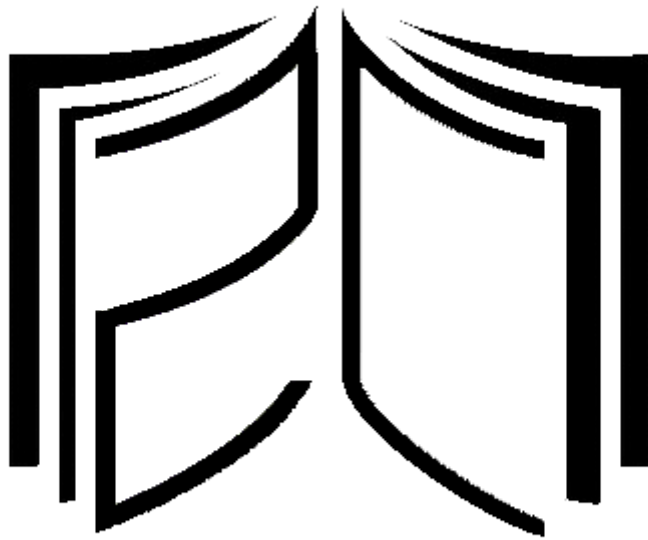


CAU 2019-02

Capstone Design

<READERS>



웹툰 통합 플랫폼

Webtoon Integration Platform
Including Recommendation System

<7조>

20165417 김소연

20163657 윤신영

20161856 한지효

목차

개요	3
동기 및 목표	3
개발 진행 일정	3
팀원 역할	4
Github	4
프로토타입	5
로고 디자인	5
플로우차트	5
UI 틀	7
주요 개발 내역	8
유저 인터페이스 (UI)	8
권한 및 인덱스 액티비티	8
로그인 및 회원가입 액티비티	8
메인 액티비티	9
홈	9
추천페이지	12
마이페이지	13
설정	15
작품 상세 페이지	17
안드로이드	17
서버와의 통신	17
권한 액티비티	18
인덱스 액티비티	18
로그인 액티비티	18
회원가입 액티비티	19
메인 액티비티	19
홈 프래그먼트	19
요일별	20
장르별	21
완결	21
연재처	22
검색	22
추천 프래그먼트	23
마이페이지 프래그먼트	23
구독	23
책갈피	23
메모	24
설정 프래그먼트	24

통합 로그인 관리	24
프로필 수정	25
숨김 작품 목록 관리	25
공지사항	26
문의	26
서버 (AWS)	26
데이터베이스	27
DB Schema	27
toon_info	28
toon_genre	28
toon_weekday	28
epi_info	28
user_info	29
user_genre	29
memo	29
subscribe	29
user_block	29
bookmark	29
연재처별 크롤링	29
데이터 분석	30
크롬 웹드라이버 Selenium	31
DB 실시간 업데이트	32
개선사항 및 확장성	33
유료 작품 결제	33
보기 방식	33
중복 작품	33
웹툰 추천 방식	33
결론	34
대상 및 범위	34
지원 연재처 수와 작품 종류	34
축적 DB 자원	34
차별화 기능	34
통합 로그인 기능	34
통합 정렬 및 검색 기능	35
유저 개인별 취향 분석 및 추천 기능	35
매뉴얼	35

1. 개요

1.1. 동기 및 목표

아고다(호텔)나 쿠팡(쇼핑), 스카이스캐너(비행기)처럼 웹툰도 한 곳에서 볼 수 있다면 어떨까.

한국콘텐츠진흥원에 따르면 만화산업 분야의 5년간 연평균 성장률은 5.6%이다. 2000년대 이후 10년이 넘는 기간동안 7000억원 대에 머물러 있었던 만화산업 매출이 현재 1.1조원에 이를 수 있었던 건 유료 웹툰 시장이 본격화 되면서부터다.

웹툰분석서비스(WAS)에 의하면 2018년 신규 등록된 웹툰 수는 전년도 대비 72편 증가했으나, 연재 중단 작품이나 조기 완결 작품은 436편 많아졌다. 투자 열기가 지속된 것에 비해 성과 판단 주기가 짧아졌고, 웹툰 불법 유통 사이트가 신작 효과를 편취했기 때문이다. 게다가 여러 플랫폼에서 다양한 웹툰이 쏟아지는 현실에, 대형 포털사이트와 같은 주류 연재처에 독자층이 쏠리는 일은 어쩌면 당연하다.




또한, 각기 다른 연재처에 있는 웹툰을 구독하기 위해선 해당하는 사이트에 접속하여 로그인 하거나 각 어플을 설치해야 하는 불편함이 존재한다.

따라서 증가하는 시장 규모에 맞추어 소비자가 좀 더 편리하게 서비스를 이용할 수 있도록 통합 프론트 페이지의 개발 필요성이 있다.

다양한 웹툰 서비스를 사용자가 한 곳에서 볼 수 있는 뷰어를 제공함으로써 그동안 독자의 발길이 쉽게 닿지 않았던 웹툰 플랫폼의 부흥까지 도모하는 것을 이 프로젝트의 목표로 한다.

1.2. 개발 진행 일정

READERS 프로젝트는 2019년 9월에 계획되어 10월 말 중간 데모를 거친 뒤 12월 초에 개발을 마무리지었다. 팀 전체 및 개별 상세 일정표는 아래 표와 같다.

	전체	9월				10월				11월				12월			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
김소연																	
윤신영																	
한지효																	
업무 주차																	
조 편성																	
제안서 제출																	
주제 보완																	
프로토타입(Axure RP) 제작																	
UI 디자인																	
사전 자료조사																	
AWS, 서버, DB 학습																	
서버 구축 및 수정																	
DB 설계 및 구축																	
웹툰 추천 알고리즘 구현																	
DB 크롤링																	
DB 업데이트																	
개인화 서비스 시스템 구축																	
추천 페이지 구현																	
웹툰 추천 알고리즘 설계																	
안드로이드 스튜디오 DB 연동																	
권한 화면, 시작 화면 구성																	
메인 화면 구성																	
로그인 화면 구성																	
웹툰 리스트 화면 구성																	
마이페이지 화면 구성																	
설정 화면 구성																	
통합 로그인 기능 구현																	
테스트																	

1.3. 팀원 역할

김소연	서버 및 DB 구축, 관리, 웹툰 정보 크롤링, 추천 알고리즘 구현
윤신영	안드로이드 개발
한지효	UI 프로토타입 제작, 추천 시스템 설계 및 화면 구성, DB 구축

1.4. Github

프로젝트에 이용된 모든 소스 코드와 문서 등은 상세 IP 주소 혹은 admin 비밀번호와 같이 기밀 정보에 해당하는 정보들을 제외하고 깃허브에 모두 공개되었다.

