웹프로그래밍실습 팀프로젝트 결과보고서

4조

김동호

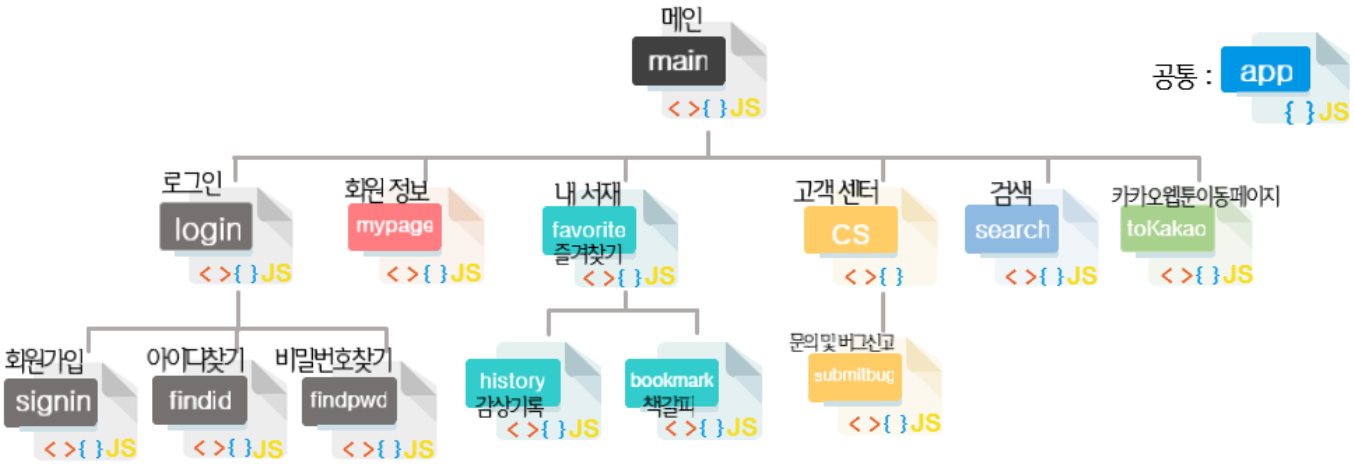
임승현

조준

1. 홈페이지 개요 및 설명

국내 웹툰 산업 규모가 매년 성장해왔고 다수의 플랫폼이 활성화됨에 따라, 다양한 웹툰 정보를 통합하여 제공하는 사이트가 있다면 유용할 것이라고 판단하였다. 따라서 여러 플랫폼의 웹툰 정보를 모은 종합 검색 사이트를 구상하였고, 검색 기능뿐만 아니라 계정별로 즐겨찾기, 감상 기록, 책갈피를 저장하는 기능을 추가하였다. 또한, 웹툰 정보 페이지에서 각 웹툰의 에피소드 페이지로 곧바로 연결되도록 하여 웹툰을 편리하게 열람할 수 있는 사이트를 구축하였다.

2. 전체 사이트 맵



3. 구성 페이지 설명(HTML, CSS)

A. 모든 페이지 공통 부문

i. head 태그: ①메타 태그, ②파비콘 관련, ③폰트 관련, ④공통 스타일시트

```
<head>
    <!-- 메타 태그 -->
    <meta charset="utf-8" />
    <meta http-equiv="x-ua-compatible" content="ie=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta property="og:image" content="image/og_iwdb_logo.png" />

    <!-- 파비콘 관련 -->
    <link rel="apple-touch-icon" sizes="57x57" href="images/favicon/apple-icon-57x57.png" />
    <link rel="apple-touch-icon" sizes="60x60" href="images/favicon/apple-icon-60x60.png" />
    <link rel="apple-touch-icon" sizes="72x72" href="images/favicon/apple-icon-72x72.png" />
    <link rel="apple-touch-icon" sizes="76x76" href="images/favicon/apple-icon-76x76.png" />
    <link rel="apple-touch-icon" sizes="114x114" href="images/favicon/apple-icon-114x114.png" />
    <link rel="apple-touch-icon" sizes="120x120" href="images/favicon/apple-icon-120x120.png" />
    <link rel="apple-touch-icon" sizes="144x144" href="images/favicon/apple-icon-144x144.png" />
    <link rel="apple-touch-icon" sizes="152x152" href="images/favicon/apple-icon-152x152.png" />
    <link rel="apple-touch-icon" sizes="180x180" href="images/favicon/apple-icon-180x180.png" />
    <link rel="icon" type="image/png" sizes="192x192" href="images/favicon/android-icon-192x192.png" />
    <link rel="icon" type="image/png" sizes="32x32" href="images/favicon/favicon-32x32.png" />
    <link rel="icon" type="image/png" sizes="96x96" href="images/favicon/favicon-96x96.png" />
    <link rel="icon" type="image/png" sizes="16x16" href="images/favicon/favicon-16x16.png" />
    <meta name="msapplication-TileColor" content="#ffffff" />
    <meta name="msapplication-TileImage" content="images/favicon/ms-icon-144x144.png" />
    <meta name="theme-color" content="#ffffff" />

    <!-- 폰트 관련 -->
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@700&display=swap" rel="stylesheet" />

    <!-- 공통 스타일시트 -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" />
    <link rel="stylesheet" href="css/foundation.css" />
    <link rel="stylesheet" href="css/app.css" />


                                                    ⋮
                                                  생략
</head>
```

④공통 스타일시트의 경우 순차적으로

1. Font awesome 아이콘 스타일시트
2. Foundation 프레임워크 스타일시트
3. 모든 페이지 공통 스타일시트 (app.css)

ii. body 태그 : ①고정된 Top bar, ②footer, ③공통 자바스크립트

```
<!-- Dark Mode 전환 시, body 태그에 class="dark-mode" 추가됨 -->
<body>
    <!-- 고정된 Top bar 파트 -->
        <!-- Top bar 좌측 파트 -->
            <!-- 로고 -->
            <!-- 로고 글자 -->

        <!-- Top bar 우측 파트 -->
            <!-- 내 서재 Dropdown Menu 파트 -->
            <!-- 여백 -->
            <!-- 계정 Dropdown Menu 파트 -->
            <!-- 로그인 파트 -->
            <!-- 여백 -->
            <!-- 검색 박스 파트 -->
            <!-- 여백 -->
            <!-- 다크 모드 스위치 -->

                                    ⋮
                                  생략

    <!-- footer 파트 -->
        <!-- 여백 -->
        <!-- 고객 센터 -->        <
        <!-- 문의 메일 -->
        <!-- 출처 -->
        <!-- 여백 -->

    <!-- 공통 자바스크립트 -->
    <script src="js/vendor/jquery.js"></script>
    <script src="js/vendor/what-input.js"></script>
    <script src="js/vendor/foundation.js"></script>
    <script src="js/app.js"></script>

</body>
```

[코드가 너무 긴 부분은 주석만 남겨 계층만 표시]

③공통 자바스크립트의 경우

1. 위의 3개는 Foundation 프레임워크를 위한 자바스크립트 파일
2. 4번째는 모든 페이지 공통 자바스크립트 파일(app.js)

iii. 모든 페이지 공통 스타일시트 (app.css) :

2. :root 로 테마 색상 지정 (black, light-black, dark-black, white, yellow, dark-yellow, grey, light-grey)
3. 자주 재사용되는 클래스별 스타일 작성 (black-bg, black-font, white-bg, white-font, radius, rounded, dark-mode 등)
4. Foundation 프레임워크 기본 색상 웹 테마에 맞추어 오버라이드
5. body 태그에 반복되는 요소들 스타일 지정 (#logo, #my-library-text, #nickname-text 등)
6. 다크모드 시 공통요소 스타일

B. HTML 작성에 사용된 Foundation 프레임워크 요소

요소 이름	요소 설명 [사용된 페이지]	관련 링크
Off-canvas	Viewport 밖에서 슬라이드하며 들어오는 패널로써, 웹툰 카드를 클릭하였을 때 웹툰 정보 페이지를 띄우는데 사용 [메인, 즐겨찾기, 검색]	https://get.foundation/sites/docs/off-canvas.html
Sticky	요소를 고정하는데 사용. Top-bar과 같이 쓰여 화면 상단에 고정 [모든 페이지]	https://get.foundation/sites/docs/sticky.html
Top-bar	상단 바 [모든 페이지]	https://get.foundation/sites/docs/top-bar.html
Dropdown Menu	드랍다운 메뉴. 로그인 후, 내 서제 메뉴와 계정 메뉴에 사용 [모든 페이지]	https://get.foundation/sites/docs/dropdown-menu.html
XY Grid	화면 크기에 반응하는 Responsive한 Grid [모든 페이지]	https://get.foundation/sites/docs/xy-grid.html
Orbit	Carousel Slider [메인 페이지]	https://get.foundation/sites/docs/orbit.html
Breadcrumbs	페이지 이동에 쓰이는 네비게이션 템플릿의 일종 [로그인, 회원가입, 즐겨찾기, 감상 기록, 책갈피]	https://get.foundation/sites/docs/breadcrumbs.html
Tabs	웹툰 요일 별 탭과 웹툰 에피소드 페이지 탭 구현에 사용 [메인, 즐겨찾기, 검색]	https://get.foundation/sites/docs/tabs.html
Abide	검증 기능이 있는 폼 [회원가입]	https://get.foundation/sites/docs/abide.html
Accordian	아코디언 컨테이너 [고객센터]	https://get.foundation/sites/docs/accordion.html

