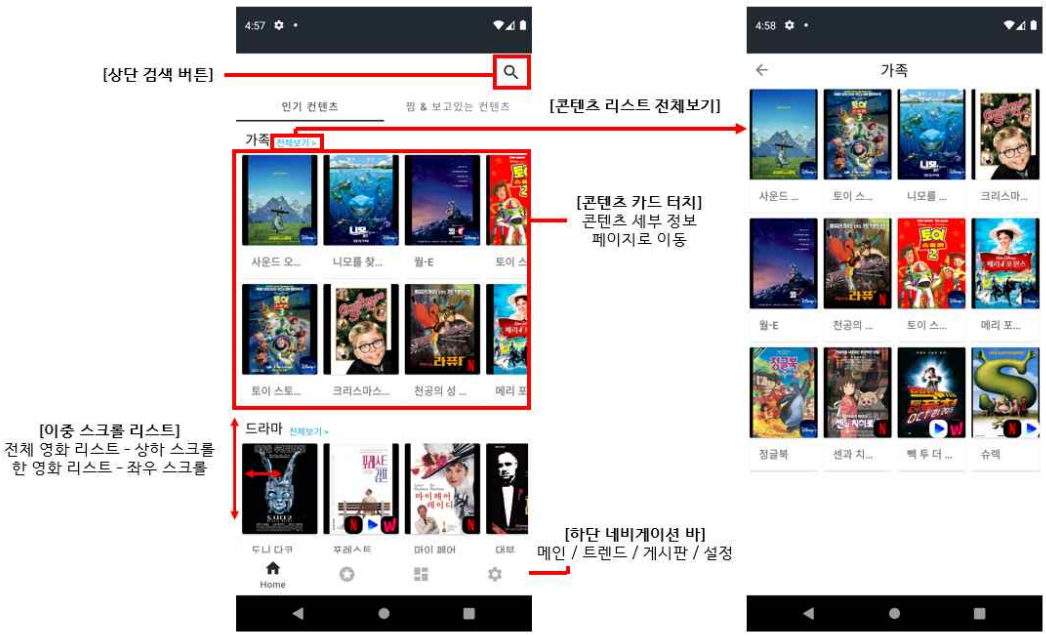


② B. 가입 인증 페이지 프로토타입

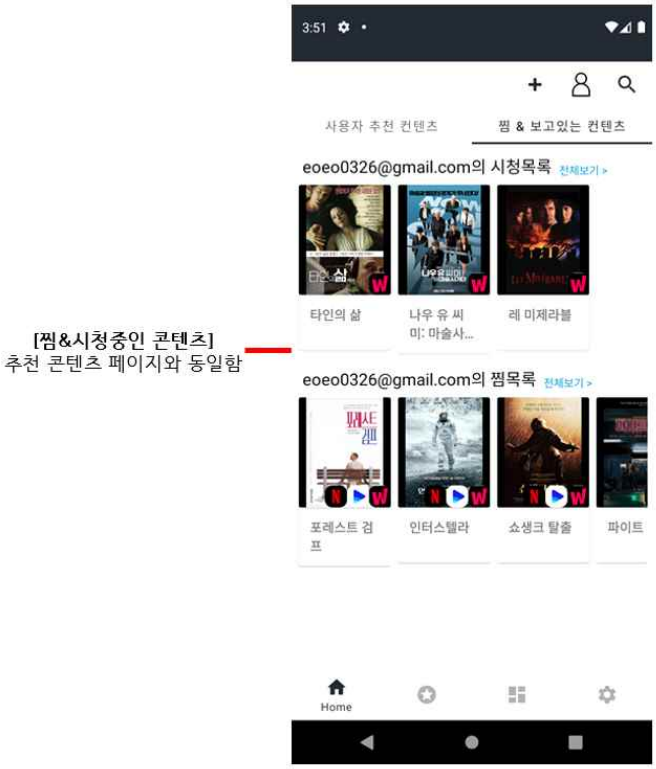


③ OTT 계정 연동 페이지 프로토타입

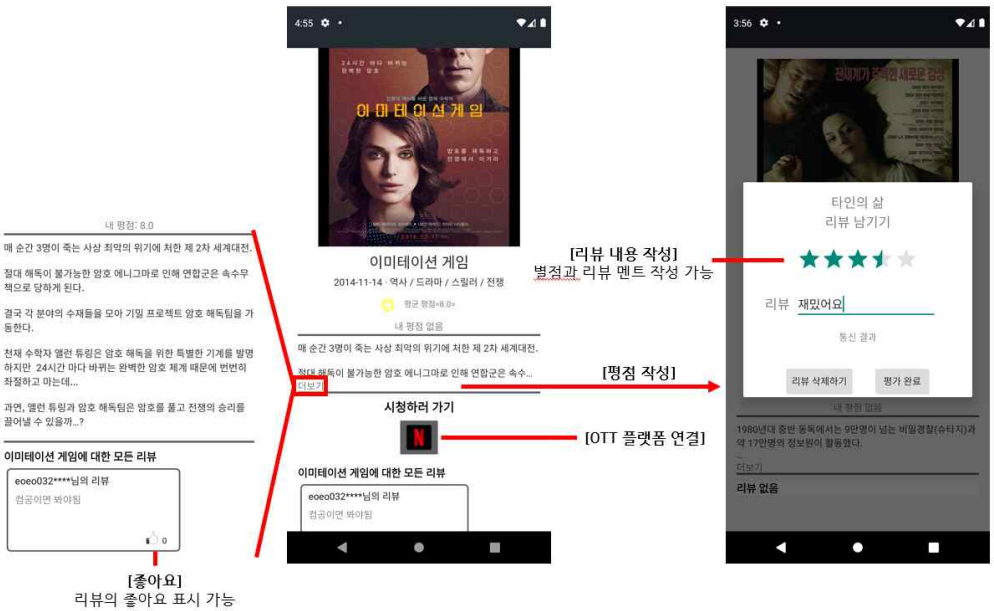