<https://github.com/I5ve/Readers>

2. 프로토타입

2.1. 로고 디자인

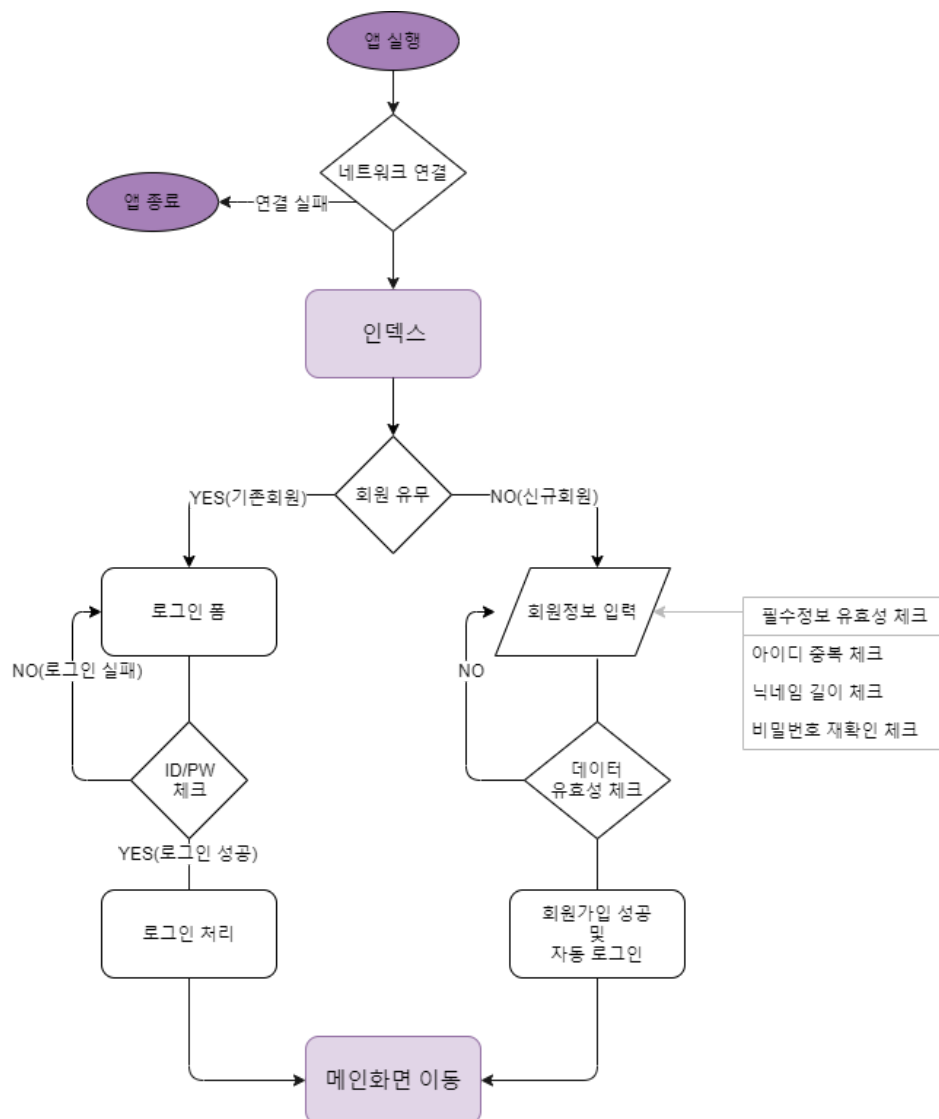


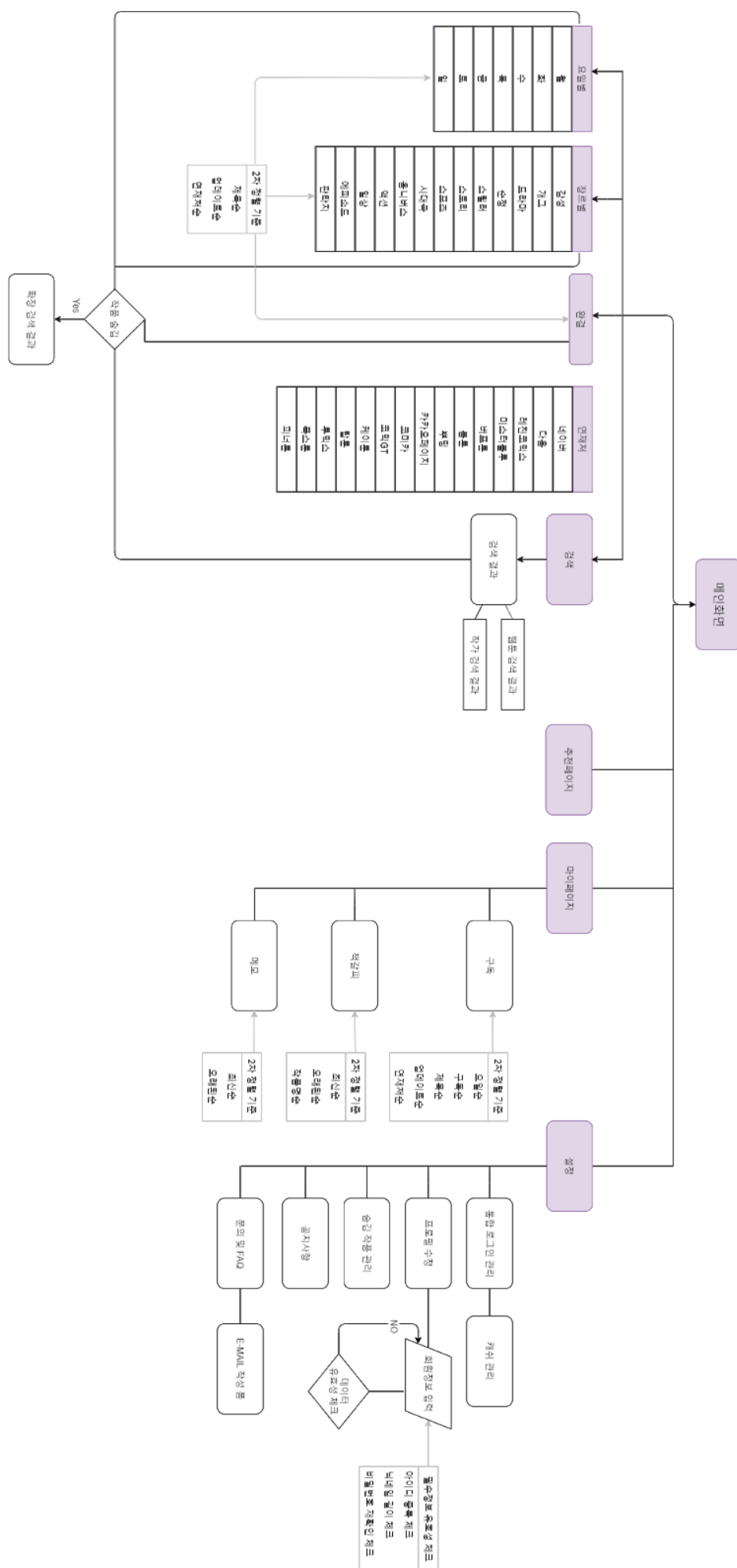
‘리더스’의 초성인 르, 드, 즈를 형상화하여 책 아이콘을 만듦으로써 서비스 이름인 READERS에 걸맞는 로고를 탄생시켰다. 해당 로고는 인덱스 화면은 물론 favicon 등과 같은 주요 아이콘에 활용되었다. 테마 색상 코드는 아래와 같다.

#281e42	#493e5f	#6a617c	#8e869c	#b2adbc	#d8d5dd	##### / #fff
---------	---------	---------	---------	---------	---------	--------------

2.2. 플로우차트

서비스의 흐름을 정의하기 위해 플로우차트를 작성하였다.





2.3. UI 툴

핵심 기능의 설계와 세부 UI, 그리고 서버와 안드로이드 간의 통신 주체 파악을 위해 프로토타입을 제작하였으며, 이때 사용된 툴은 Axure RP 9이다. 대략적인 인터랙션이 적용된 초기의 디자인은 깃허브의 design 폴더 내 readers_prototype.rp 파일에서 확인할 수 있으며, <https://xleegk.axshare.com>로 접속하면 온라인에서도 볼 수 있다. 제작된 페이지 구성은 다음과 같다.

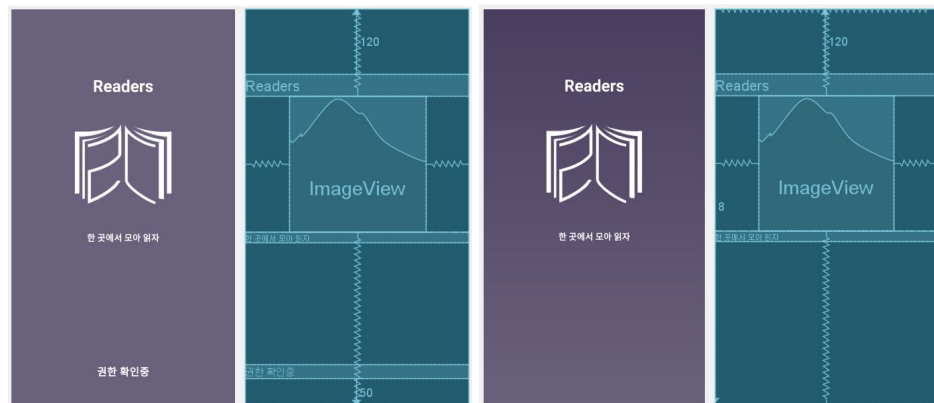
- ▼ ☰ Index
 - ☰ 로그인
 - ☰ 회원가입/프로필 수정
- ▼ ☰ 홈(요일별 웹툰)
 - ☰ 장르별
 - ☰ 완결
 - ☰ 연재처
 - ☰ 작품 상세 페이지
 - ☰ 검색
 - ☰ 추천
- ▼ ☰ 마이페이지
 - ☰ 책갈피
 - ☰ 메모
- ▼ ☰ 설정
 - ☰ 통합 로그인 관리
 - ☰ 숨김 작품 관리
 - ☰ 공지사항
 - ☰ 문의 및 FAQ

3. 주요 개발 내역

3.1. 유저 인터페이스 (UI)

대부분의 유저 인터페이스는 프로토타입을 참고하여 제작하였다. 여기서는 주요한 액티비티 및 프래그먼트에 대해서 설명하였고, 해당 기능과 관련이 있는 모든 xml파일을 하단부에 회색으로 표시하였다.

3.1.1. 권한 및 인덱스 액티비티



권한 액티비티와 인덱스 액티비티는 상단에 App의 이름과 로고를 넣고, 하단에 안내 메시지를 넣었다. RelativeLayout을 사용하여 제작하였다.

activity_permission.xml, activity_index.xml

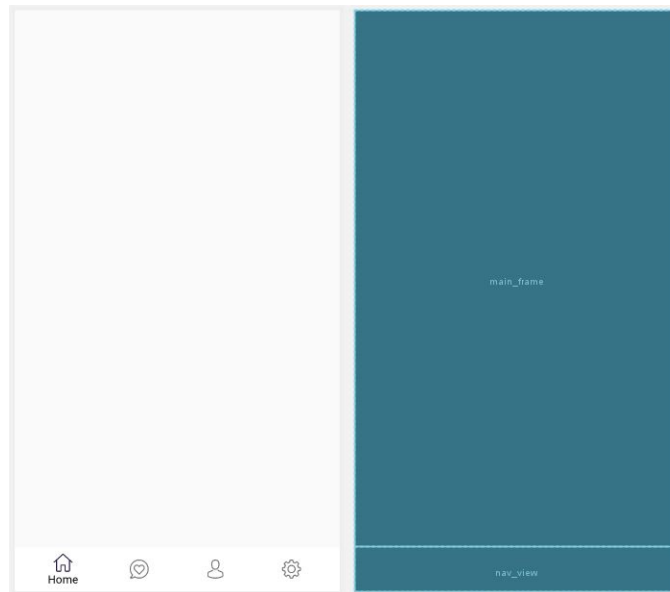
3.1.2. 로그인 및 회원가입 액티비티



로그인 액티비티와 회원가입 액티비티는 하단에서부터 위치하는 View가 없기 때문에 LinearLayout을 이용하여 제작하였다. 회원가입 액티비티의 경우는 화면 중간에 장르 리스트를 RecyclerView를 이용하여 버튼으로 출력하기 위해서 RecyclerView를 위치시켰다.

activity_login.xml, activity_signin.xml, recyclerview_taste.xml

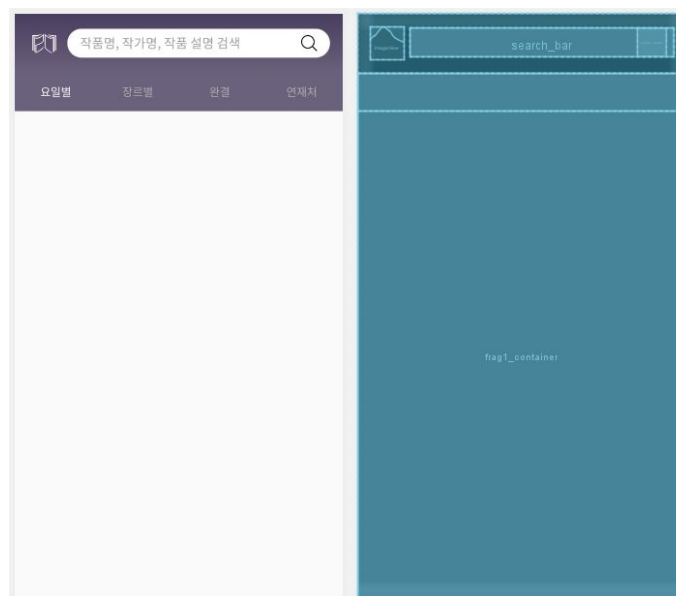
3.1.3. 메인 액티비티



메인 액티비티는 CoordinatorLayout을 틀로 하여 제작하였다. BottomNavigationView를 하단에 위치시키고, 각 메뉴 페이지를 보여줄 fragment를 출력할 FrameLayout을 BottomNavigationView의 세로 길이만큼 하단에 여백을 두고 위치시켰다.