C. 개별 페이지 HTML, CSS

페이지 명	HTML [코드가 너무 긴 까닭에 공통된 부문은 제외한 body 의 나머지 부문을 주석만을 이용해 계층으로 표현]	CSS
메인, 즐겨찾기, 검색	<!-- 웹툰 목록 페이지와 웹툰 정보 페이지가 공존하기 위한 Off-canvas --> <!-- 웹툰 목록 페이지 --> <!-- 회색선 위 파트 --> <!-- 여백 --> <!-- 페이지 이동 항목 --> <!-- 회색선 --> <!-- 아래 파트 --> <!-- 웹툰 카드 모음 화면 --> <!-- 웹툰 정보 페이지 : 페이지 하단에서 위쪽 방향으로 출현 애니메이션 적용된 Off-canvas --> <!-- 위 파트 --> <!-- 뒤로가기 버튼 --> <!-- 웹툰 정보 관련 --> <!-- 썸네일 --> <!-- 정보 --> <!-- 여백 --> <!-- 아래 파트 --> <!-- 웹툰 에피소드 모음 화면 --> <!-- 1 페이지 --> <!-- 에피소드 카드 예시 --> <!-- 2 페이지 --> <!-- 페이지 관리 --> <!-- 여백 -->	Carousel Slider 기본 스타일 오버라이드 [메인만] 웹툰 썸네일 호버링 시 확대 효과 탭 기본, 호버링, 클릭 시 스타일 즐겨찾기 버튼 기본 및 호버링 스타일 페이지 이동 버튼 기본, 호버링, 클릭 스타일 그외 버튼(에피소드 카드 포함) 기본, 호버링, 클릭 스타일 "검색 결과" 글자 스타일 [검색만] 다크모드 시 스타일
로그인	<!-- 회색선 위 파트 --> <!-- 여백 --> <!-- 페이지 이동 항목 --> <!-- 회색선 --> <!-- 로그인 상자 -->	로그인 탭 적당한 위치로 이동 로그인, 회원가입 란을 정돈 로그인 상자 속 input 들 form 스타일 로그인 버튼 스타일 아이디 찾기, 비밀번호 찾기 등의 스타일 다크모드 시 스타일
회원가입	<!-- 회색선 위 파트 --> <!-- 여백 --> <!-- 페이지 이동 항목 --> <!-- 회색선 --> <!-- 회색선 아래 파트 --> <!-- 검증 기능을 탑재한 폼 -->	Foundation 기존 버튼 스타일 오버라이드 다크모드 시 스타일 폼 내부 input, 버튼, 컨테이너 스타일

아이디 찾기	<!-- 제목 div --> <!-- 폼 --> <!-- 테이블 --> <!– 회원 이름 레이블 --> <!– 회원 이름 input --> <!-- 이메일 레이블 --> <!-- 이베일 input --> <!-- 아이디 찾기 버튼	아이디 찾기 폼 둥글게 스타일 아이디 찾기 버튼 기본, 호버링 스타일 폼 내부의 레이블과 input 스타일
비밀번호 찾기	<!-- 제목 div --> <!-- 폼 --> <!-- 테이블 --> <!-- 아이디 레이블 --> <!-- 아이디 input --> <!-- 회원이름 레이블 --> <!-- 회원이름 input --> > <!-- 이메일 레이블 --> <!-- 이메일 input --> <!-- 비밀번호 찾기 버튼 -->	비밀번호 찾기 제목 스타일 비밀번호 찾기 폼 둥글게 스타일 비밀번호 찾기 버튼 스타일 폼 내부의 레이블과 input 스타일 테이블 스타일
회원정보	<!-- 내 정보가 나타나는 상자 --> <!-- 홍길동 님 예시 --> <!-- 개인 정보 테이블 --> <!-- "회원 아이디" --> <!-- 회원 아이디 정보 --> <!-- "닉네임" --> <!-- 닉네임 정보 --> < <!-- "이메일" --> <!-- 이메일 정보 -->	내 정보가 나타나는 상자 스타일 개인정보 출력되는 칸 스타일 테이블 스타일 다크모드 시 스타일
감상 기록, 책갈피	<!-- 회색선 위 파트 --> <!-- 여백 --> <!-- 페이지 이동 항목 --> < <!-- 회색선 --> <!-- 아래 파트 --> <!-- 웹툰 에피소드 카드 모음 화면 -->	감상 기록 버튼 기본 및 호버링 스타일 [감상 기록만] 책갈피 버튼 기본 및 호버링 스타일 [책갈피만] 에피소드 카드 포함 기본, 호버링, 클릭 스타일 다크모드 시 스타일
고객센터	<!-- 회색선 위 파트 --> <!-- 페이지 제목 --> <!-- 회색선 --> <!-- FAQ 형식 예시 --> < <!-- FAQ 컨테이너 --> <!-- 1번 질문 --> <!-- 1번 답 -->	페이지 제목 스타일 회색선 스타일 "문의 및 버그신고" 글자 스타일 다크모드 시 스타일

	<!-- 2 번 질문 --> <!-- 2 번 답 --> <!-- 3 번 질문 --> <!-- 3 번 답 --> <!-- 4 번 질문 --> <!-- 4 번 답 --> <!-- 5 번 질문 --> <!-- 5 번 답 -->	
문의 및 버그신고	<!-- 회색선 위 파트 --> <!-- 여백 --> <!-- 페이지 제목 --> <!-- 회색선 --> <!-- 이메일 입력 부분 --> <!-- 이메일 아이디 입력 --> <!-- 이메일 도메인 입력--> <!--도메인 선택--> <!-- 주제 선택 부분 --> <!-- selected : 초기선택 옵션으로 설정, disabled : 옵션 선택할수 없음 (비활성화) --> <!-- 설명 입력 부분 --> <!-- 제출 버튼 -->	회색선 스타일 이메일 입력 스타일 주제 선택 부분 스타일 설명 입력 부분 스타일
카카오 웹툰 이동페이지	<!-- 카카오 페이지로 이동함을 알리는 상자 --> <div class="callout secondary warning title large-12" style="margin: 0 auto; border-radius: 5px; border: 1px solid #ddd; transform: translate(0, 20%); width: 60%" > <h2 class="text" id="msg"> 카카오웹툰 사이트 로그인이 필요한 서비스입니다. </h2> <h3 class="text"> 3초 후에 카카오 웹툰의 해당 웹툰 페이지로 이동합니다... </h3> <!-- 3 초를 세는 프로그레스 바 --> <div class="progress warning" role="progressbar" aria-valuenow="100" aira-valuemin="0" aria-valuemax="0" style="width: 60%; margin: 0 auto; transform: translate(0, 500%)" > 	상자 스타일 프로그레스 바 스타일

4. 기능 설명(Javascript, Libraries)

A. 웹툰 크롤링 데이터 JSON 형식 예시


```
[
   {
      "title": "싸움독학",
      "author": "박태준 만화회사,김정현 스튜디오",
      "thumbnail": "https://image-comic.pstatic.net/webtoon/736277/thumbnail/thumbnail_IMAG21_bbbe3f76-021e-4dbc-830a-4358c1abec0c.jpg",
      "day": ["sun"],
      "about": "힘없고 가진거 하나 없이 맞고만 살던 나였는데...우연히 비밀의 뉴튜브를 발견하게 되고 갑자기 떼돈을 벌었다.",
      "tag": ["#액션", "#학원물", "#인플루언서", "#소년왕도물"],
      "id": 736277,
      "platform": "naver",
      "episode": [
         {
            "episodeTitle": "180 화 : 넌 오늘 뒤졌다!",
            "episodeURL": "https://comic.naver.com/webtoon/detail?titleId=736277&no=184"
         },
         {
            "episodeTitle": "179 화 : 더 해요 우리",
            "episodeURL": "https://comic.naver.com/webtoon/detail?titleId=736277&no=183"
         }]
   }
]
```


웹툰 제목, 작가, 썸네일, 연재 요일, 줄거리, 태그, 그리고 각 웹툰의 회차 별 제목, url주소를 크롤링하여 총 1400여개의 작품 정보를 IWDB사이트에 반영하였다. 회차 정보는 최신 회차 스무 편만 크롤링하였다.