④ A. 메인 페이지1 (인기 콘텐츠) 프로토타입



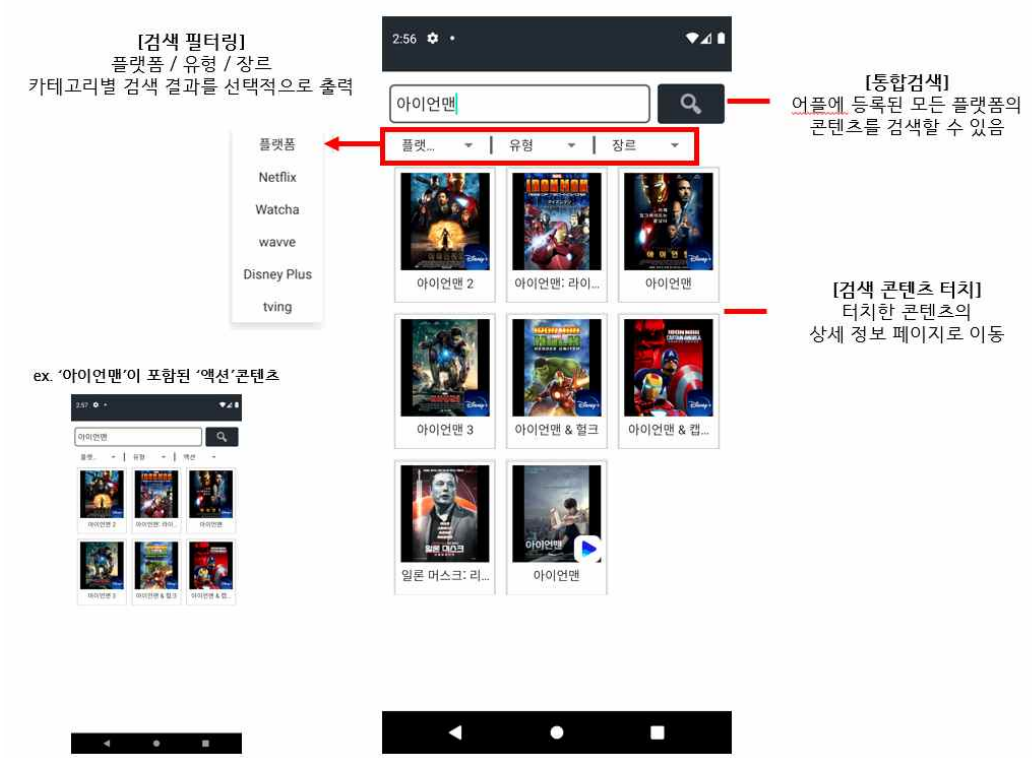
④ B. 메인 페이지2 (찜 & 시청중인 콘텐츠) 프로토타입



⑤ 콘텐츠 상세 페이지 프로토타입

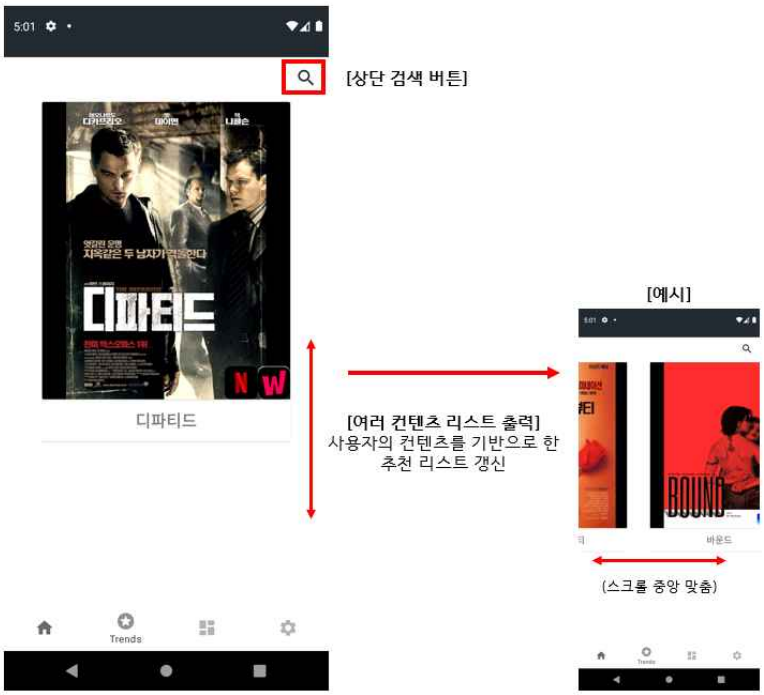


⑥ 검색 페이지 프로토타입