activity_main.xml

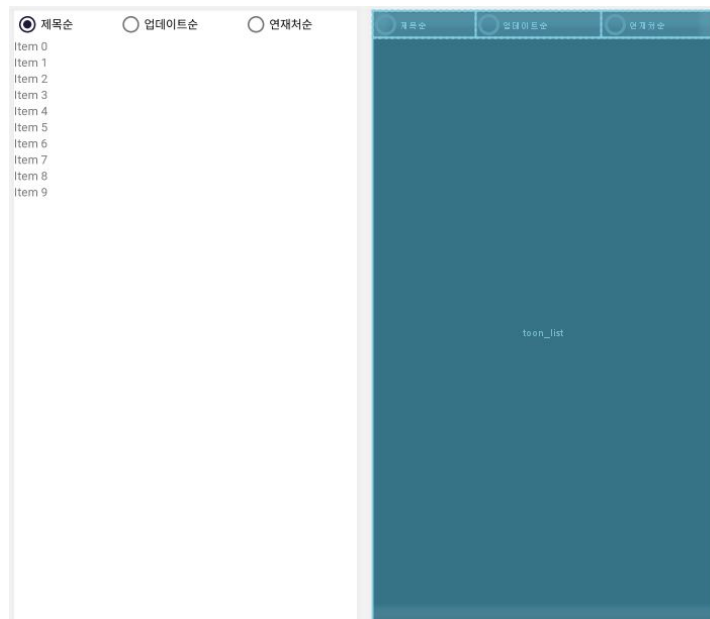
3.1.4. 홈



홈화면은 LinearLayout으로 제작하였다. 상단에 Horizontal LinearLayout안에 App의 로고 ImageView가 있고, 그 오른쪽에

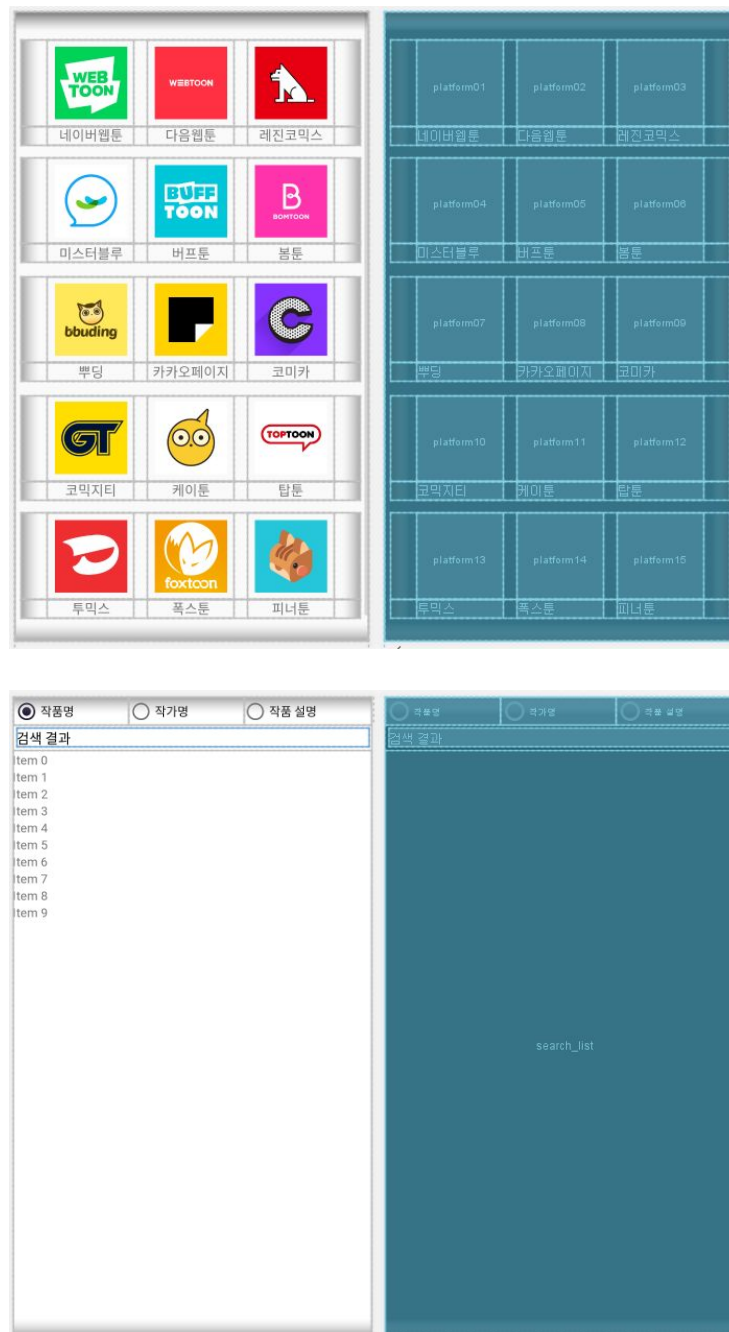
ConstraintLayout을 넣었다. ConstraintLayout 안에는 검색할 문자열을 입력받을 EditText와 검색 아이콘을 넣을 ImageButton을 넣었다. 하단에는 웹툰 클릭 시 작품 상세 페이지 fragment를 담기 위한 id가 frag1_big_container인 FrameLayout을 위치시키고, 그 안에 요일별, 장르별, 완결, 연재처 메뉴가 있는 탭을 위치시켰다. 탭 하단부에는 웹툰 리스트를 출력하기 위한 id가 frag1_container인 FrameLayout을 위치시켰다.

장르별, 요일별, 완결 탭 클릭 시 frag1_container FrameLayout에 fragment1_menu1.xml, fragment2_menu1.xml, fragment3_menu1.xml 파일이 출력된다. 각 파일의 상단부에는 해당 탭에 알맞은 탭이 들어있다. 즉, 요일별 탭의 경우는 월요일부터 일요일까지의 탭이 있고, 장르별 탭의 경우는 13개의 장르탭이 들어있다. 각 파일에는 FrameLayout이 들어있다. 해당 FrameLayout에는 fragment_toon.xml 파일이 출력된다.



fragment_toon.xml | 안에는 제목순, 업데이트순, 연재처순 라디오버튼 그룹과, 웹툰 리스트를 출력하기 위한 RecyclerView가 위치한다.

연재처 탭 클릭 시에는 fragment4_menu1.xml 파일이 출력된다. 해당 파일 안에는 15개의 ImageButton이 들어있다.



특정 키워드를 입력하고 검색 버튼을 누르면 fragment_search.xml 파일을 통해 검색 결과가 출력된다. fragment_search.xml에는 작품명, 작가명, 작품 설명으로 검색을 하는 라디오버튼 그룹이 있고 그 밑에는 검색 결과 리스트를 출력하는 RecyclerView가 있다.



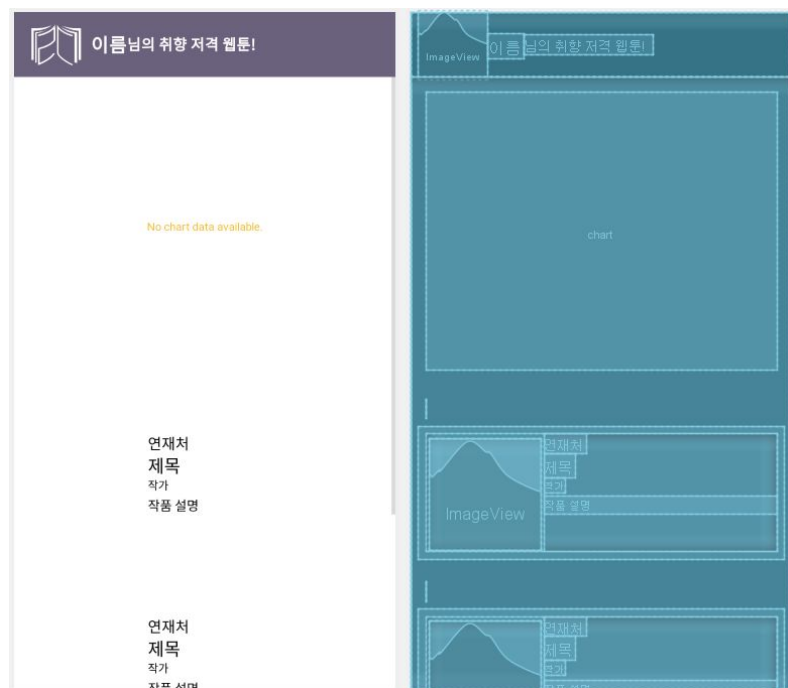
작품명 또는 작가명으로 검색시 검색 결과는 다음과 같이 출력된다. LinearLayout으로 제작하였다.



작품 설명으로 검색 시 검색 결과는 다음과 같이 출력된다. LinearLayout으로 제작하였다.

fragment_menu1.xml, fragment1_menu1.xml, fragment2_menu1.xml, fragment3_menu1.xml, fragment4_menu1.xml, fragment_toon.xml, recyclerview_toon.xml, fragment_search.xml, recyclerview_search.xml, recyclerview_descsearch.xml

3.1.5. 추천페이지

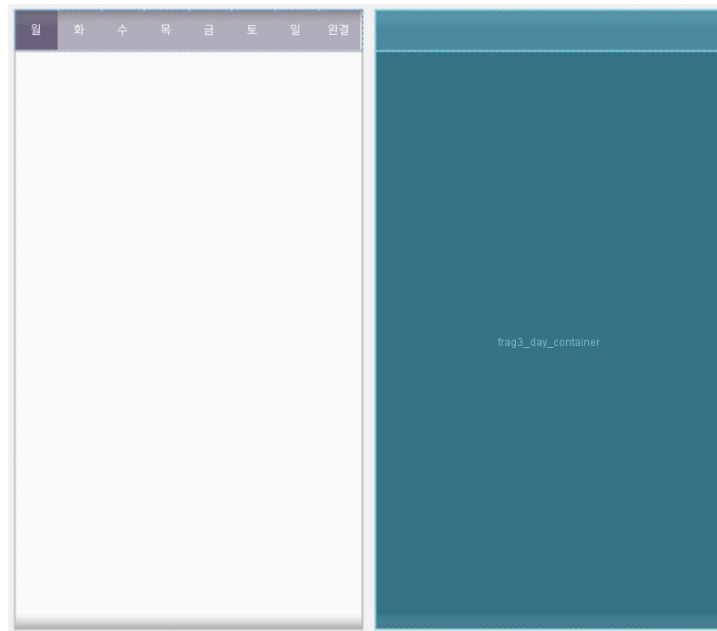


추천페이지의 경우 LinearLayout을 이용해 틀을 제작하였다. 상단에는 유저의 닉네임을 위한 추천 문구가 명시되어 있으며, 아래로는 MPAndroidChart를 이용한 그래프 정보가 존재한다. 여기서는 추천 웹툰의 개수가 상위 3 장르에 대해서 각각 한개씩으로 고정되어 있기 때문에 RecyclerView를 사용하지 않았다. 상단바 하단부는 추천 작품 클릭 시 상세페이지를 보여주기 위해 FrameLayout으로 제작하였다.