B. 더미 계정 데이터 JSON 형식


```
[
 {
  "id": "myeongnyun",
  "password": "qwer1234",
  "name": "명륜이",
  "nickname": "인문사회최고",
  "email": "myeongnyun@naver.com",
  "favorite": [807831, 602910],
  "history": ["https://comic.naver.com/webtoon/detail?titleId=790713&no=123", "https://comic.naver.com/webtoon/detail?titleId=790713&no=122"],
  "bookmark": ["https://comic.naver.com/webtoon/detail?titleId=70260&no=283", "https://comic.naver.com/webtoon/detail?titleId=602910&no=439"]
  }
 ]
```

C. 모든 페이지 공통 자바스크립트 (app.js)

```
// Foundation 플러그인
$(document).foundation();

// 키
const DARK_MODE_KEY = "dark_mode";
const LOGIN_USER_KEY = "login_user";
const ACCOUNT_KEY = "account";
const WEBTOON_INFO_KEY = "webtoon_info";

// 요소 선언
const darkModeSwitch = document.querySelector(".switch-input");
const body = document.body;
const myLibrayMenu = document.querySelector("#my-library-text");
const nicknameMenu = document.querySelector("#nickname-text");
const loginMenu = document.querySelector("#login-text");
const searchInput = document.querySelector("#search-box input");
const searchButton = document.querySelector("#search-button");


// 다크 모드
let isDarkModeOn = false;
const savedDarkMode = localStorage.getItem(DARK_MODE_KEY);
if (savedDarkMode !== null) {
  isDarkModeOn = savedDarkMode === "true";
  if (isDarkModeOn) {
    body.classList.add("dark-mode");
    darkModeSwitch.checked = true;
  }
}
function toggleDarkMode(event) {
  body.classList.toggle("dark-mode");
  isDarkModeOn = darkModeSwitch.checked;
  localStorage.setItem(DARK_MODE_KEY, isDarkModeOn);
}
darkModeSwitch.addEventListener("change", toggleDarkMode);

// 로그인 여부 확인
let login_user = null;
const savedLoginUser =
JSON.parse(localStorage.getItem(LOGIN_USER_KEY));
if (savedLoginUser !== null) {
  login_user = savedLoginUser;
  myLibrayMenu.classList.remove("hide");
  nicknameMenu.classList.remove("hide");
  loginMenu.classList.add("hide");
  nicknameText = document.querySelector("#nickname-text");
  nicknameText.innerText = login_user["nickname"];
}
```

```
// 로그아웃 및 계정 정보 업데이트
const logout = document.querySelector("#logout-text");
logout.addEventListener("click", (event) => {
  event.preventDefault();
  let accounts = JSON.parse(localStorage.getItem(ACCOUNT_KEY));
  accounts.forEach(account => {
    if(account["id"] === login_user["id"]) {
      account["favorite"] =
JSON.parse(JSON.stringify(login_user["favorite"]));
      account["history"] =
JSON.parse(JSON.stringify(login_user["history"]));
      account["bookmark"] =
JSON.parse(JSON.stringify(login_user["bookmark"]));
    }
  });
  localStorage.setItem(ACCOUNT_KEY, JSON.stringify(accounts));
  localStorage.removeItem(LOGIN_USER_KEY);
  location.href = "./main.html";
});

// 검색 기능
searchButton.addEventListener("click", ()=> {
  location.href = `./search.html?query=${searchInput.value}`;
});
searchInput.addEventListener("keyup", (event)=>{
  // 엔터키인지 확인
  if (event.keyCode === 13) {
  location.href = `./search.html?query=${searchInput.value}`;
  }
});
```

1. 다크모드 스위치의 "change" 이벤트가 발생하면 다크모드 여부를 토글하고 이를 로컬 저장소에 키를 dark_mode로 boolean 값을 저장한다. 이후 매 페이지가 시작될 때 로컬저장소에 해당 값을 읽어드리고 boolean 값에 따라서 body 태그에 클래스에 dark-mode를 추가한다. 이와 관련하여 css 파일에 스타일을 지정해 놓아서 다크모드를 구현하였다.

2. 로그인 여부는 로컬 저장소에 키를 login_user로 하는 값이 존재하는지에 따라 판별하게 된다. 앞선 더미계정의 JSON 형식의 값이 저장되어 있다면 로그인 상태이고 그에 따라 Top-bar의 숨겨진 메뉴(내 서재, 회원정보 등)는 보이게 하고 기존에 보이던 메뉴(로그인)는 숨긴다.

3. 로컬 저장소에는 키를 account로 하는 값이 저장되어 있는데 이는 전체 계정들에 대한 정보이다. 따라서 로그아웃 버튼을 클릭하면 현재 로그인 되어있는 유저의 정보인 login_user을 로컬저장소의 전체 계정들 정보에 업데이트를 수행한 후, 로컬저장소에서 키를 login_user로 하는 값을 삭제한다. 이후 모든 페이지에서 로그인 여부를 확인할 때 해당 값이 존재하지 않아 비로그인 상태로 여겨진다.

4. Top-bar에 존재하는 검색 창에 검색 키워드를 입력하고 검색 버튼에 해당하는 돋보기 버튼을 누르거나 또는 엔터를 누르면 검색 페이지의 주소의 꼬리에 ?query={검색 키워드} 형태로 키워드를 전달해주면서 검색 페이지로 이동한다.

D. 개별 페이지 자바스크립트

페이지 명	자바스크립트 [코드가 너무 긴 까닭에 주석만을 이용해 계층으로 표현]]	간략한 설명
메인, 즐겨찾기, 검색, 감상 기록, 책갈피	// 요소 및 변수 선언 // 더미 계정 데이터 localStorage 에 저장 // 요일 탭 전환 구현 // 요일별 분류 // 네이버 요일별 카드 생성 함수 // 웹툰 카드 클릭 이벤트 발생 시 웹툰 정보 캔버스 업데이트 // 즐겨찾기 버튼이 클릭된것이 아닌 경우에만 // 에피소드 카드 페이지 설정 부분 // 2 페이지 채워질 경우 // 페이지 버튼 초기화 // 페이지 내용 초기화 // 첫 번째 페이지 채우기 // 에피소드 카드 생성 // 감상 기록 및 책갈피 반영 부분 // 감상 기록 여부 반영 // 책갈피 여부 반영 // 에피소드 카드 클릭 이벤트 추가 // 책갈피 관련 클릭 시 // 로그인 여부 확인 // 비로그인 시 // 로그인 시 // 책갈피 해제 작업 // 책갈피 추가 작업 // 에피소드 카드 본체 클릭 시 // 로그인 여부 확인 후 감상 기록 추가 // 두 번째 페이지 채우기 // 에피소드 카드 생성	로컬 저장소에 더미 계정 데이터(전체 계정들에 대하 정보)가 없다면 저장한다. [메인만] 웹툰 JSON 데이터를 순차적으로 접근하면서 웹툰 카드를 생성한다. 이때, 1) 메인 페이지는 필터링 없이 모든 웹툰을 2) 즐겨찾기 페이지는 현재 로그인한 계정의 즐겨찾기 목록에 있는 웹툰을 필터링해서 3) 검색 페이지는 검색 키워드가 제목, 작가, 또는 태그에 포함된 웹툰들만을 필터링해서 카드를 생성한다. 이렇게 생성된 카드가 즐겨찾기 버튼이 아닌 부분이 클릭 될 시, 클릭된 웹툰의 ID 를 바탕으로 웹툰 정보 상세 페이지의 정보를 채우게 된다. 이때 해당 웹툰의 크롤링된 에피소드 개수에 따라서 페이지 관련 부문에 에피소드 카드를 채우게 된다. 에피소드 카드는 책갈피 부문이 클릭되면 우선 로그인 여부를 확인한다. 로그인이 되어있을 경우 login_user 의 책갈피(bookmark) 리스트에 해당 에피소드 URL 의 존재 여부에 따라 토글을 수행한다. 만약 비로그인시라면 해당 기능은 로그인후 사용할 수 있다는 alert 를 내보낸다. 에피소드 카드의 본체 부문이 클릭되면 해당 웹툰의 URL 로 페이지가 이동되고 login_user 의 감상기록(history) 항목에 해당 에피소드 URL 이 추가된다. 물론 해당 값이 전에 추가된 적이 있는지와 같은 중복 확인도 수행한다. 에피소드 부문의 기능은 이로써 끝났고 다시 웹툰 카드로 돌아와서 웹툰 카드의 즐겨찾기 버튼이 클릭된 상황이라면 이 또한 앞서 책갈피 항목과 거의 동일한 확인을 거친 후 login_user 의 즐겨찾기(favorite) 리스트에 해당 웹툰의 ID 를 추가한다. 참고로 즐겨찾기, 감상 기록, 책갈피 모두 중복확인을 수행하기 때문에 리스트에 중복값이 존재할 일은 없다. 추가적으로 메인 페이지에는 Carousel Slider 가 존재하고 해당 썸네일을 클릭 시 마찬가지로 웹툰 상세정보 페이지가 나와야 하므로 이와 관련된 코드가 추가된다. 웹툰 상세 정보 페이지에 사용된 Off-Canvas 의 높이는 정확히 화면 맨 아래서부터 Top-bar 바로 아래까지여야 하기 때문에 웹브라우저 화면 크기의 변화가 생길 때마다 이를 감지하여 Off-Canvas 크기를 조종해준다.

```
          // 감상 기록 및 책갈피 반영 부분
            // 감상 기록 여부 반영
            // 책갈피 여부 반영

          // 에피소드 카드 클릭 이벤트 추가
            // 책갈피 관련 클릭 시
              // 로그인 여부 확인
                // 비로그인 시

              // 로그인 시
                // 책갈피 해제 작업
                // 책갈피 추가 작업

            // 에피소드 카드 본체 클릭 시
              // 로그인 여부 확인 후 감상 기록 추가

       // 1 페이지만 채워질 경우
         // 페이지 버튼 초기화

         // 페이지 내용 초기화
         // 첫 번째 페이지 채우기
           // 에피소드 카드 생성

          // 감상 기록 및 책갈피 반여 부분
            // 감상 기록 여부 반영
            // 책갈피 여부 반영

          // 에피소드 카드 클릭 이벤트 추가
            // 책갈피 관련 클릭 시
              // 로그인 여부 확인
              // 비로그인 시
                // 책갈피 해제 작업
                // 책갈피 추가 작업

            // 에피소드 카드 본체 클릭 시
              // 로그인 여부 확인 후 감상 기록 추가

       // 에피소드 카드 기본 페이지 초기화

    // 즐겨찾기 여부 반영

    // 즐겨찾기 버튼 클릭 이벤트

// 카카오 요일별 카드 생성 함수
                                    ⋮
                                  생략

// 네이버 요일별 카드 생성
// 카카오 요일별 카드 생성

// 웹툰 정보 페이지 크기 조정

//  웹툰 정보 페이지에서 뒤로가기 버튼 클릭 시 이벤트
```

또한 웹툰 상세 정보 페이지에 존재하는 뒤로가기 버튼이 클릭되었을 때 Off-Canvas 가 도로 사라지는 이벤트를 Foundation 프레임워크가 제공하는 API 를 통해 구현하였다.