⑦ 내 정보 조회 페이지 프로토타입

⑧ 추천 컨텐츠 페이지 프로토타입



⑨ 설정 페이지 프로토타입

VI 시험 / 테스트 결과

1. 데이터 일치 테스팅 (실제 데이터 반영)

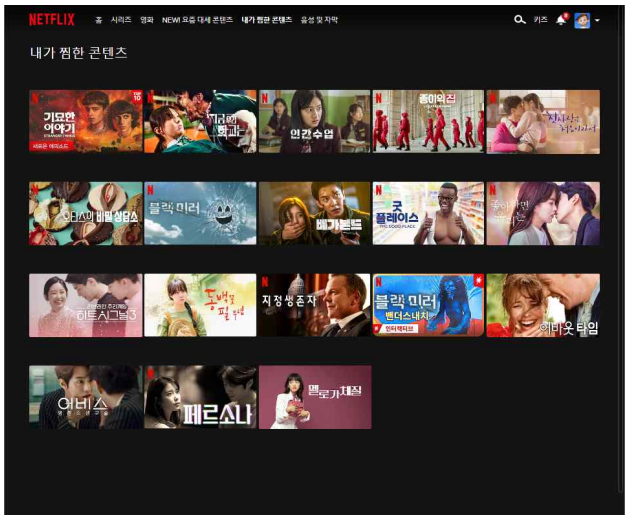
■ 데이터 일치 여부 확인 (각 플랫폼에서 모두 진행) – 예시) Netflix