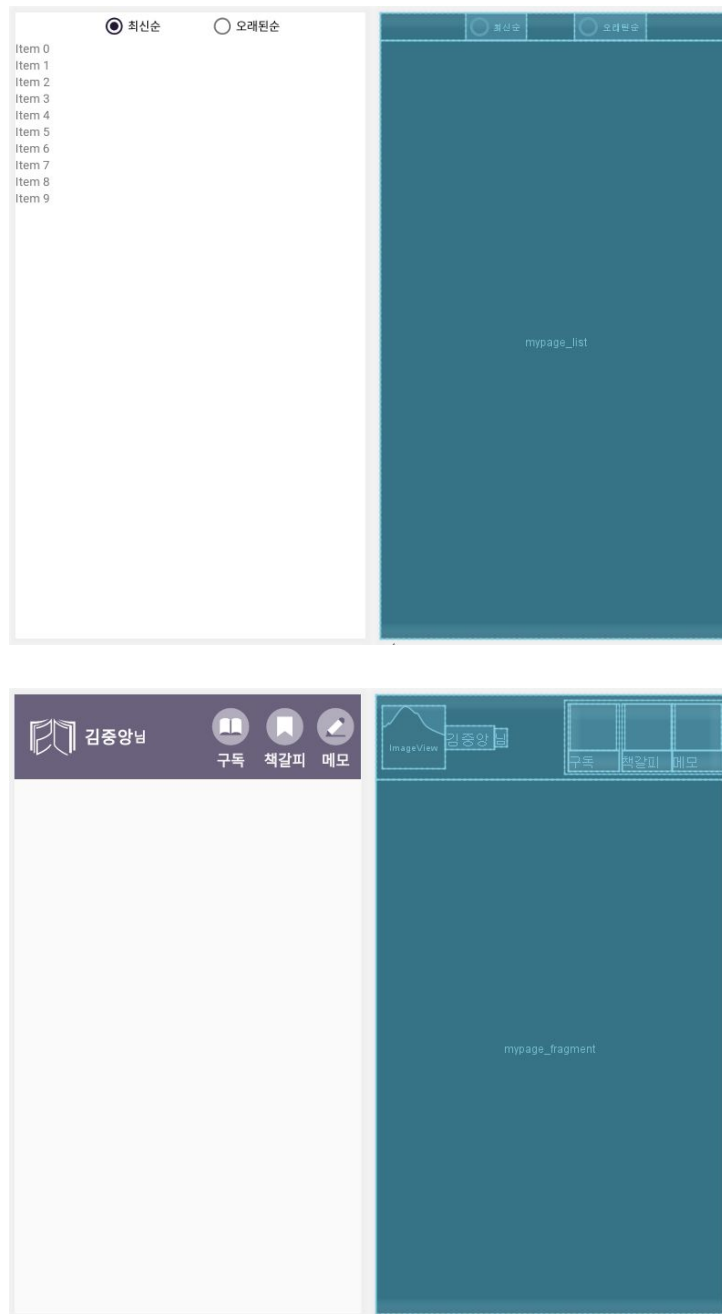
fragment_menu2.xml

3.1.6. 마이페이지

마이페이지의 경우 LinearLayout으로 제작하였다. 상단바 부분은 ConstraintLayout으로 제작하여 App 로고 및 사용자 닉네임은 화면의 좌측에, 구독, 책갈피, 메모 버튼은 화면의 우측에 정렬되도록 하였다. 상단바 밑부분은 각각 구독, 책갈피, 메모에 맞는 fragment를 출력하기 위해 FrameLayout을 넣었다.



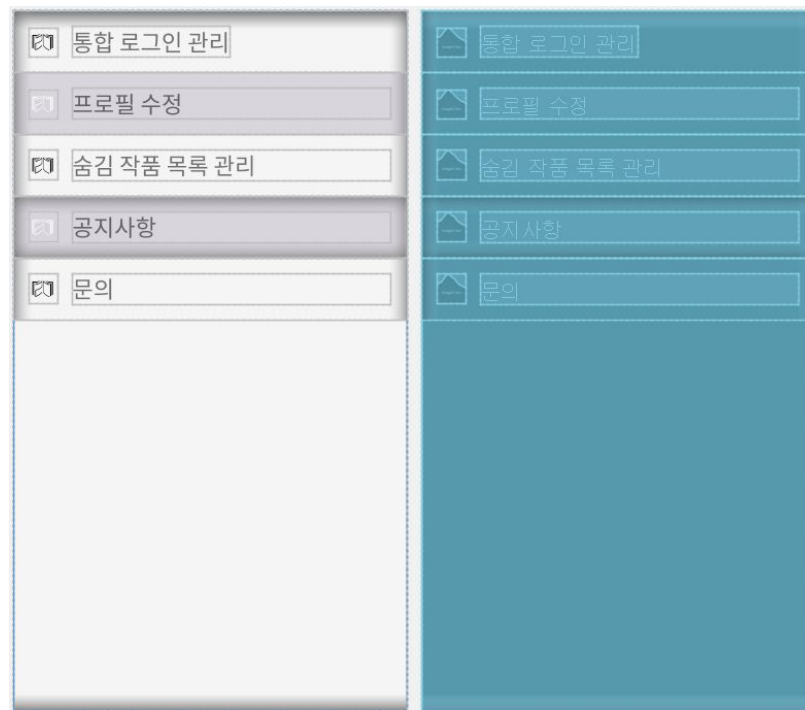
마이페이지 초기화면이나 구독 버튼 클릭 시 fragment1_menu3.xml 파일이 출력된다. fragment1_menu3.xml 파일에는 월요일부터 일요일, 그리고 완결 메뉴가 있는 탭이 있고 그 밑에 요일별 구독 웹툰 리스트를 출력할 FrameLayout이 있다.



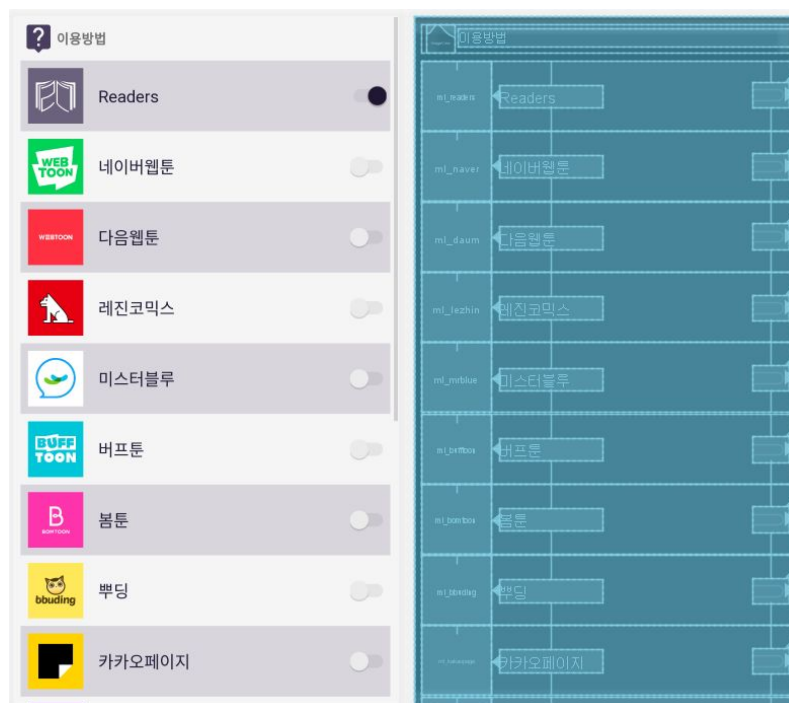
책갈피와 메모 버튼 클릭 시, fragment_mypage.xml 파일이 출력된다. 해당 파일 상단부에는 최신순, 오래된순으로 정렬을 선택하는 라디오버튼 그룹이 위치해있고, 그 밑에는 책갈피와 메모 리스트를 출력할 부분인 FrameLayout이 있다.

fragment_menu3.xml, fragment1_menu3.xml, fragment_mypage.xml,
fragment_toon.xml, recyclerview_toon.xml,
recyclerview_bookmark.xml, recyclerview_memo.xml

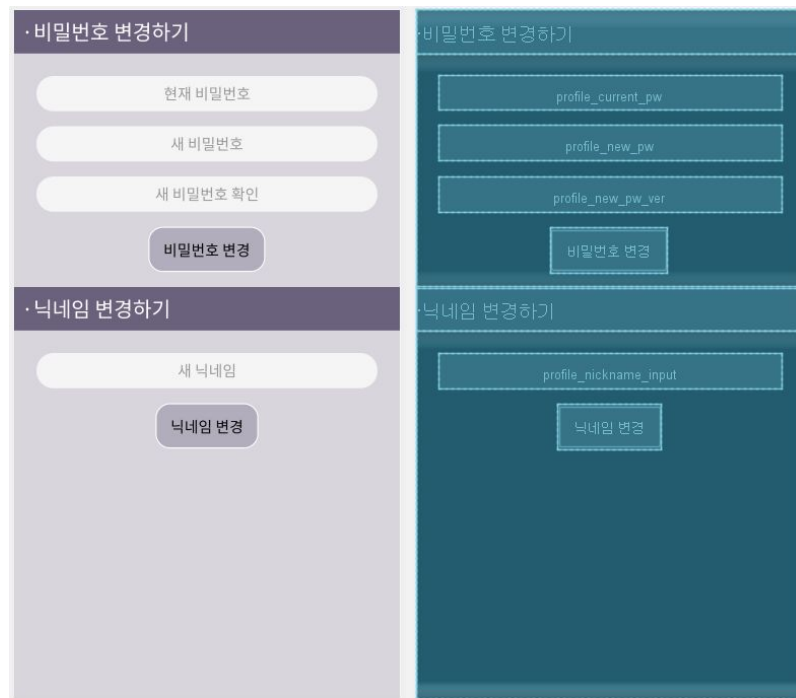
3.1.7. 설정



설정페이지의 경우 LinearLayout으로 제작하였다.



통합 로그인 관리 페이지 메뉴를 선택하면 fragment_managelogin.xml 파일이 출력된다. 해당 파일은 LinearLayout으로 제작하였다. 스크롤 뷰 안에 있는 각 행은 RelativeLayout으로 제작하였다.



두번째 메뉴인 프로필 수정을 클릭하면 fragment_profile.xml 파일이 출력된다. 해당 파일은 LinearLayout으로 제작하였다.