감상 기록과 책갈피 페이지는 웹툰 카드 생성 과정은 생략하고 에피소드 카드 생성 과정만 이용한다. 그리고 추가적으로 해당 에피소드 카드 버튼에 감상 기록 삭제와 책갈피 해제 버튼을 구현하여서 해당 이벤트가 발생시 에피소드 카드가 우측으로 슬라이딩하면서 목록에서 사라지도록 구현하고 login_user 의 감상기록, 책갈피 리스트에서 해당 에피소드 URL 을 제거한다.

	`` // Carousel Slider 현재 화면 클릭 이벤트 발생 시 웹툰 정보 캔버스 업데이트 ⋮ 생략 ```	
로그인	``` const id = document.querySelector('input[type="text"]'); const password = document.querySelector('input[type="password"]'); const loginButton = document.querySelector('input[type="submit"]'); const savedAccount = JSON.parse(localStorage.getItem(ACCOUNT_KEY)); loginButton.addEventListener("click", () => { let login_user = savedAccount.find((currentAccount) => currentAccount["id"] === id.value && currentAccount["password"] === password.value); if (login_user === undefined) { swal('로그인 실패!', "아이디와 비밀번호를 확인해 주세요", 'error'); /*alert("아이디 또는 비밀번호가 올바르지 않습니다.");*/ id.value = ""; password.value = ""; } else { localStorage.setItem(LOGIN_USER_KEY, JSON.stringify(login_user)); location.replace("./main.html"); } }); ```	로그인을 시도 시 로컬 저장소에 존재하는 전체 계정 정보와 비교하여 유효한 경우에만 로컬 저장소에 login_user 키값으로 로그인 한 계정의 정보를 저장한다.
회원가입	``` // 요소 및 변수 선언 // 추가적인 다크 모드 // ID 중복 확인 // 공백 시 // 비중복 시 //alert("사용 가능한 아이디입니다."); // 중복 시 //alert("이미 사용 중인 아이디입니다."); // 닉네임 중복 확인 // 공백 시 // 비중복 시 //alert("사용 가능한 닉네임입니다."); // 중복 시 //alert("이미 사용 중인 닉네임입니다."); // 이메일 중복 확인 // 공백 시 // 비중복 시 //alert("사용 가능한 이메일입니다."); } // 중복 시 //alert("이미 사용 중인 이메일입니다."); // 제출 버튼 기능 ```	우선 ID, 닉네임, 이메일 input 값이 공백인지 확인하고 공백이 아닌 경우 로컬 저장소에 존재하는 전체 계정 정보와 비교하여 중복 여부를 확인한다. 최종적으로 회원가입 버튼을 눌렀을 때 모두 중복이 아니고 Foundation Abide 폼이 제공하는 검증에서 오류가 없음이 확인되었을 때 회원가입을 진행시켜준다. 로컬 저장소에 저장되어있는 전체 계정 정보에 폼에 입력된 정보를 바탕으로 새로운 계정 정보를 업데이트 한다. 또한 중복 확인을 수행한 뒤에 해당 폼의 데이터가 변화할 경우 이를 감지하여 해당 항목의 중복 여부를 다시 초기화한다.

<table>
<tr>
<td></td>
<td>

```
  // 중복 없음이 확인 되었고 폼에 오류가 없을 시
    //alert("회원가입이 완료되었습니다.");

  // 폼에 오류가 있을 시

  // 중복이 있을 시
    //alert("모든 중복확인을 수행해주십시오");
```

</td>
<td></td>
</tr>
<tr>
<td>아이디 찾기</td>
<td>

```
const idInput = document.querySelector('input[type="text"]');
const emailInput = document.querySelector('input[type="email"]');
const findIdButton = document.querySelector('input[type="submit"]');
const savedAccounts = JSON.parse(localStorage.getItem(ACCOUNT_KEY));

findIdButton.addEventListener("click", (event) => {
  event.preventDefault();

  let foundAccounts = savedAccounts.filter((currentAccount) => currentAccount["email"] === emailInput.value);
  // 입력된 이메일과 일치하는 계정을 찾음

  if (foundAccounts.length === 0) {     // 일치하는 계정이 없는 경우
     //swal('error!', "일치하는 계정을 찾을 수 없습니다.", 'error');

    alert("일치하는 계정을 찾을 수 없습니다.");
    emailInput.value = "";
  } else {
    let foundIds = foundAccounts.map((account) => account["id"]); // 찾은 계정의 아이디를 추출하여 배열로 만듦.

    // 가려진 아이디 생성
    let hiddenIds = foundIds.map((id) => {
      const hiddenLength = Math.floor(id.length / 2); // 가려질 길이
      const hiddenPart = "*".repeat(hiddenLength); // 가려진 부분
      const visiblePart = id.slice(hiddenLength); // 가려지지 않은 부분
      return hiddenPart + visiblePart; // 가려진 아이디 반환
    });

    alert("찾은 아이디: " + hiddenIds.join(", "));
  }
});
```

</td>
<td>아이디 찾기 시도 시 로컬 저장소에 존재하는 전체 계정 정보와 비교하여 유효한 경우에만 아이디 일부를 알려준다</td>
</tr>
<tr>
<td>비밀번호 찾기</td>
<td>

```
const idInput = document.querySelector('input[type="text"]');
const nameInput = document.querySelector('input[name="name"]');
const emailInput = document.querySelector('input[type="email"]');
const findPasswordButton = document.querySelector('input[type="submit"]');
const savedAccounts = JSON.parse(localStorage.getItem(ACCOUNT_KEY));

findPasswordButton.addEventListener("click", (event) => {
  event.preventDefault();

  let foundAccounts = savedAccounts.filter((currentAccount) => currentAccount["email"] === emailInput.value &&
currentAccount["id"] === idInput.value);
    // 입력된 이메일과 ID와 일치하는 계정을 필터링하여 찾음
  if (foundAccounts.length === 0) {   // 일치하는 계정이 없는 경우
    swal('error!', "일치하는 계정을 찾을 수 없습니다.", 'error');
```

</td>
<td>비밀번호 찾기 시도 시 로컬 저장소에 존재하는 전체 계정 정보와 비교하여 유효한 경우에만 해당 계정의 이메일로 비밀번호를 전송했다는 메시지를 출력한다</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

```
    // alert("일치하는 계정을 찾을 수 없습니다.");


    idInput.value = "";   // 입력필드들의 값을 초기화
    nameInput.value = "";
    emailInput.value = "";
  } else {
    swal('success', "등록된 이메일로 비밀번호를 전송했습니다.", 'success');
    //alert("등록된 이메일로 비밀번호를 전송했습니다.");
    idInput.value = "";
    nameInput.value = "";
    emailInput.value = "";
  }
});
```

</td>
<td></td>
</tr>
<tr>
<td>회원정보</td>
<td>

```
// 요소 선언
const userName = document.querySelector("#userName");
const userID = document.querySelector("#userID");
const userNickname = document.querySelector("#nickname");
const userEmail = document.querySelector("#email");

// 로그인된 회원정보
userName.innerText = login_user["name"];
userID.innerText = login_user["id"];
userNickname.innerText = login_user["nickname"];
userEmail.innerText = login_user["email"];
```