I. Netflix 찜목록

– Netflix_Wish.py 실행결과

```
C:\Users\ncclu\PycharmProjects\OOSOO\venv\Scripts\python.ex
넷플릭스 로그인 이메일 또는 전화번호를 입력해주세요 : [redacted]
넷플릭스 비밀번호를 입력해주세요 : [redacted]
로그인 중...
        <프로필 리스트>
------------------------------
[redacted]
------------------------------
넷플릭스 프로필 이름을 입력해주세요(프로필 선택) : 찬스
찜목록 불러오는 중...
        < 찜목록 >
------------------------------
기묘한 이야기
지금 우리 학교는
인간수업
종이의 집
첫사랑은 처음이라서
오티스의 비밀 상담소
블랙 미러
배가본드
굿 플레이스
좋아하면 울리는
하트시그널
동백꽃 필 무렵
지정생존자
블랙 미러: 밴더스내치
어바웃 타임
어비스
페르소나
멜로가 체질
------------------------------
```

– 실제 플랫폼 찜한 목록 페이지 (Netflix.com)



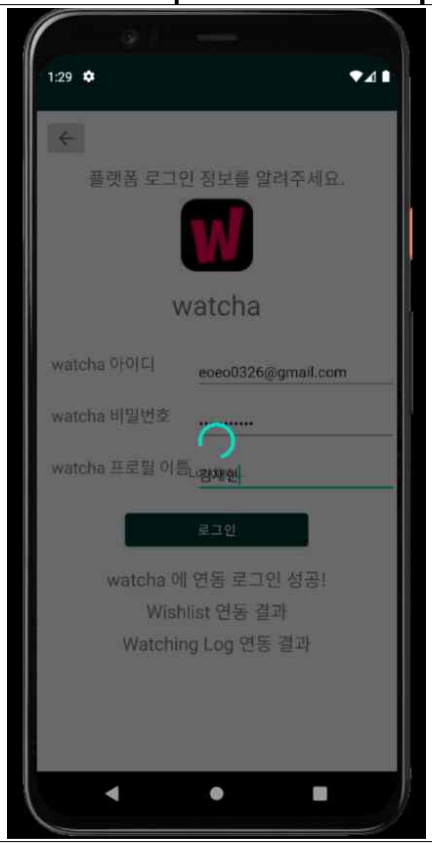
II. Netflix 시청중인 목록

- Netflix_Watching.py 실행결과

```
넷플릭스 로그인 이메일 또는 전화번호를 입력해주세요 :
넷플릭스 비밀번호를 입력해주세요 :
로그인 중...
        <프로필 리스트>
------------------------------
차노  히주  키즈
------------------------------
넷플릭스 프로필 이름을 입력해주세요(프로필 선택) :
시청중인 목록 불러오는 중...
        < 시청중인 목록 >
------------------------------
나의 해방일지
장르만 로맨스
------------------------------

Process finished with exit code 0
```

- 실제 플랫폼 시청중인 목록 페이지 (Netflix.com)

2. 연동 테스팅 (플랫폼별 진행)

■ Watcha 계정연동 및 작업시간 테스트			
테스트 개요	서버에서 Selenium과 BeautifulSoup 등을 사용하여 웹 자동화 모듈을 통해 선택된 OTT 플랫폼에 로그인하고 성공 여부에 따라 DB에 계정정보를 저장한다. 이 기능은 시간복잡성, 효율성과 데이터 일치에 중점을 두고 테스트한다.		
전체 기능 시간	평균 73초	데이터 일치 오류	0% (10/10)
테스트 예시			
해결 방안	계정 최초 로그인 시점에 세션 정보의 쿠키를 저장하여 이를 통해 로그인 절차를 생략하여 OTT 플랫폼 연동 시간 복잡성을 완화한다.		