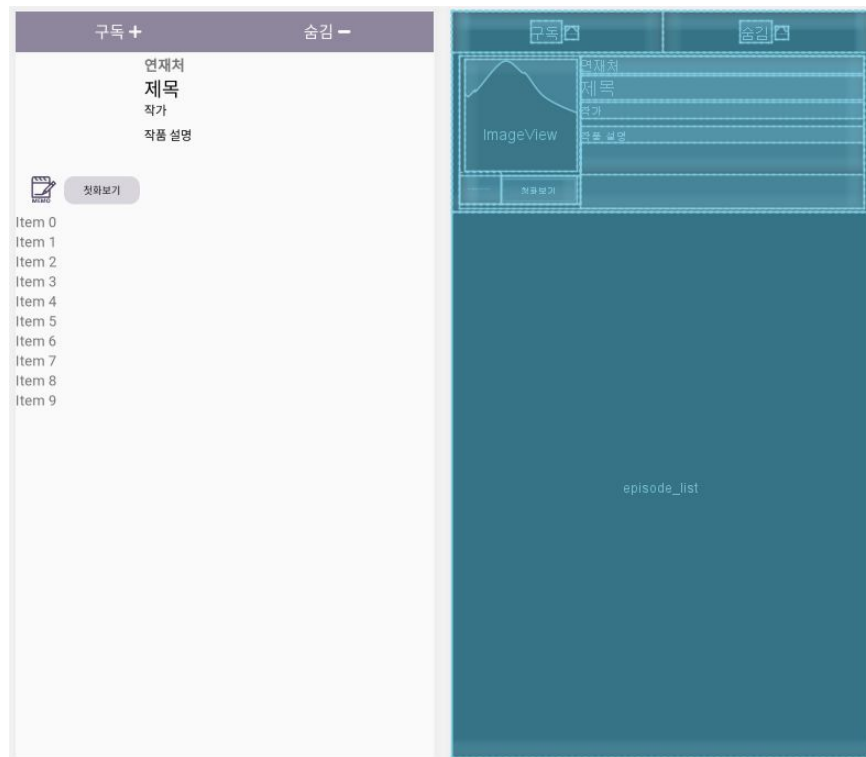
세번째 메뉴인 숨김 작품 목록 관리를 클릭하면 fragment_manageblock.xml 파일이 출력된다. 해당 파일에는 숨김 웹툰 리스트를 출력하기 위한 RecyclerView를 넣었다.



RecyclerView에 출력되는 숨김 웹툰의 형태는 다음과 같으며, CardView를 이용해 제작하였다.

fragment_menu4.xml, fragment_managelogin.xml,
fragment_profile.xml, fragment_manageblock.xml,
recyclerview_block.xml

3.1.8. 작품 상세 페이지



작품 상세 페이지의 경우 LinearLayout으로 제작하였다. 상단부에 구독 버튼과 숨김 버튼이 있고, 그 밑에 작품에 대한 내용이 출력된다. 하단부에는 작품의 에피소드를 출력하는 RecyclerView를 넣었다.



작품 에피소드는 다음과 같은 형태로 출력되며, CardView로 제작하였다.

fragment_detail_page.xml, recyclerview_episode.xml

3.2. 안드로이드

3.2.1. 서버와의 통신

서버와의 통신은 Retrofit 라이브러리와 ServiceApi 인터페이스를 제작하여 사용하였다. 통신 방식은 모두 POST 방식을 사용하였다.

RetrofitClient.java, ServiceApi.java

3.2.2. 권한 액티비티

시작 액티비티이며, 권한을 확인하고 권한이 부여되어 있지 않다면 권한을 요청한다. 권한 처리 결과를 보고 인덱스 액티비티를 실행할지, 권한 설정 요청 다이얼로그를 보여줄지 결정한다. 모든 권한이 승인되었을 경우에는 인덱스 액티비티를 실행하고 현재 액티비티를 종료한다.

PermissionActivity.java

3.2.3. 인덱스 액티비티

가장 먼저 RemoteLib 라이브러리를 이용해 기기가 인터넷에 연결되어 있는지를 확인한다. 인터넷에 연결되어있지 않으면 showNoService() 메소드를 호출하여 현재 인터넷에 접속할 수 업식 때문에 서비스를 사용할 수 없다는 메시지와 함께 화면 종료 버튼을 보여준다.

인터넷에 연결되어 있다면, 기존에 저장되어 있던 설정값을 불러온다. 저장된 로그인 정보가 있다면 메인 액티비티로, 없다면 로그인 액티비티로 이동한다. 이때, 핸들러를 이용하여 인덱스 액티비티는 0.5초동안 보여지게 된다.

IndexActivity.java, RemoteLib.java, MyApp.java

3.2.4. 로그인 액티비티

로그인 액티비티에서는 입력받은 ID와 비밀번호의 유효성 검사를 한다. ID와 비밀번호가 모두 6자 이상 입력되었을 경우에는 입력받은 비밀번호를 SHA256 알고리즘을 이용하여 암호화한다. 암호화한 비밀번호와 아이디를 서버로 전송한다. 서버에서 200이라는 코드를 돌려받았을 시에는 메인 액티비티를 시작하면서 로그인 액티비티를 종료한다. 이때, 로그인 정보 저장 여부를 검사하고 저장하기에 체크가 되어있을 때에는 전역 클래스를 통해 로그인 정보를 저장한다. 아이디가 없거나 비밀번호가 틀린 경우에는 각 경우에 맞는 에러메시지를 토스트 메시지로 띄운다.

회원가입 버튼을 클릭했을 시에는 startActivityForResult 메소드를 이용하여 회원가입 액티비티를 시작한다. 이후 REQUEST_CODE_SIGNIN에 해당하는 값을 request code로 돌려받았을 시에는 메인 액티비티를 시작하고 로그인 액티비티를 종료한다.

LoginActivity.java, MyApp.java

3.2.5. 회원가입 액티비티

회원가입 액티비티의 경우 13개의 장르를 RecyclerView를 이용하여 버튼으로 출력한다. 사용자가 화살표 버튼을 클릭하면 입력받은 비밀번호, 아이디, 닉네임의 유효성을 검사하고 이상이 없다면 사용자가 선택한 선호 장르가 3개 이상인지 검사한다. 3개 이상이라면 먼저 사용자가 입력한 아이디와 닉네임, 비밀번호를 서버에 전송한다.

서버에서 코드 200을 돌려받으면 전역 클래스에 아이디와 닉네임, 암호화된 비밀번호를 저장한다. 회원가입 시에는 로그인 정보를 유지하는 것으로 판단하여 전역 클래스에 로그인 정보를 유지한다고 저장한다. 그리고 사용자의 아이디와 사용자가 고른 선호 장르를 서버에 전송한다.

이후, 인텐트를 생성하여 RESULT_OK값을 result로 설정한다. 그리고 회원가입 액티비티를 종료한다

SignInActivity.java

3.2.6. 메인 액티비티

App의 핵심 액티비티로, 아래쪽에 navigation menu를 가진다. 다양한 프래그먼트를 보여주는 컨테이너 역할을 한다. BottomNavigationView를 통해 입력을 받을 때마다 FrameLayout에 각 메뉴의 fragment를 바꾼다. 기본 화면으로는 홈화면을 출력하고 1, 2, 3, 4번 메뉴가 클릭됐을 때에는 각각 홈, 추천, 마이페이지, 설정 fragment를 보여준다.

MainActivity.java

3.2.7. 홈 프래그먼트

메인 액티비티에서 첫번째 메뉴를 눌렀을 때 보이는 화면이다. 요일별, 장르별, 완결, 연재처 총 4개의 탭을 가지고 있으며, 각 탭이 클릭되었을 때 id가 frag1_container인 FrameLayout에 Menu1Fragment1.java, Menu1Fragment2.java, Menu1Fragment3.java, Menu1Fragment4.java를 출력한다.

특정 검색어를 입력하고 엔터키를 누르거나 검색 아이콘을 클릭했을 때에는 id가 frag1_big_container인 FrameLayout에 SearchFragment.java를 출력한다.

3.2.7.1. 요일별

사용자가 특정 요일 탭을 클릭할 때마다 해당 요일을 전역 클래스에 저장한다. 그리고 ToonFragment.java를 id가 frag1_day_container인 FrameLayout에 출력한다.

ToonFragment.java에서는 RecyclerView에 표시할 데이터 리스트를 생성하고, 넘겨받은 정보가 요일, 장르, 완결 중 어느 것인지 전역 클래스를 통해 정보를 확인한다.

RecyclerView에 열이 3개인 GridLayout을 지정한다.

RecyclerView는 한 번에 12개의 웹툰을 출력하고, 마지막까지 스크롤되었을 때는 서버에서 받아온 웹툰 목록에서 12개의 웹툰을 더 출력하도록 한다. 이때, 목록에 남아있는 웹툰이 12개 보다 적을 경우에는 해당 개수만큼 출력을 한다.

제목순, 업데이트순, 연재처순 라디오버튼 클릭 시에는 정렬 기준을 바꾸어 서버에 요청을 보낸다.

ToonFragment.java에서는 getData() 메소드를 이용해서 서버에 정보를 전송하는데, 요일별, 장르별, 완결 각 상황에 알맞은 정보를 서버로 전송하여 웹툰 목록을 리스트 형태로 돌려받는다. 리스트 중 12개, 리스트 길이가 12보다 작다면 리스트의 길이만큼만 RecyclerView의 데이터셋에 넣어준다. 이후 ToonListAdapter.java를 RecyclerView의 adapter로 설정한다.

ToonListAdapter.java에서는 recyclerview_toon.xml에 해당하는 레이아웃을 설정한다. 스레드를 이용하여 인터넷에서 이미지를 비트맵 형태로 가져와 썸네일 이미지로 설정한다.

해당 웹툰이 클릭된 경우에는 전역 클래스에 웹툰의 정보를 저장하고 EpisodeFragment.java를 출력한다.

EpisodeFragment.java에서는 서버에 사용자 아이디와 해당 웹툰의 아이디를 서버로 전송하여 작품 설명, 메모, 사용자의 구독 여부, 숨김 여부에 대한 데이터를 받아온다. 해당 작품에 대한 장르 정보와 작품의 첫 에피소드 url도 받아온다. 그리고 마지막으로 사용자 아이디와 웹툰 아이디를 서버로 전송하여 해당 웹툰의 에피소드 리스트를 받아온다. 에피소드 리스트에는 해당 사용자의 책갈피 여부에 대한 정보도 담겨있다. 에피소드 리스트를 받아온 다음에는 RecyclerView의 Adapter를 EpisodeListAdapter.java로 설정한다.

EpisodeListAdapter.java에서는 recyclerview_episode.xml 레이아웃에 알맞게 설정을 한다. 썸네일의 경우 스레드를 사용하여 인터넷에서 이미지를 비트맵 형태로 받아와 설정한다. 사용자가 해당 에피소드를 클릭했을 경우, 전역 클래스에 해당 에피소드의 url을 저장하고, WebViewFragment를 이용해 해당 url에 WebView를 통하여 접속한다.

사용자가 해당 에피소드를 책갈피를 설정하거나 해제한 경우, getAdapterPosition() 메소드를 이용하여 해당 데이터의 인덱스를 받아온다. 받아온 인덱스를 통해 데이터셋에서 해당 에피소드의 정보를 찾고, 이 정보와 사용자의 아이디를 서버로 전송하여 책갈피를 설정하거나 해제한다.