</td>
<td>현재 로그인된 계정의 정보로 채워넣는다.</td>
</tr>
<tr>
<td>문의 및 버그신고</td>
<td>

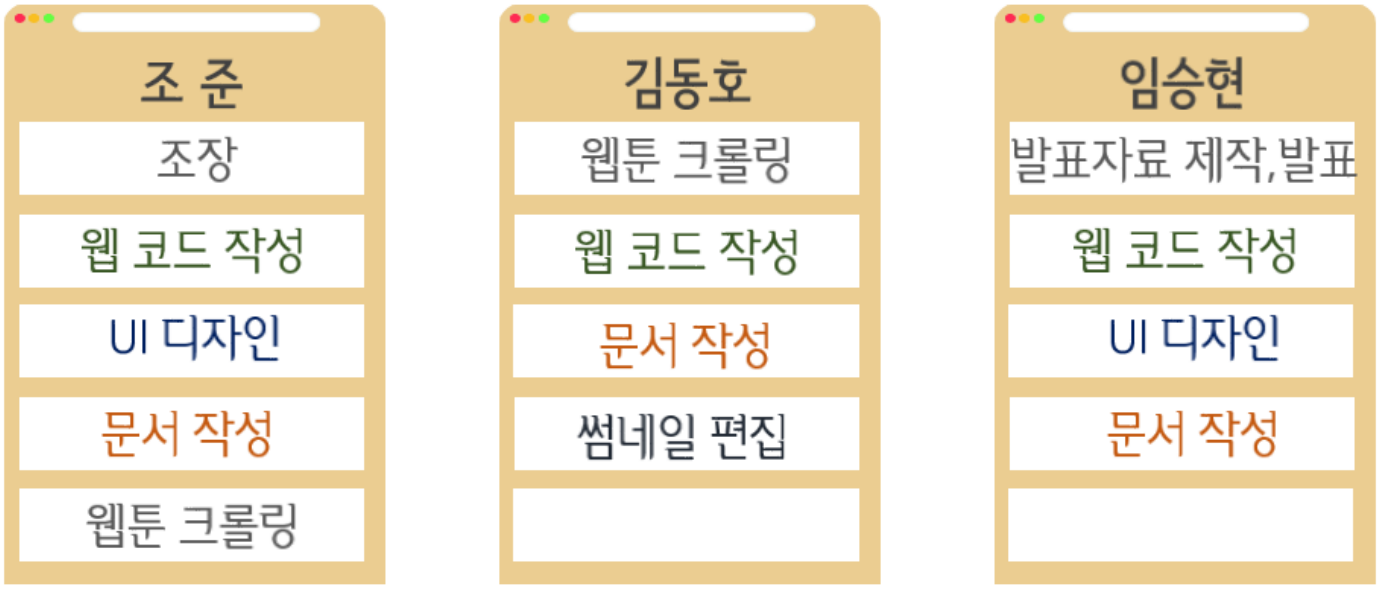
```
href="./cs.html"  // 현재 페이지와 동일한 경로에 있는 cs.html 파일로 이동하는 링크
function MailSend() {  //이메일 ID와 도메인을 검사하여 유효성을 확인한 후 처리
    var fm = document.getElementById("bug-report-form");
    var emailID = fm.elements["EmailID"].value;
    var emailDomain = fm.elements["EmailDomain"].value;

    if (emailID === "" || emailDomain === "") {
        alert("사용자 이메일을 정확하게 입력해주세요.");
        fm.elements["EmailID"].focus();
        return false;
    } else {
        var strEmail = emailID + "@" + emailDomain;

        if (!CheckEmail(strEmail)) {        // 생성된 이메일 주소가 유효하지 않은 경우
            alert("사용자 이메일을 정확하게 입력해주세요.");
            fm.elements["EmailID"].focus();
            return false;
        } else {

            alert("제출되었습니다. 문의하신 사항에 관해서는 이메일을 통해 답변드리겠습니다. 소통해 주셔서
감사합니다. "); // 성공 팝업 메시지 표시
            window.open(this.href); // 다른 페이지로 이동
            return false;
        }
    }
}
```

</td>
<td>입력된 이메일이 유효한지 확인하고 이를 제출 버튼이 클릭되었을 때도 다시 수행한뒤 문제가 없는 경우에만 성공 메시지를 표시한다.</td>
</tr>
</table>

	`` function CheckEmail(strEmail) { // 유효성 검사 var regDoNot = /(@.*@)|(\.\.)|(@\.)|(\.@)|(^\.)/; var regMust = /^[a-zA-Z0-9\-\._]+\@[a-zA-Z0-9\-\.]+\.([a-zA-Z]{2,3})$/; if (!regDoNot.test(strEmail) && regMust.test(strEmail)) // 유효성 검사를 통과한 경우 return true; else return false; } function SearchDomain(frm) { // 도메인 입력 필드에 입력된 값을 가져와서 해당 도메인을 선택하는 이메일 도메인 선택 필드를 설정 var emailDomain = frm.elements["EmailDomain"].value.toLowerCase(); // 도메인 입력필드의 값을 가져옴 var selEmailDomain = frm.elements["SelEmailDomain"]; // 도메인 선택 필드를 가져옴 if (emailDomain) { // 입력필드의 값이 존재하는경우 selEmailDomain.value = "user"; // 도메인 선택필드의 값을 user 로 함. } for (var i = 0; i < selEmailDomain.options.length; i++) { if (emailDomain === selEmailDomain.options[i].value) { // 입력필드의 값과 일치하는 도메인을 찾은 경우 selEmailDomain.value = emailDomain; // 이메일 도메인 선택 필드의 값을 일치하는 도메인으로 설정 if (emailDomain !== "user" && selEmailDomain.options[i].value !== "") { frm.elements["EmailDomain"].disabled = true; // 도메인 입력필드 비활성화 } break; } } } function domainChange(frm) { // 도메인 선택 필드에서 선택된 도메인에 따라 도메인 입력 필드를 제어함 with (frm) { var strSelDomain = SelEmailDomain[SelEmailDomain.selectedIndex].value; if (strSelDomain == "" || strSelDomain == "user") { //도메인 선택필드가 비어있거나 직접입력인 경우 EmailDomain.disabled = 0; EmailDomain.value = ""; EmailDomain.focus(); } else { EmailDomain.disabled = 1; // 입력필드를 비활성화 EmailDomain.value = strSelDomain; // 선택된 도메인으로 값을 설정. } } } ```	
카카오 웹툰 이동페이지	``` const rawURL = location.href const url = rawURL.slice(rawURL.indexOf('?') + 1, rawURL.length); console.log(url); var progressMeter = document.getElementById('progress-meter'); var duration = 3000; ```	타이머를 구현하여서 남은 시간을 기반으로 프로그레스 바의 width 를 업데이트하고 타임아웃이 발생시 전달 받은 카카오 웹툰 URL 로 이동한다.

```
var endTime = Date.now() + duration;


function animateProgressBar ()
{
  var currentTime = Date.now();
  var remainingTime = Math.max(0, endTime - currentTime);
  var progress = 1 - (remainingTime / duration);
  var currentWidth = (1 - progress) * 100 + '%';


  //width를 업데이트
  progressMeter.style.width = currentWidth;


  if (remainingTime > 0) {
    requestAnimationFrame(animateProgressBar);
  } else {
    progressMeter.style.width = '0%';
  }
}
requestAnimationFrame(animateProgressBar);
function startCountdown()
{
    var countdownElement = document.getElementById('countdown');
    countdownElement.innerHTML = countdown;


    if (countdown > 0)
    {
        countdown--;
        setTimeout(startCountdown, 1000);
    }
    else
    {
        window.location.href = url;
    }
}
var countdown = 3;
setTimeout(startCountdown(),100);
```

5. 팀 구성 및 역할 분담



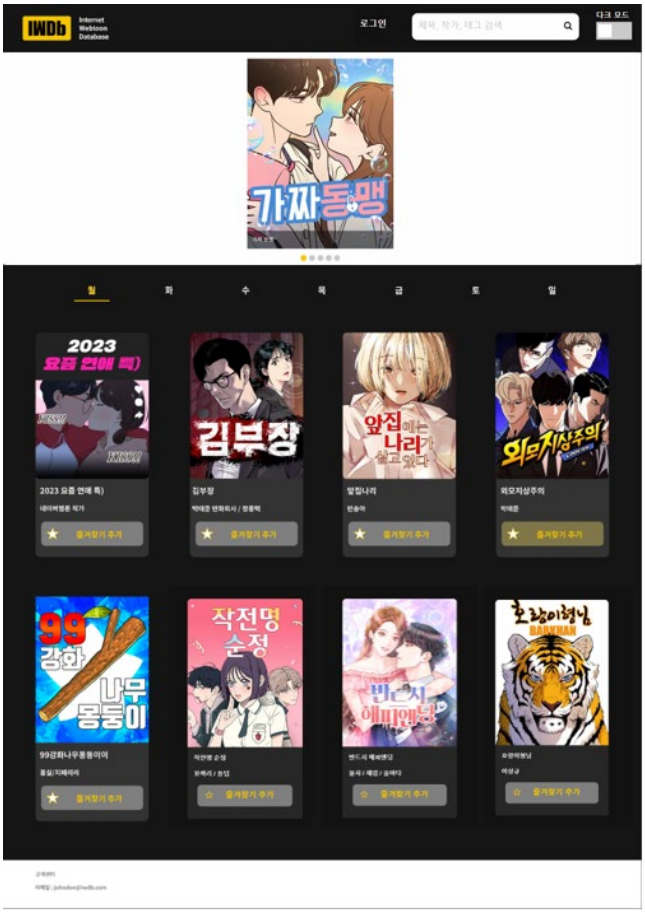
- 웹 코드 작성 역할 분담

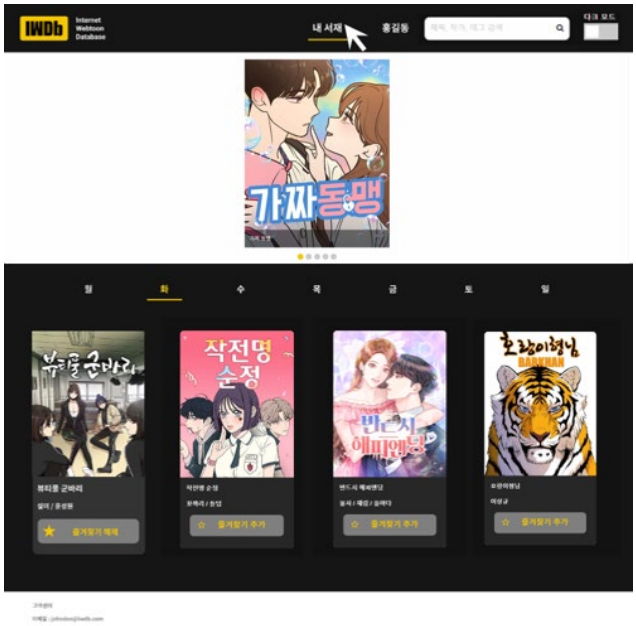
	app		메인			회원가입			즐겨찾기			감상 기록			책갈피			로그인		
	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS
김동호																				
임승현																				
조준																				