VII 주요 코드

1. 웹 크롤링 (시청 중인 목록 불러오기)

〈 Watching_log.py 〉

```
from bs4 import BeautifulSoup          #BeautifulSoup4 import
from selenium import webdriver         #Selenium webdriver import
from pyvirtualdisplay import Display
import time
from api.DisneyPlus.Login import d_login

def d_watchings(email, pwd, name):
    # display = Display(visible=0, size=(1920, 1080))  # PyCharm 테스트시 주석처리
    # display.start()  # PyCharm 테스트시 주석처리

    # chrome창(웹드라이버) 열기
    path = "/webserver/chromedriver"
    options = webdriver.ChromeOptions()  #Webdriver 옵션 설정
    options.add_argument('--headless')
    options.add_argument('--no-sandbox')
    options.add_argument('disable-gpu')  #브라우저Display disable : GPU 사용량 완화
    options.add_argument('--disable-dev-shm-usage')
    driver = webdriver.Chrome(path, chrome_options=options) #Webdriver (Chrome) 생성
    driver.maximize_window()

    #디플 로그인 시작
    driver.get('https://www.disneyplus.com/login/')    #로그인URL 접속
    driver.implicitly_wait(10)

    # 로그인 자동화
    driver.find_element(By.NAME, 'email').send_keys(email)  #아이디 전송
    driver.find_element(By.NAME, 'dssLoginSubmit').click()  #아이디 입력
    driver.implicitly_wait(5)

    driver.find_element(By.NAME, 'password').send_keys(pwd) #비밀번호 전송
    driver.find_element(By.NAME, 'dssLoginSubmit').click()  #비밀번호 입력
    driver.implicitly_wait(5)

    # 프로필 리스트 출력 후 선택
    userProfiles = driver.find_elements(By.XPATH,
        '//*[@id="remove-main-padding_index"]/div/div/section/ul/div')

         # 캡쳐된 프로필 리스트 중 요청된 프로필 이름과 동일한 객체 클릭
    for profile in userProfiles:
        if profile.text.strip() == name:
            profile.click()
            break

    time.sleep(3)
    driver.implicitly_wait(5)

    result = list()

    # 페이지 스크롤
```

```
    last_height = driver.execute_script("return document.body.scrollHeight")

                    # 페이지 맨 아래까지 계속 스크롤
    while True:
        scroll_down = 0
        while scroll_down < 10:
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
            time.sleep(0.2)
            scroll_down += 1

        # 스크롤 내린 후 스크롤 높이 다시 가져옴
        new_height = driver.execute_script("return document.body.scrollHeight")
        if new_height == last_height:
            break
        last_height = new_height

    # selenium(driver)의 동적 페이지-> bs4의 정적 페이지 형태로source 전달
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')

    try:
         #시청중인 목록 개체 지정
        watchings = soup.select('#home-collection > div:nth-child(6) > div > div > div > div >
div > div')

        # cross-classify type : 컨텐츠의 종류 분리
        for watching in watchings:
            movie = watching.find("div", attrs={"data-program-type": "movie"}) #영화 컨텐츠
            episode = watching.find("div", attrs={"data-program-type": "episode"}) #드라마 컨텐츠

            if movie:
                result.append(movie.select('#asset-metadata > h5')[0].text) #시청중인 영화 저장
            elif episode:
                result.append(episode.select('#asset-metadata > p')[0].text) #시청중인 드라마 저장
    except:
        #예외 처리
        print("데이터 로드에 오류가 발생했습니다.")
    finally:
        print(result)

    driver.quit() #selenium driver 작업 종료
    display.stop() #chrome 종료

    return result  #결과리스트 반환
```

2. 콘텐츠 추천 알고리즘 (추천 알고리즘 모델 학습 및 생성)