다시 EpisodeFragment.java로 돌아와서, 사용자가 구독 버튼을 눌러서 구독을 설정하거나 해제하면, 사용자 아이디와 해당 웹툰의 아이디를 서버로 전송하여 구독을 설정 또는 해제한다. 이때, 마이페이지의 구독 화면에서 이 상세페이지로 넘어온 경우, 구독을 해제했다면 마이페이지의 구독 화면에서 해당 웹툰의 정보를 삭제한다. 구독을 해제했다가 다시 설정한 경우에는 해당 웹툰을 마이페이지의 구독 리스트에 다시 추가한다.

숨김의 경우도 마찬가지이다. 홈화면(요일별, 장르별, 완결)에서 해당 상세 페이지로 넘어온 경우, 숨김 시 해당 웹툰을 홈화면의 웹툰 리스트에서 삭제한다. 이때, 숨김을 설정했다가 해제할 시에는 해당 웹툰을 다시 웹툰 리스트에 추가한다.

메모를 입력하고 저장 버튼을 누를 시에는 사용자의 아이디와 해당 웹툰의 정보, 메모 내용을 서버에 전송한다. 이때, 마이페이지의 메모 화면에서 상세페이지로 넘어온 경우 메모를 저장했다는 것은 메모를 수정했다는 것을 의미하므로 마이페이지의 메모 리스트에서 해당 웹툰에 대한 메모 내용을 업데이트한다. 메모를 삭제한 경우도 마찬가지이다.

3.2.7.2. 장르별

사용자가 특정 장르 탭을 클릭할 때마다 해당 장르를 전역 클래스에 저장한다. 그리고 ToonFragment.java를 id가 frag1_genre_container인 FrameLayout에 출력한다. 이후의 과정은 1. 요일별과 동일하다.

3.2.7.3. 완결

사용자가 완결 탭을 클릭했다는 정보를 전역 클래스에 저장한다. 그리고 ToonFragment.java를 id가

frag1_end_container인 FrameLayout에 출력한다. 이후의 과정은 1. 요일별과 동일하다.

3.2.7.4. 연재처

사용자가 연재처 탭을 클릭 하면 15개의 이미지 버튼이 출력된다. 버튼을 클릭하는 경우전역 클래스에 해당 연재처 사이트의 url을 저장하고, WebViewFragment를 이용해 해당 url에 WebView를 통하여 접속한다.

3.2.7.5. 검색

사용자가 특정 키워드를 검색하면 해당 키워드를 전역 클래스에 저장하고 SearchFragment.java를 출력한다. RecyclerView에 LinearLayoutManager를 설정한다. 작품명, 작가명, 작품 설명 라디오버튼을 클릭하는 경우 현재 존재하는 데이터셋을 삭제하고 서버에 상황에 맞는 데이터를 요청한다.

작품명, 작가명으로 검색 시에는 검색 키워드를 서버에 요청하여 작품 설명이 포함되어 있지 않은 검색 결과 리스트를 돌려받는다. 그리고 SearchListAdapter를 RecyclerView의 adapter로 설정한다.

SearchListAdapter에서는 recyclerview_search.xml에 해당하는 레이아웃을 설정하고, 특정 검색 결과를 클릭할 시 전역 클래스에 해당 웹툰 정보를 저장한 다음 EpisodeFragment.java를 출력하여 작품 상세 페이지로 이동한다.

작품 설명으로 검색 시에는 검색 키워드를 서버에 요청하여 작품 설명이 포함되어 있는 검색 결과 리스트를 돌려받는다. 그리고 DescSearchListAdapter를 RecyclerView의 adapter로 설정한다. DescSearchListAdapter는 SearchListAdapter와 달리 recyclerview_descsearch.xml에 해당하는 레이아웃, 즉, 작품 설명에 대한 레이아웃도 포함되어 있다는 점이 다르다. 이외의 부분은 동일하다.

Menu1Fragment.java, Menu1Fragment1.java, Menu1Fragment2.java, Menu1Fragment3.java, Menu1Fragment4.java, ToonCard.java, ToonFragment.java, ToonListAdapter.java, SearchCard.java, DescSearchCard.java, SearchFragment.java, SearchListAdapter.java, DescSearchListAdapter.java, EpisodeCard.java, EpisodeFragment.java, EpisodeListAdapter.java

3.2.8. 추천 프래그먼트

추천 페이지의 경우, 사용자의 취향 분석을 시각적으로 제공하기 위해 MPAndroidChart 라이브러리 중 HorizontalBarChart를 이용하였다. 사용자가 App을 사용하면서 수집한 활동 정보를 바탕으로 기록된 모든 장르 정보가 가중치에 따라 차트 형식으로 표시된다.

이를 위해 먼저 사용자의 아이디를 서버로 전송하여 장르에 대한 가중치를 받아온 후 전달받은 count 값을 토대로 그래프를 설정한다.

또한, 사용자의 아이디를 서버로 전송하여 사용자의 상위 3 장르에 대해서 1개씩 추천 작품을 받아온다. 이때 작품의 개수는 총 3개로 고정되어 있으므로 RecyclerView를 사용하지 않았다.

추천 작품 정보를 레이아웃에 설정해주고, 해당 작품을 클릭했을 시에는 전역 클래스에 해당 작품의 정보를 저장하고 EpisodeFragment.java를 통해 작품 상세 페이지로 이동하도록 설정하였다.

Menu2Fragment.java, EpisodeCard.java, EpisodeFragment.java, EpisdoeListAdapter.java, WebviewFragment.java

3.2.9. 마이페이지 프래그먼트

마이페이지에서는 앱 내에 저장해둔 사용자 닉네임을 불러와 화면에 표시한다. 그리고 구독, 책갈피, 메모 버튼이 클릭되면 각각

Menu3Fragment1.java, BookmarkFragment.java,

MemoFragment.java를 출력한다. 아무것도 클릭되지 않았을 경우에는 기본 화면으로 사용자의 구독 화면을 출력한다.

3.2.9.1. 구독

fragment1_menu3.xml 파일을 출력한다. 월요일부터 일요일, 그리고 완결 탭 중 특정 탭을 사용자가 클릭하면 해당 탭에 대한 정보를 전역 클래스에 저장하고 id가 frag3_day_container인 FrameLayout에 MypageSubscribeFragment.java를 출력한다.

MypageSubscribeFragment.java와

MypageSubscribeListAdapter.java는 ToonFragment.java와 ToonListAdapter.java와 동일한 방식으로 동작한다.

3.2.9.2. 책갈피

BookmarkFragment.java에서는 레이아웃 파일로 fragment_mypage.xml 파일을 사용한다. 최신순, 오래된순의 라디오 버튼 그룹이 있고, 책갈피 리스트를 출력하기 위한

RecyclerView가 있다. RecyclerView에 LinearLayoutManager를 설정한다. 이후 서버에 사용자 아이디와 정렬 기준을 전송하여 책갈피 리스트를 돌려받고, BookmarkListAdapter.java를 adapter로 설정한다.

BookmarkListAdapter.java에서는 recyclerview_bookmark.xml에 해당하는 레이아웃을 설정한다. 그리고 책갈피의 오른쪽에 있는 X 버튼이 클릭될 경우 getAdapterPosition() 메소드를 이용하여 데이터셋에서의 해당 웹툰의 인덱스를 받아온다. 받아온 인덱스를 이용하여 데이터셋에서 해당 책갈피의 정보를 찾는다. 찾은 책갈피 정보와 사용자의 아이디를 서버에 전송하여 북마크를 삭제한다. 코드 200을 돌려받을 경우 데이터셋에서 해당 북마크 정보를 삭제하고 notifyItemRemoved() 메소드를 이용해서 해당 책갈피가 삭제되었음을 adapter에 알린다.

책갈피가 클릭되었을 경우 해당 에피소드 url을 전역 클래스에 저장하고 WebViewFragment.java를 통해서 해당 주소로 접속한다.

3.2.9.3. 메모

MemoFragment.java와 MemoListAdapter.java는 BookmarkFragment.java와 BookmarkListAdapter.java와 유사한 방식으로 동작한다.

다만, MemoListAdapter의 경우 해당 메모가 클릭되었을 때에 해당 웹툰의 상세페이지로 이동한다.

Menu3Fragment.java, Menu3Fragment1.java,
MypageSubscribeFragment.java, MypageSubscribeListAdapter.java,
BookmarkCard.java, BookmarkFragment.java,
BookmarkListAdapter.java, WebViewFragment.java, MemoCard.java,
MemoFragment.java, MemoListAdapter.java

3.2.10. 설정 프래그먼트

Menu4Fragment.java에는 5가지 기능이 있다. 통합 로그인 관리, 프로필 수정, 숨김 작품 목록 관리, 공지사항, 문의 기능이다.

3.2.10.1. 통합 로그인 관리

통합 로그인 관리 버튼을 클릭할 경우 ManageLoginFragment.java 파일이 id가 menu4_frame인 FrameLayout에 출력된다.

Readers의 스위치 버튼을 클릭하면 LoginActivity class를 담은 인텐트를 만들고 FLAG_ACTIVITY_CLEAR_TOP을 인텐트 플래그로 설정한다. 그리고 전역 클래스에 있는 initialize() 메소드를 호출하여 15개 연재처에 로그인한 내역을 모두 삭제하고 로그인 유지 정보도 삭제한다. 그리고 로그인 액티비티를 실행한 후 메인 액티비티를 종료한다.

15개 연재처의 경우, 꺼져 있는 스위치 버튼을 클릭했을 때에는 LoginWebViewActivity.java를 통해 해당 연재처로 접속한다. 켜져 있는 스위치 버튼을 클릭했을 때에는 LoginWebViewActivity.java를 통해 해당 연재처에서 로그아웃을 할 수 있는 링크로 접속한다.

이때, url과 연재처 이름, 화면 종료 여부, 자동 로그아웃 여부를 인텐트 정보에 담아서 전송한다. 로그아웃을 할 수 있는 url 링크를 제공하는 연재처의 경우 WebView를 통해 로그아웃 링크로 접속하여 로그아웃을 실행하고 난 후 바로 창을 닫으면 되므로 화면 종료 여부 값을 true로 전달하면 사용자가 직접 창을 닫지 않아도 된다. 이때 곧바로 로그아웃할 수 있는 링크를 제공하지 않는 연재처의 경우 화면 종료 여부와 자동 로그아웃 여부를 false 값으로 전달하여 사용자에게 직접 로그아웃을 하라고 토스트 메시지로 안내한다.

3.2.10.2. 프로필 수정

프로필 수정 버튼을 클릭할 경우 ProfileFragment.java 파일이 id가 menu4_frame인 FrameLayout에 출력된다.