	회원정보			검색			카카오 이동페이지			아이디 찾기			비밀번호 찾기			문의 및 버그신고			고객센터	
	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS	JS	HTML	CSS
김동호																				
임승현																				
조준																				

6. 홈페이지 화면 캡쳐 설명

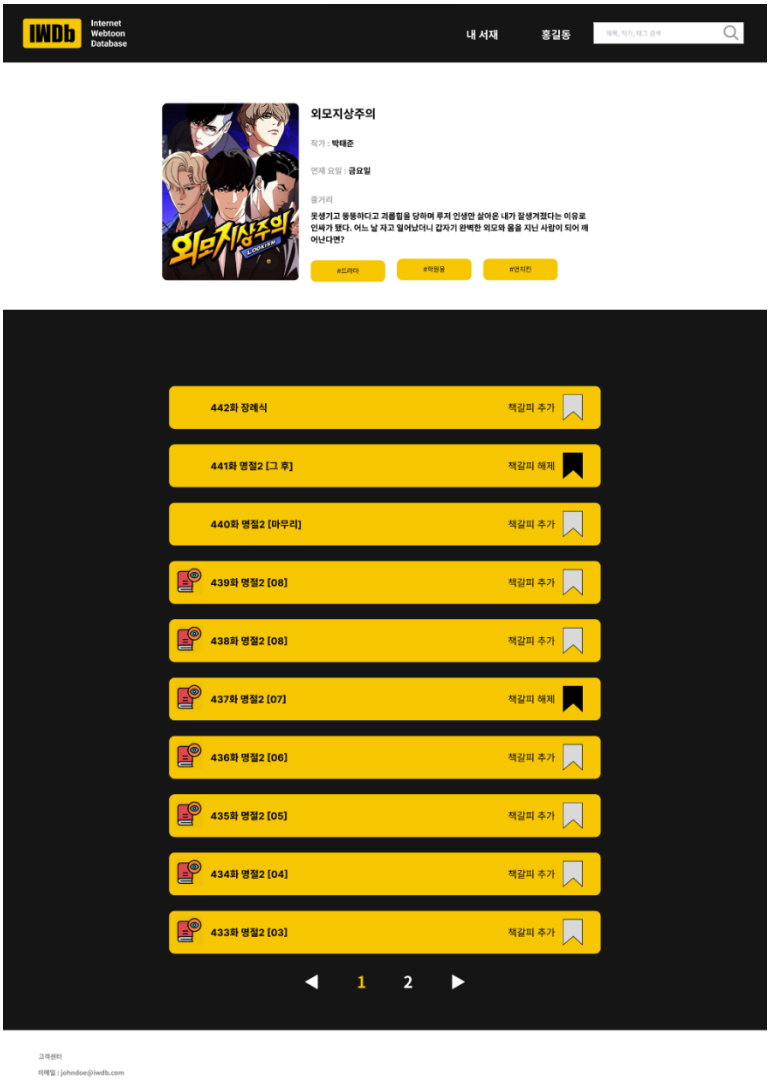
1) 메인 페이지 [좌 : 로그인 전, 우 : 로그인 후]





웹사이트의 첫 화면이다. 특징은 다음과 같다.

- 맨 위의 검은색 영역은 고정되어 있어 스크롤을 내려도 항상 위에 위치한다.
- IWDb 로고를 클릭하면 메인 페이지로 연결된다.
- 로그인 글자에 호버링하면 아래에 노란색 밑줄이 생겨나며 클릭하면 로그인 페이지로 연결된다.
- 검색 폼에 있는 돋보기 아이콘을 호버링하면 노란색으로 변하고 클릭하면 검색 페이지로 연결된다.
- 하얀색 영역에는 Carousel Slider가 존재하며 웹툰 썸네일이 나타난다. 썸네일에 커서를 얹으면 살짝 확대된다.
- Carousel Slider는 3초 마다 자동으로 돌아간다. 커서를 얹고 있는 동안에는 자동 넘김 기능이 중지된다.
- Carousel Slider를 클릭하면 해당 웹툰 정보가 나타난다.
- 그 아래 검은색 영역에는 요일별로 웹툰 카드들이 보이며, 접속한 날짜의 요일이 기본값으로 설정되어 있다. 선택된 요일은 노란색 글자와 노란색 밑줄이 그어져 있다. 또한 요일별 글자에 호버링이 되면 노란색 밑줄이 그어진다.
- 웹툰 카드들을 클릭하면 해당 웹툰 정보가 나타난다.
- 카드별로 즐겨찾기 추가 버튼이 존재하며 해당 버튼에 호버링하면 버튼 색이 회색에서 노란색으로 변경된다
- 비로그인 시: 즐겨찾기 추가 버튼을 클릭하면 로그인하라는 알림이 뜬다.
- 맨 아래에 위치한 고객센터 글자를 클릭하면 고객센터 페이지로 이동한다.
- Carousel Slider의 사진이나 웹툰 카드의 썸네일에 호버링하면 이미지가 약간 확대된다.
- 웹툰 카드를 누르면 솟아오르는 애니메이션과 함께 해당 웹툰 정보가 나타난다.



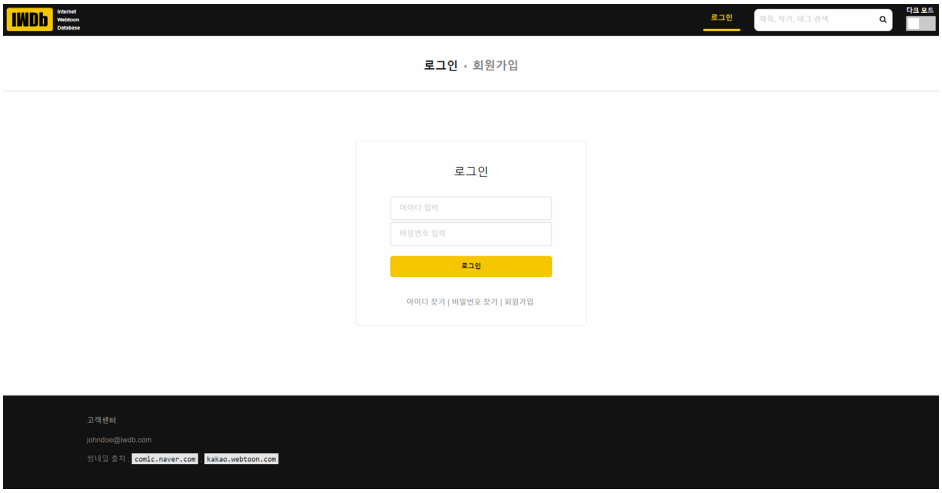
웹툰 상세 정보 페이지

- 한번 클릭된 에피소드 카드의 좌측에 감상 기록 아이콘이 생겨난다.
- 비로그인 시: 책갈피 추가 버튼이 비활성화 되어있다. (로그인하라는 알림이 뜬다.)
- 로그인 시: 책갈피 추가 버튼을 클릭한 경우 검은색으로 칠해지며 책갈피 해제로 글자가 변경된다. 또한 기존에 책갈피를 추가한 에피소드는 웹툰 정보 페이지가 로딩될 시 책갈피가 이미 추가되어 있음을 확인할 수 있다.
- 한번 클릭된 에피소드 카드의 좌측에 감상 기록 아이콘이 생겨난다.
- 한 페이지에 최대 10개의 에피소드 카드가 보여지며 아래의 화살표 버튼이나 숫자를 통해 페이지를 이동할 수 있다.
- 태그 버튼을 클릭하면 해당 태그를 검색한 검색 페이지가 새 탭에 열린다.

로그인 후 달라지는 특징은 다음과 같다.

- 로그인 글자 옆에 내 서재 항목이 생겨나며 호버링하면 즐겨찾기, 감상기록, 책갈피 탭이 나타난다. “내 서재” 클릭 시 내 서재 페이지로 이동한다.
- 로그인 글자는 닉네임으로 바뀌며 호버링하면 회원정보, 로그아웃 탭이 나타난다. 클릭 시 회원정보 페이지로 이동한다.
- 즐겨찾기 추가 버튼을 클릭하면 별의 내부가 노란색으로 채워지고 즐겨찾기 해제 버튼으로 변경되며 이를 다시 클릭하면 본래의 즐겨찾기 추가 버튼으로 돌아온다. 즐겨찾기 추가 시 즐겨찾기 탭에 해당 웹툰이 저장된다.