〈 MovielensSampleSVD.ipynb 〉

```
#모듈 불러오기
import mariadb
import sys
import pandas as pd
import numpy as np

from surprise import Dataset
from surprise import Reader
from surprise.model_selection import train_test_split
from surprise import SVD
from surprise import accuracy
from surprise.model_selection import cross_validate
from surprise.model_selection import GridSearchCV
from surprise.dataset import DatasetAutoFolds

## 데이터 불러오기(데이터 프레임)
# 다운로드 링크: https://files.grouplens.org/datasets/movielens/ml-25m.zip
ratings = pd.read_csv('./ml-25m/ratings.csv')
links = pd.read_csv('./ml-25m/links.csv')
movies = pd.read_csv('./ml-25m/movies.csv')

## Movies, Ratings 데이터 불러오기(MovieLens)
# 데이터 필터링(필요한 데이터만 원하는 형식으로 변환)
links = links.fillna(0)
links = links.astype({'tmdbId':'int64'})

ratings_combined = pd.merge(links, ratings, on='movieId')
ratings_combined.drop(['movieId', 'imdbId', 'timestamp'], axis=1, inplace=True)

movies_combined = pd.merge(links, movies, on='movieId')
movies_combined.drop(['movieId', 'imdbId', 'genres'], axis=1, inplace=True)

#파일 형태로 저장
with open('./ml-25m/ratings.csv','r') as f:
with open('./ml-25m/ratings_noh.csv','w') as f1:
next(f) # skip header line
for line in f:
f1.write(line)

## Surprise SVD
# Reader 객체 생성
reader = Reader(line_format='user item rating timestamp', sep=',', rating_scale=(0.5, 5))

# DatasetAutoFolds 객체 생성(index 제거한ratings_noh 파일 사용)
data_folds = DatasetAutoFolds(ratings_file='./ml-25m/ratings_noh.csv', reader=reader)

# 전체 데이터를trainset으로 지정
trainset = data_folds.build_full_trainset()

# Reader 객체 생성
reader = Reader(rating_scale=(0.5, 5))
```

```python
surprise_data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)

# surprise의train_test_split() 사용. trainset : testset = 3 : 1
trainset, testset = train_test_split(surprise_data, test_size=0.5, random_state=0)

algo = SVD() # GridSearchCV를 이용한 최적 하이퍼 파라미터 적용 필요
algo.fit(trainset)

# 사용자 아이디(uid), 아이템 아이디(iid)는 문자열로 입력
uid = str(196)
iid = str(302)

# 추천 예측 평점(.predict)
pred = algo.predict(uid, iid)

# 추천 예측 평점(.test)
predictions = algo.test( testset )

## 성능 평가
accuracy.rmse(predictions)

## 최적 파라미터 탐색
# n_epochs: SGD 수행 시 반복 횟수, n_factors: 잠재 요인 크기
param_grid = {
'n_epochs': [20, 40, 60],
'n_factors': [50, 100, 200]
}

# GridSearchCV
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3) # algo가 아닌SVD
입력하였다.
gs.fit(surprise_data)

# 최적 하이퍼 파라미터 및 그 때의 최고 성능
print(gs.best_params['rmse'])
print(gs.best_score['rmse'])

## 예상 평점 기반 영화 추천
# 아직 보지 않은 영화 리스트 함수
def get_unseen_surprise(ratings, movies, userId):
# 특정userId가 평점을 매긴 모든 영화 리스트
seen_movies = ratings[ratings['userId'] == userId]['movieId'].tolist()

# 모든 영화명을list 객체로 만듬.
total_movies = movies['movieId'].tolist()

# 한줄for + if문으로 안 본 영화 리스트 생성
unseen_movies = [movie for movie in total_movies if movie not in seen_movies]

# 일부 정보 출력
total_movie_cnt = len(total_movies)
seen_cnt = len(seen_movies)
unseen_cnt = len(unseen_movies)

print(f"전체 영화 수: {total_movie_cnt}, 평점 매긴 영화 수: {seen_cnt}, 추천 대상 영화 수:
{unseen_cnt}")
```

```python
return unseen_movies


# 예상 평점 기반 영화 리스트 추천 함수
def recomm_movie_by_surprise(algo, userId, unseen_movies, top_n=10):
# 아직 보지 않은 영화의 예측 평점: prediction 객체 생성
predictions = []
for movieId in unseen_movies:
predictions.append(algo.predict(userId, movieId))

# 리스트 내의prediction 객체의est를 기준으로 내림차순 정렬
def sortkey_est(pred):
return pred.est

predictions.sort(key=sortkey_est, reverse=True)  # key에 리스트 내 객체의 정렬 기준을 입력

# 상위top_n개의prediction 객체
top_predictions = predictions[:top_n]

# 영화 아이디, 제목, 예측 평점 출력
print(f"Top-{top_n} 추천 영화 리스트")

for pred in top_predictions:
movie_id = int(pred.iid)
movie_title = movies[movies["movieId"] == movie_id]["title"].tolist()
movie_rating = pred.est

print(f"{movie_id}- {movie_title}: {movie_rating:.2f}")

unseen_movies = get_unseen_surprise(ratings, movies, 234)
recomm_movie_by_surprise(algo, 234, unseen_movies, top_n=20)
```