비밀번호 변경의 경우 로그인과 회원가입 시에 했던 것처럼 현재 비밀번호가 일치하는지 확인하고 새 비밀번호의 유효성을 검사한다. 알맞은 비밀번호일 경우 사용자 아이디와 암호화된 새 비밀번호를 서버에 전송하여 비밀번호를 변경한다.

닉네임의 경우도 새 닉네임을 입력받아 유효성 검사를 실시한 후 알맞은 닉네임이라고 판단이 되면 사용자 아이디와 닉네임을 서버로 보내 닉네임을 변경한다.

3.2.10.3. 숨김 작품 목록 관리

숨김 작품 목록 관리의 경우 BlockFragment.java 파일이 id가 menu4_frame인 FrameLayout에 출력된다.

BlockFragment.java는 RecyclerView를 포함한 fragment_manageblock.xml 파일을 레이아웃 파일로 설정한다. 열이 3개인 GridLayoutManager를 RecyclerView의 LayoutManager로 설정한다. 그리고 서버에 사용자 아이디를

전송하여 사용자가 숨김 설정한 웹툰 리스트를 돌려받는다. 그리고 BlockListAdapter.java를 RecyclerView의 adapter로 설정한다.

BlockListAdapter.java는 recyclerview_block.xml에 해당하는 레이아웃을 설정한다. 사용자가 숨김 해제 버튼을 클릭한 경우 getAdapterPosition() 메소드를 이용하여 현재 데이터셋에서 해당 웹툰의 인덱스를 알아낸다. 인덱스를 통해 데이터셋에서 웹툰의 정보를 찾아내어 웹툰의 아이디와 사용자의 아이디를 서버로 보내 숨김을 해제한다. 서버에서 코드 200을 돌려받은 경우 현재 데이터셋에서도 해당 웹툰을 삭제하고, notifyDataSetChanged() 메소드를 통해서 데이터가 변경되었음을 adapter에 알린다.

3.2.10.4. 공지사항

공지사항의 경우 현재 공지할 내용이 많지 않아 공지사항 버튼을 클릭하면 팝업 메시지가 나오는 형식으로 간단하게 구현하였다

3.2.10.5. 문의

문의 버튼을 클릭하면 ACTION_SEND를 속성으로 가지는 인텐트를 통해 readers.cau@gmail.com으로 문의 이메일을 작성할 수 있도록 구현하였다.

Menu4Fragment.java, ManageLoginFragment.java,
ProfileFragment.java, BlockCard.java, BlockFragment.java,
BlockListAdapter.java, LoginWebViewActivity.java

3.3. 서버 (AWS)

기본 서버는 아마존 AWS EC2를 사용하여 구성하였다. 안드로이드와 DB, 서버 간 통신을 위해 사용한 소프트웨어 플랫폼은 nodejs이고, 다음과 같은 방식으로 서버를 사용하였다.

```

app.use('/users', usersRouter);
app.use('/bookmark', require('./routes/bookmark'));
app.use('/memo', require('./routes/memo'));
app.use('/subscribe', require('./routes/subscribe'));
app.use('/toon', require('./routes/toon'));
app.use('/block', require('./routes/block'));
app.use('/recommend', require('./routes/recommend'));
app.use('/search', require('./routes/search'));

```

users는 유저 정보에 관한 통신을 담당, bookmark는 북마크 정보에 관한 통신을 담당, memo는 유저의 메모 정보에 관한 통신을 담당하는 등, 위와 같은 식으로 코드를 작성하였다. 안드로이드와 nodejs 간 통신은 유저 정보 보안을 위하여 @GET 방식이 아닌 @POST 방식으로 진행하였다. 예를 들어 유저가 닉네임을 변경한다는 정보를 통신하는 코드는 다음과 같다.

```

router.post('/changenname', function(req, res){
    console.log(req.body);
    var user_id = req.body.user_id;
    var user_name = req.body.user_name;
    var params = [user_name, user_id];
    var sql = 'update user_info set user_name = ? where user_id = ?';

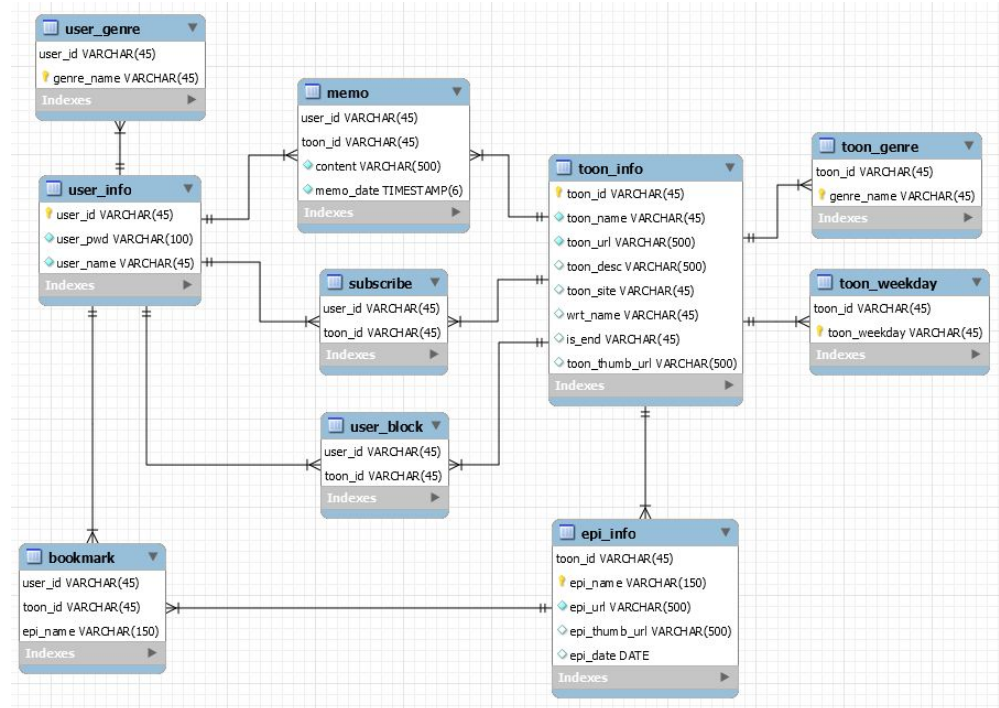
    connection.query(sql, params, function(err, rows){
        if(err) return res.sendStatus(400);
        res.status(200).json(rows);
    });
});

```

3.4. 데이터베이스

3.4.1. DB Schema

Readers 앱의 DB schema는 아래와 같다.



해당 DB Schema를 바탕으로 각 DB 테이블에 대해 살펴본다.

3.4.1.1. toon_info

웹툰의 정보를 저장하는 테이블이다. 본 앱에서 고유적으로 부여한 웹툰 ID, 웹툰 이름, 웹툰 URL, 웹툰 설명, 웹툰 연재처, 작가 이름, 완결 여부, 웹툰 썸네일의 정보를 담고 있다.

3.4.1.2. toon_genre

웹툰의 장르를 저장한 테이블이다. 웹툰의 장르는 다중값 속성이기 때문에 따로 테이블을 생성하였다. 웹툰 ID, 장르 이름을 포함하고 있다.

3.4.1.3. toon_weekday

웹툰의 연재 요일을 저장한 테이블이다. 웹툰의 연재 요일 또한 다중값 속성이기 때문에 따로 테이블을 생성하여 관리한다. 웹툰 ID, 연재 요일의 값을 갖고 있다.

3.4.1.4. epi_info

각 웹툰의 회차 정보를 저장한 테이블이다. 해당하는 웹툰의 ID, 에피소드 이름, 에피소드 URL, 에피소드 썸네일, 에피소드 업데이트 날짜를 담고 있다.

3.4.1.5. user_info

유저 정보를 저장하고 있는 테이블이다. 유저의 ID, 유저의 암호화된 비밀번호, 유저의 닉네임을 갖고 있다.

3.4.1.6. user_genre

유저가 회원가입 시에 선택한 선호 장르를 저장하고 있는 테이블이다. 장르는 최소 3개 이상 선택 가능하므로 다중값 속성이기 때문에 따로 테이블을 두어 관리한다. 유저 ID, 장르 이름을 포함하고 있다.

3.4.1.7. memo

유저가 웹툰을 보고 해당 웹툰에 관한 메모를 작성했을 때 그 정보를 담고 있는 테이블이다. 유저 ID, 웹툰 ID, 메모 내용, 메모 작성 시간을 담고 있다.

3.4.1.8. subscribe

유저의 웹툰 구독 정보를 저장한 테이블이다. 유저 ID, 웹툰 ID를 포함하고 있다.

3.4.1.9. user_block

유저의 웹툰 숨김 정보를 저장한 테이블이다. 유저 ID, 웹툰 ID를 포함하고 있다.

3.4.1.10. bookmark

유저가 해당 에피소드를 북마크한 정보를 저장하는 테이블이다. 유저 ID, 웹툰 ID, 에피소드 제목을 포함하고 있다.

3.4.2. 연재처별 크롤링

각 연재처마다 해당 정보가 존재하지 않는 등의 특수한 경우를 제외하고 공통적으로 수집한 정보는 다음과 같다.

연재 및 완결 작품: 작품명, 작가명, 작품 설명, 작품 썸네일 이미지 주소, 작품 상세 페이지 주소, 장르 정보
연재 중 작품: 연재 요일
모든 작품: 에피소드 제목, 에피소드 업데이트 날짜, 에피소드 썸네일 이미지 주소, 에피소드 링크

*위는 DB를 갖다쓰는 수준이 아니기 때문에 법적 문제의 소지가 없다고 자문 받음

웹툰의 구분을 위해 하나의 작품 ID를 [연재처_고유번호]의 형식으로 설정하였다. 이때 고유번호란 작품 상세페이지로 이동할 때 쓰이는 링크 주소 중 그 고유값을 의미한다. 예를 들어, 네이버에서 연재중인 <신의 탑> 작품의 경우 연재처가 네이버이고 작품의 링크는 <https://comic.naver.com/webtoon/list.nhn?titleId=183559> 이므로 작품 ID는 naver_183559이 된다.

DB를 끌어오기 위해 사용한 방법은 총 두가지이다.

3.4.2.1. 데이터 분석

네이버의 경우 현재 연재중인 웹툰과, 완결 웹툰의 목록을 끌어온 다음, 해당 웹툰의 URL에 들어가 에피소드를 크롤링하는 방식으로 진행한다. 연재중인 웹툰과, 완결 웹툰 목록의 링크는 다음과 같다.