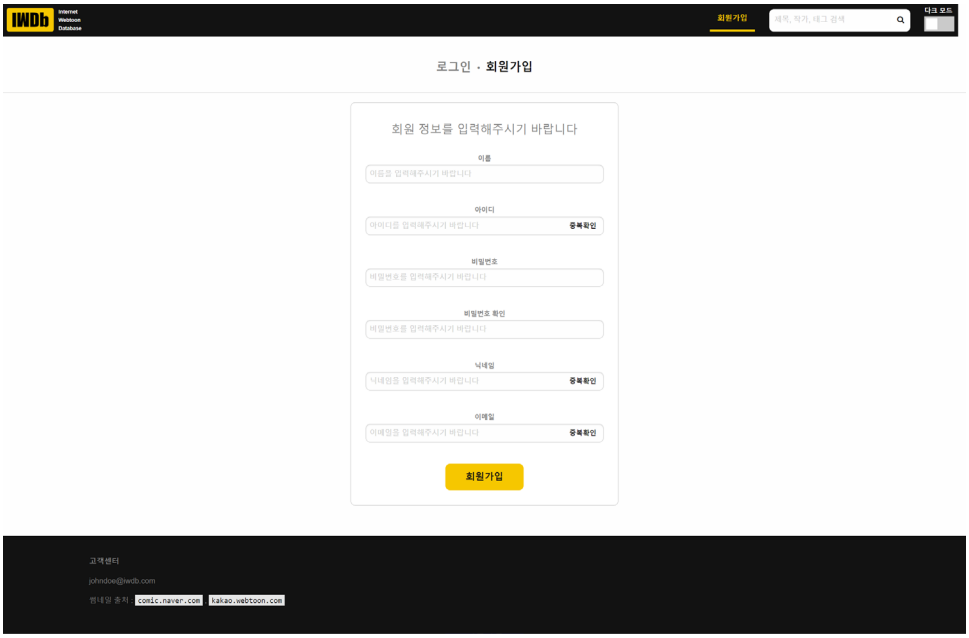
2) 로그인 페이지



특징은 다음과 같다.

- 우측 상단의 로그인 글자를 클릭하면 로그인 페이지로 연결된다.
- 화면 중간 지점에 위치한 "로그인 • 회원가입"에서 현재 페이지에 해당하는 글자는 검은색 볼드체로 표시되며 클릭이 불가능하고 다른 글자는 호버링 시 노란색으로 변하며 클릭 시 해당 페이지로 이동한다.
- 폼에 계정의 아이디와 비밀번호를 입력하여 유효한 입력이면 로그인 되어 메인 페이지로 연결된다.
- 아이디 찾기, 비밀번호 찾기 글자 클릭 시 해당 페이지가 팝업 된다.

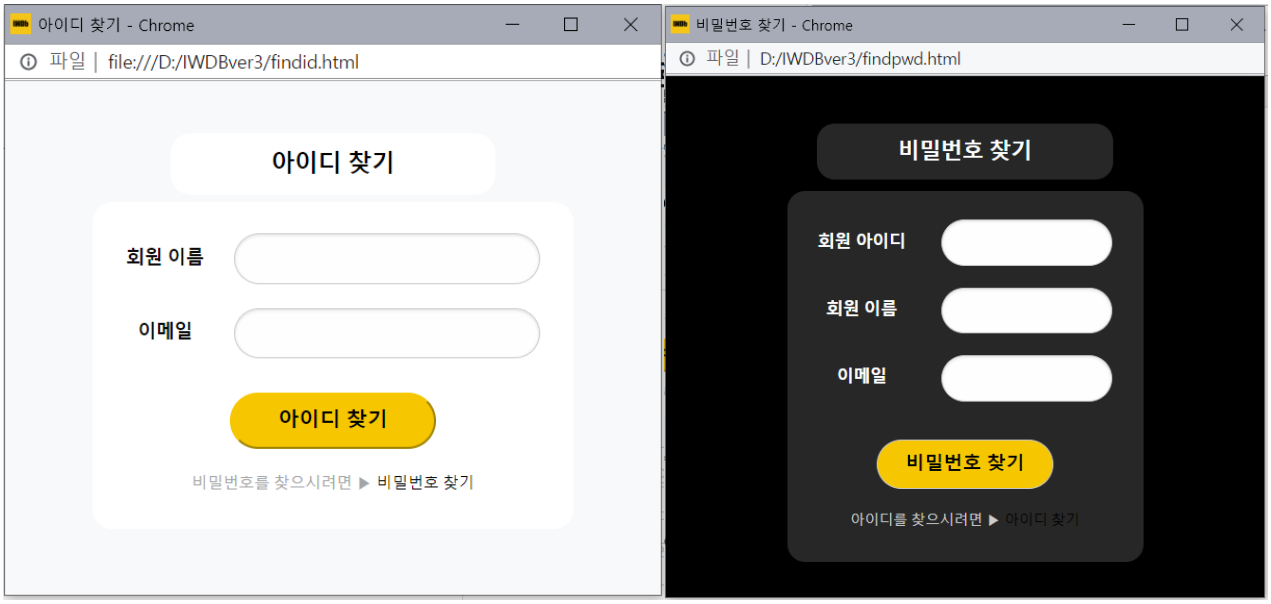
3) 회원가입 페이지



특징은 다음과 같다

- 기존에 존재하는 계정 정보와 중복확인을 진행하여 중복되지 않아야만 하며, 첫번째 비밀번호 폼과 두번째 비밀번호 폼의 값이 일치해야만 회원가입 버튼 클릭 시, 회원가입이 되었다는 알람이 뜨고 로그인 페이지로 연결된다.
- 우측 상단의 회원가입 글자를 클릭하면 회원가입 페이지로 이동한다.
- 세 개의 중복확인 버튼을 모두 눌러서 통과되어야만 회원가입이 가능하다. 중복 확인 버튼을 눌러 중복되지 않음이 확인된 뒤에 해당 폼에 적힌 내용의 변화가 생기면 다시 중복 확인을 받아야만 한다.
- 아이디는 영어와 숫자로 이뤄져야하고 이메일은 이메일 형식을 갖춰야 한다.

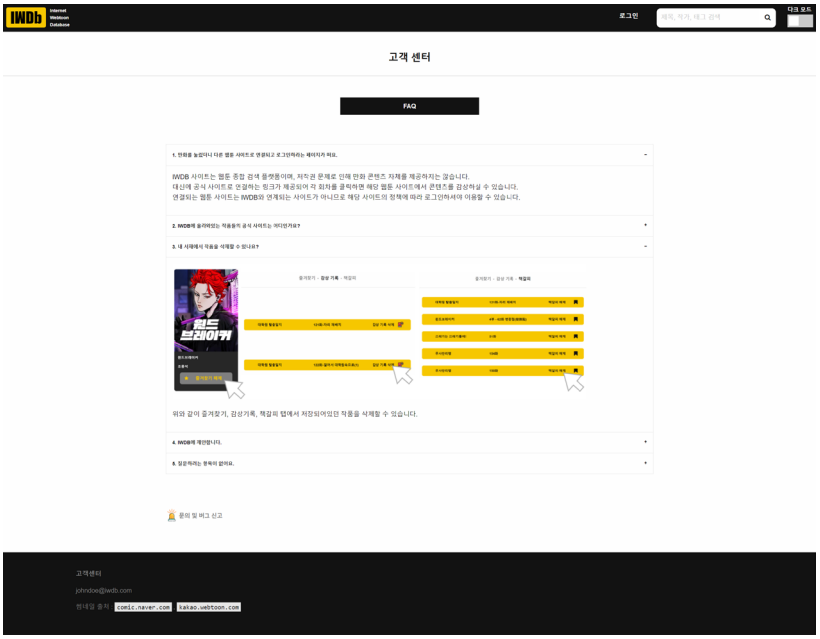
4) 아이디찾기 비밀번호찾기 페이지 [팝업]



특징은 다음과 같다

- 아이디 찾기: 기존에 존재하는 계정 정보와 일치하면 일부 가려진 아이디가 알람에 뜨고, 틀리면 잘못된 정보를 기입했다고 알람이 뜬다.
- 비밀번호 찾기: 기존에 존재하는 계정 정보와 일치하면 해당 이메일로 비밀번호 변경 메일이 갔다는 내용의 알람이 뜨고, 틀리면 잘못된 정보를 기입했다고 알람이 뜬다.
- 팝업창은 사이트에서 다크모드가 켜져 있는지의 여부를 반영하여 색깔이 바뀐다.