3. 영화 추천 (추천 알고리즘 반영된 영화 추천)

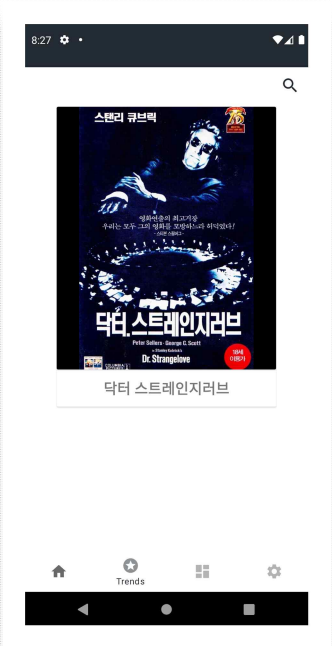
〈 RecommendMovies.py 〉

```
from api.models import Contents


def recommend_movie_list(algo, userId, unseen_movies, movies, top_n=10):
recommend_list = list()

# 아직 보지 않은 영화의 예측 평점: prediction 객체 생성
predictions = []
for tmdbId in unseen_movies:
predictions.append(algo.predict(userId, tmdbId))

# 리스트 내의prediction 객체의est를 기준으로 내림차순 정렬
def sortkey_est(pred):
return pred.est

predictions.sort(key=sortkey_est, reverse=True)  # key에 리스트 내 객체의 정렬 기준을 입력

# 상위top_n개의prediction 객체
top_predictions = predictions[:top_n]

# 영화 아이디, 제목, 예측 평점 출력
print(f"Top-{top_n} 추천 영화 리스트")

for pred in top_predictions:
tmdb_id = int(pred.iid)
movie_title = movies[movies["tmdbId"] == tmdb_id]["title"].tolist()[0]
movie_rating = pred.est

print(f"{tmdb_id}- {movie_title}: {movie_rating:.2f}")

db_cid = 'm_' + str(tmdb_id)
content = Contents.objects.filter(id=db_cid).values()
print("count : " + str(content.count()))

if content.count() != 0:
for cont in content:
recommend_list.append({'recommend': cont, 'est_rating': movie_rating})

return recommend_list
```