<http://comic.naver.com/webtoon/weekday.nhn>

<https://comic.naver.com/webtoon/finish.nhn>

우선 연재중인 웹툰 목록 크롤링과 그 정보에 대해 살펴본다. 'div class=thumb' 태그의 내용에 요일별 웹툰 목록이 들어있는 것을 확인하고, 해당 웹툰의 URL을 추출하여 웹툰 ID와 웹툰의 연재 요일을 추출하였다. 예를 들어

<https://comic.naver.com/webtoon/list.nhn?titleId=183559&weekday=mon> 가 추출한 웹툰의 URL이라면, ID는 'titleId' 값을 이용하여 'naver_183559'로 추출, 요일은 'weekday' 값을 이용하여 'mon'으로 추출한다. 웹툰에 관한 상세 정보는 'div class=detail'에서 찾을 수 있었고, 그 안에 'h2' 태그 안에서 제목 정보와 'span class=wrt_nm'에서 작가 정보를 추출할 수 있었다. 또한 장르 정보는 'span class=genre'에서, 작품 설명은 'p' 태그에서 확인할 수 있었다. 마지막으로 썸네일 주소는 'img'의 'src' 태그에서 추출하였다.

완결 웹툰 목록 크롤링도 이와 같은 형식이다. 'div class=thumb' 태그의 내용에 완결 웹툰 목록이 들어있는 것을 확인하고, 해당 웹툰의 URL을 추출하여 웹툰 ID와 웹툰의 연재 요일을 추출하였다. 제목, 작가, 요일, 썸네일, 장르 등등의 정보는 위에 설명한 요일별 웹툰 크롤링 시의 태그와 일치함을 확인하여 크롤링 코드를 동일하게 진행하였다.

마지막으로 각 웹툰의 회차정보를 분석하고, 크롤링하는 방식에 대해 살펴본다. 앞의 요일별 웹툰, 완결 웹툰 정보에서 크롤링한 해당 웹툰의 URL에 들어가, 페이지의 HTML 소스 코드를 분석하고 원하는 내용을 추출하는 방식으로 진행하였다. "table", {"class": "viewList"}에 해당 웹툰들의 회차정보를 포함한 내용이 들어있음을 알 수 있었고, 이 안에서 에피소드 회차, 에피소드 썸네일, 에피소드 제목, 업데이트 날짜를 알 수 있다. img['src'] 태그를 통해 썸네일 링크를 추출할 수

있고, `img['title']` 태그를 통해 에피소드 제목을 추출할 수 있었다. 또한 업데이트 날짜는 `'td', {'class' : 'num'}` 태그 값에 들어있는 것을 알 수 있었다. 이와 같이 추출한 정보는 다음과 같은 SQL 쿼리문을 통해 DB에 삽입하였다.

```
ins_epi_info_sql = 'insert into epi_info(toon_id, epi_name, epi_url,
epi_thumb_url, epi_date) values (%s, %s, %s, %s, %s)'
```

3.4.2.2. 크롬 웹드라이버 Selenium

네이버를 제외한 총 6개의 연재처는 크롬의 Webdriver를 이용하여 정보를 크롤링하였다. 대략적인 순서는 연재중인 웹툰, 완결된 웹툰, 에피소드 수집의 단계로 이루어졌다.

구체적인 방법을 설명하면 우선 HTML 소스를 읽어 작품명, 작가명, 썸네일 주소 등에 해당하는 class name이나 xpath를 파악한다. 연재 요일의 경우 중복되는 경우가 존재할 수 있으나, `toon_info`나 `epi_info`는 하나의 값에 대해 하나의 row만 존재해야 하므로 해당 정보가 duplicate한 지 `fetchall()`을 실행한 뒤 DB 입력을 진행한다. 데이터의 순차적 액세스를 위해 데이터베이스 커서를 이용하였으며 쿼리문은 다음과 유사하다.

```
ins_sql = "insert ignore into toon_info(toon_id, toon_name, toon_url,
toon_desc, toon_site, wrt_name, is_end, toon_thumb_url) values (%s,
%s, %s, %s, %s, %s, %s, %s)"
```

이 과정을 요일별 탭과 같은 ``에 존재하는 모든 아이템에 대해 반복한다.

셀레니움을 이용하며 웹 스크래핑 시 그 속도가 너무 빨라 정보를 제대로 읽어오지 못하는 문제가 간혹 발생한다. 이런 경우, `implicitly_wait(3)`와 같은 옵션을 이용하여 `NoSuchElementException`를 방지하며, 추가로 blank 이미지를 불러왔을 때엔 루프문을 돌려 해당 소스를 얻을 때까지 작업을 되풀이한다.

```
if webtoon.get_attribute('srcset') != "http://cdn.lezhin.com/files/assets/blank.png":
    toon_thumb_url = webtoon.get_attribute('srcset')
else: toon_thumb_url = webtoon.get_attribute('data-srcset')
```


3.4.3. DB 실시간 업데이트

웹툰의 경우, 연재 요일에 따라 작품이 업로드되므로 하루에 한 번 이상의 업데이트가 필요하다. 이를 위해 nodejs의 'node-schedule', 'child-process' 라는 라이브러리를 이용하여 매 서버 시간마다 위에서 작성한 크롤링 코드를 작동시키는 방식으로 DB 실시간 업데이트를 구현하였다. 웹툰들은 대부분 11시~12시 사이에 연재처에 업데이트가 되므로, 우리는 넉넉하게 오전 12시 30분에 해당 연재처의 웹툰 정보를 크롤링하고자 하였다. 실시간으로 DB를 업데이트하는 코드는 다음과 같다.

```
var schedule = require('node-schedule');
var spawn = require('child_process').spawn;

var job = schedule.scheduleJob('0 30 0 * * *', function(){
  var process1 = spawn('python3', ["/crawling/naver_end.py"]);
  var process2 = spawn('python3', ["/crawling/naver_in_series.py"]);
  var process3 = spawn('python3', ["/crawling/naver_episode.py"]);
  console.log(`I'm crawling now`);
});
```

4. 개선사항 및 확장성

4.1. 유료 작품 결제

각 연재처마다 유료 작품 결제 단위와 방식이 상이하여 현재 개발하려는 플랫폼 자체에 통합 결제 시스템을 도입하는 것은 무리가 있다. 따라서 우선 무료로 열람할 수 있는 작품들에 한해 서비스를 제공하고, 각 사이트 별 사용자가 접근 가능한 유료 작품은 지원, 향후 연재처 등과의 파트너십을 통해 플랫폼 자체 코인/포인트와 같은 제도를 도입하도록 한다.

4.2. 보기 방식

연재처와 협의가 되지 않은 방식이라 웹툰 회차 내 상세 이미지를 크롤링하는 것은 저작권에 위배된다. 따라서 단순한 브라우저 로딩 방식만을 이용해야 하는 한계점이 있다.

4.3. 중복 작품

상이한 연재처에 동일 작품이 연재 중인 경우가 존재한다. 엄연히 다른 사이트에, 개별의 아이디와 고유의 방식으로 연재되고 있기에 DB를 크롤링해 오는 차원에서 이를 구분할 수 없으며, 구분할 필요도 없다.

4.4. 웹툰 추천 방식

현재 웹툰 추천 방식은 장르 태그를 기반으로 한다. Cold-start 문제를 해결하기 위해 회원가입 시에 사용자가 기존에 선호하던 장르 정보를 3개 이상 받으며, APP을 사용하면서 축적한 활동 데이터(구독, 책갈피, 메모, 숨김 등)를 바탕으로 추천 범위를 확대한다. 차후 작품 뷰 타임, 작가 정보 등 추가적인 장치들을 활용하여 추천 시스템을 발전시킬 예정이다.

5. 결론

5.1. 대상 및 범위

5.1.1. 지원 연재처 수와 작품 종류

지원하는 연재처(네이버, 다음, 레진코믹스, 미스터블루, 버프툰, 봄툰, 카카오페이지 등 7곳) 내 무료로 열람할 수 있는 전체이용가 웹툰을 기본적으로 제공한다. 단, 사용자가 특정 에피소드에 대해 구매한 내역이 있고 해당 사이트에 로그인 한 상태면 어플 내에서 보기 가능하다.

5.1.2. 축적 DB 자원

(2019년 12월 첫째주 기준) 현재 어플 내에 저장된 작품 수와 에피소드 개수를 연재처별로 분류하면 아래 표와 같다.

연재처명	작품수	에피소드수
네이버	943	58009
다음	372	23447
레진코믹스	365	20779
미스터블루	325	16932
버프툰	151	12513
봄툰	658	30660
카카오페이지	84	4653
합계	2898	166781

5.2. 차별화 기능

5.2.1. 통합 로그인 기능

유저의 필요에 따라 각 사이트 별 로그인 정보를 저장하여 이후 구매한 에피소드를 보거나 성인 인증 등의 단계에서 추가적인 절차가 없도록 한다. 이 상태는 유저가 별도로 로그아웃 조치를 취하지 않고 브라우저 내에 쿠키 정보가 살아있다면 유지된다. 리더스 앱 자체 로그아웃 시 카카오페이지를 제외한 연재처는 자동 로그아웃되는 편의도 제공한다.

5.2.2. 통합 정렬 및 검색 기능

여러 연재처에 존재하는 모든 작품들을 다양한 기준으로 정렬하여 사용자에게 보다 편리한 탐색 기능을 제공한다. 1차 정렬으로는 요일/장르/ 완결 정보가 사용되었으며, 2차 정렬으로는 제목/ 업데이트/ 연재처순이 이용되었다.

상단의 검색바를 이용하여 특정 키워드를 입력하면 결과에 해당하는 작품명, 작가명 혹은 작품 설명을 포함한 웹툰 작품을 찾을 수 있다.

5.2.3. 유저 개인별 취향 분석 및 추천 기능

연재처의 구분 없이 사용자가 좋아하는 웹툰을 구독하고 짧은 코멘트를 남길 수 있으며, 각 에피소드 별로 책갈피를 표시할 수 있다. 보기 불편한 작품은 차단 개념인 숨김 기능을 이용하면 된다.

이 정보들은 앱 설치 시에 입력받은 장르 선호도와 함께 분석되어 사용자의 취향을 파악 후 새로운 웹툰을 추천해준다. 상위 3개 장르 작품에 대해 작품명, 작가명, 설명을 제공하여 단순 나열 형식을 탈피해 흥미를 유발한다. 또한, 사용자 개인의 취향에 대한 분석 데이터는 그래프 형식으로 본인이 직접 시각적으로 확인할 수 있게 돕는다.

5.3. 매뉴얼

어플리케이션을 사용하기 위한 매뉴얼은 아래 링크를 통해 접속하면 확인하면 수 있다. 회원가입/로그인, 구독/책갈피/메모 사용 방법, 추천페이지 구성, 설정 등을 어떻게 이용하면 되는 지 간단하게 확인할 수 있다.

<https://bit.ly/350vwUL>