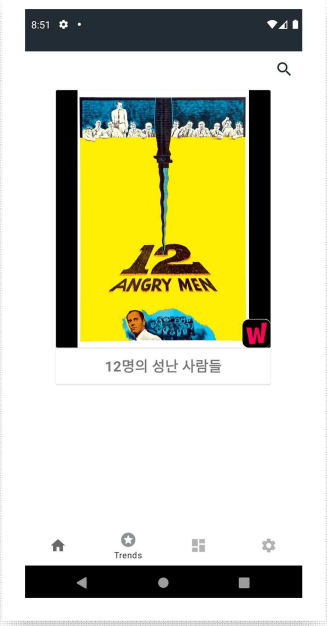
4. DEMO

1) 추천 컨텐츠 리스트(유저: lantern*******)


```
"recommend": {
    "id": "m_935",
    "_type": "movie",
    "title": "닥터 스트레인지러브",
    "genre": "드라마, 코미디, 전쟁, ",
    "production_countries": "United Kingdom, United States of America,
    "vote_count": 4366,
    "vote_average": 8.1,
    "number_of_seasons": null,
    "number_of_episodes": null,
    "release_date": "1964-01-29",
    "adult": 0,
    "poster_path": "/2VrsnlM63ab3oAuMt3XOnbGYMY6.jpg",
    "runtime": 95,
    "overview": "미 공군의 잭 리퍼 장군은 공산주의자들이 미국인의 신성한 혈통을
        오염시킬 음모를 꾸미고 있다는 망상에 사로잡혀 핵폭격기를 출격시킨다. 미국
        대통령은 절대절명의 위기를 해결 하기 위해 자문회를 소집하는데, 그 자리에서
        대사는 만일 소련이 핵공격을 당한다면 지구상의 모든 동식물이 파멸되는 운명
        날이 다가오게 될 것이라고 경고한다. 전 나치주의자였던 천재 과학자
        스트레인지러브 박사는 핵무기에 지구의 운명이 달려있다는 사실이 너무 명백하
        핵무기로 상황을 대응할 수 없다는 결론을 내린다. 과연 폭격기는 제 시간에
        수 있을 것인가? 아니면 잭 리퍼 장군이 전세계를 파멸시키는데 성공 할 것인
    "now_status": null,
    "flatrate": "",
    "rent": "Naver Store, ",
    "buy": "Naver Store, ",
    "link": "https://www.themoviedb.org/movie/
        935-dr-strangelove-or-how-i-learned-to-stop-worrying-and-love-
        b/watch?locale=KR"
},
"est_rating": 4.491379982160347
```

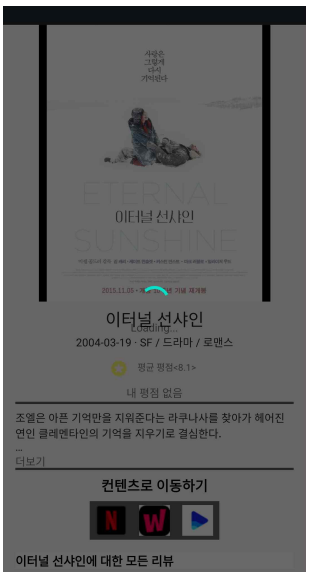




```
"recommend": {
    "id": "m_389",
    "_type": "movie",
    "title": "12명의 성난 사람들",
    "genre": "드라마, ",
    "production_countries": "United States of America, ",
    "vote_count": 6161,
    "vote_average": 8.5,
    "number_of_seasons": null,
    "number_of_episodes": null,
    "release_date": "1957-04-10",
    "adult": 0,
    "poster_path": "/ppd84D2i9W8jXmsyInGyihiSyqz.jpg",
    "runtime": 97,
    "overview": "뉴욕시의 법정에 아버지를 칼로 찌른 한 소년의 살인혐의를 두고
        배심원들은 만장일치 합의를 통해 소년의 유무죄 여부를 가려줄 것을 요구
        판사는 유죄일 경우 이 소년은 사형이 불가피하다는 것을 이들에게 미리 일
        배심원 방에 모인 이들은 투표를 통해 유무죄 여부를 가리기로 한다. 사람
        소년이 유죄로 판단하는 가운데, 오직 배심원 8(헨리 폰다)는 소년이 무죄
        주장하는데...",
    "now_status": null,
    "flatrate": "Watcha, ",
    "rent": "",
    "buy": "Naver Store, ",
    "link": "https://www.themoviedb.org/movie/389-12-angry-men/watch
        locale=KR"
},
"est_rating": 4.313645443058764
```

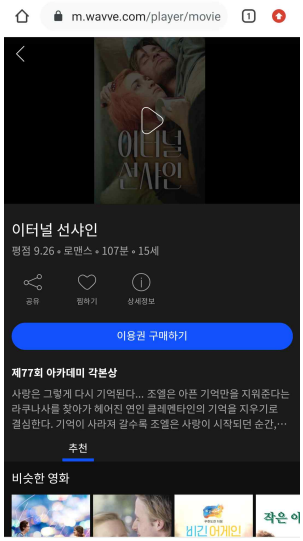



2) OTT 연동 링크

Contents Detail 화면



해당 컨텐츠 OTT URL 로딩 화면



로딩 완료 OTT 플랫폼