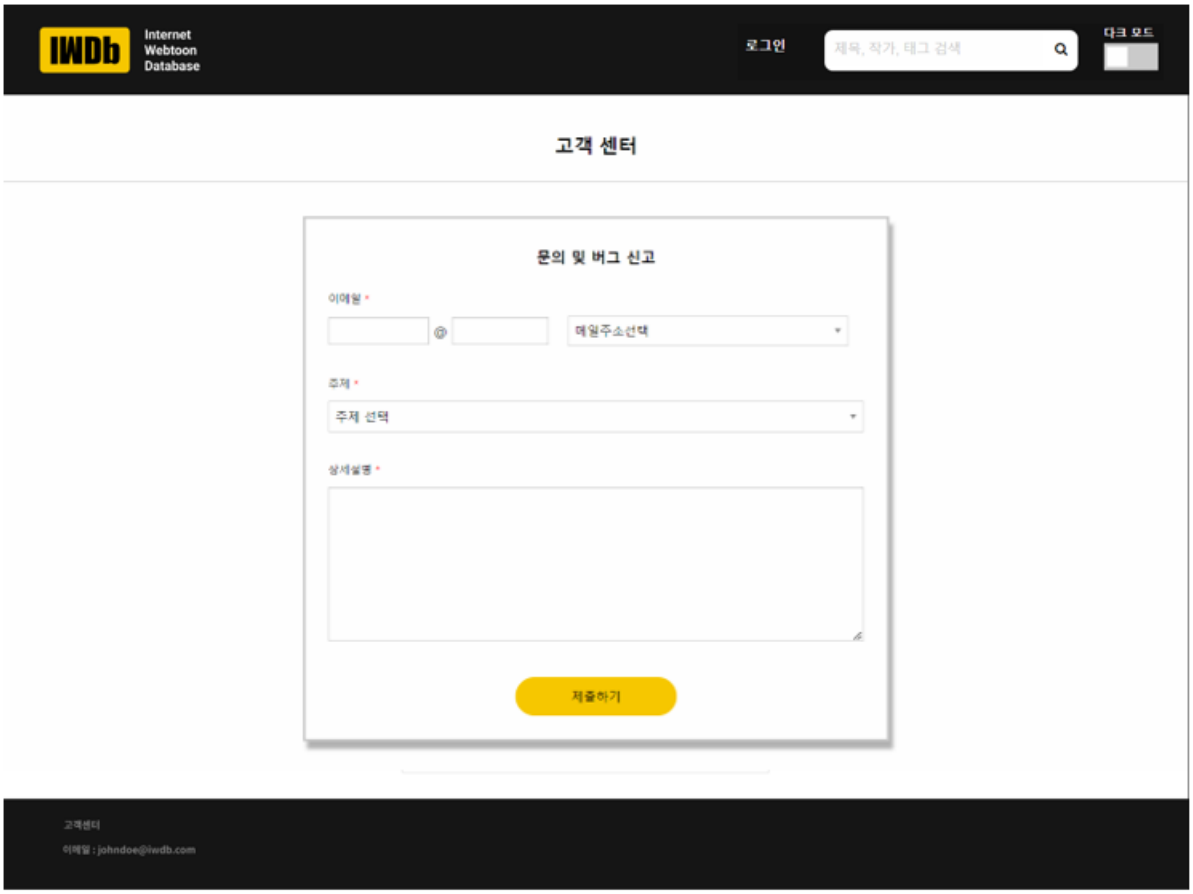
5) 고객센터 페이지



특징은 다음과 같다

- FAQ의 항목을 클릭하면 개별 답변이 슬라이드 다운되어 보여진다.
- 최소 하나 이상의 답변이 열려 있도록 강제되어 있다.
- 문의 및 버그 신고 글자를 클릭하면 문의 및 버그 신고 페이지로 연결된다.

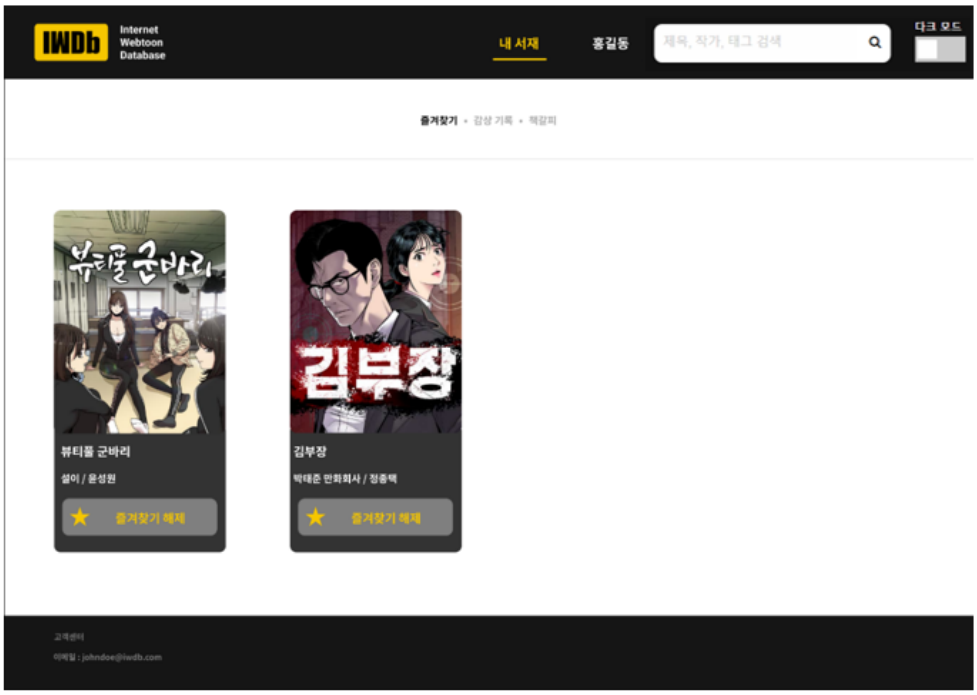
6) 고객센터 페이지 - 문의 및 버그신고



특징은 다음과 같다

- 이메일, 주제, 상세 설명을 모두 기입한 후 제출 버튼을 클릭해야만 올바르게 제출되었다는 알람이 뜨고 고객 센터 페이지로 연결된다.
- 이메일 도메인을 선택 항목 중 고를 수 있고 직접입력 선택 시 입력한 도메인이 형식에 맞아야 제출 가능하다.

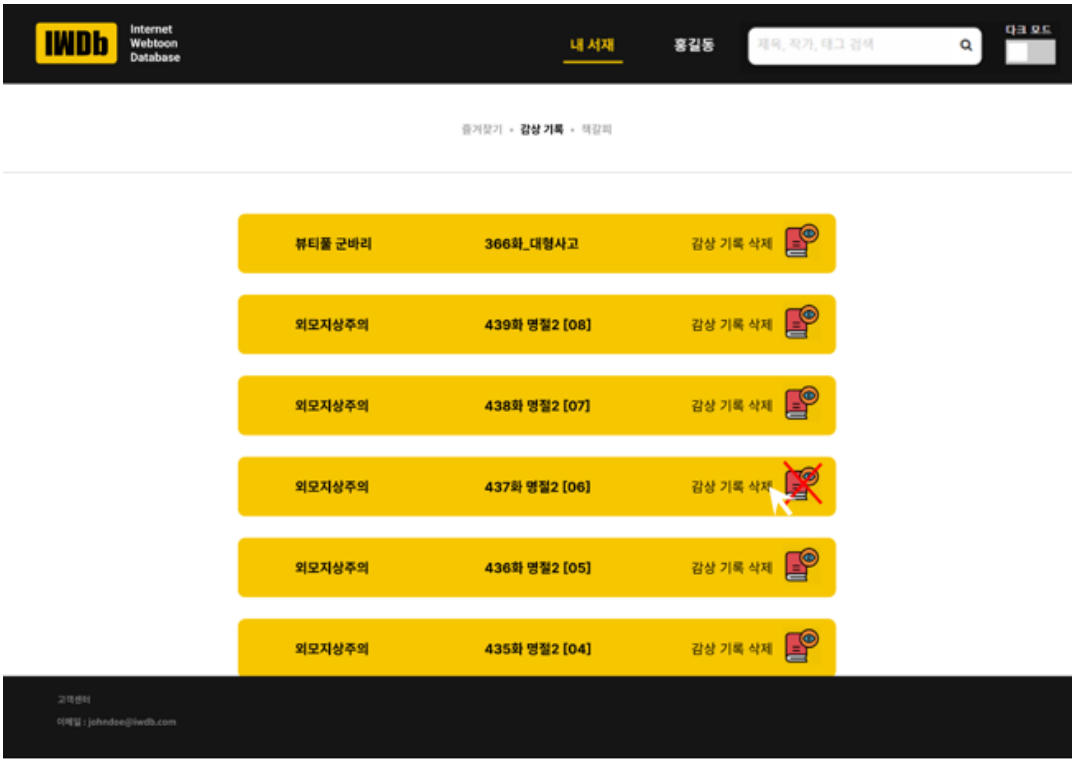
7) 내 서재 페이지 – (1) 즐겨찾기



내 서재 페이지는 즐겨찾기 페이지가 기본으로 설정되어 있다. 특징은 다음과 같다

- 화면 중간 지점에 위치한 "즐겨찾기 • 감상 기록 • 책갈피"에서 현재 페이지에 해당하는 글자는 검은색 볼드체로 표시되며 클릭이 불가능하고 다른 글자는 호버링 시 노란색으로 변하며 클릭 시 해당 페이지로 이동한다.
- 웹툰 카드를 클릭하면 해당 웹툰 정보 페이지로 이동한다.
- 즐겨찾기 해제 버튼을 누르면 해당 카드가 위로 올라가는 애니메이션과 함께 목록에서 사라진다.
- 로그인 후에만 접근 가능한 페이지이다.

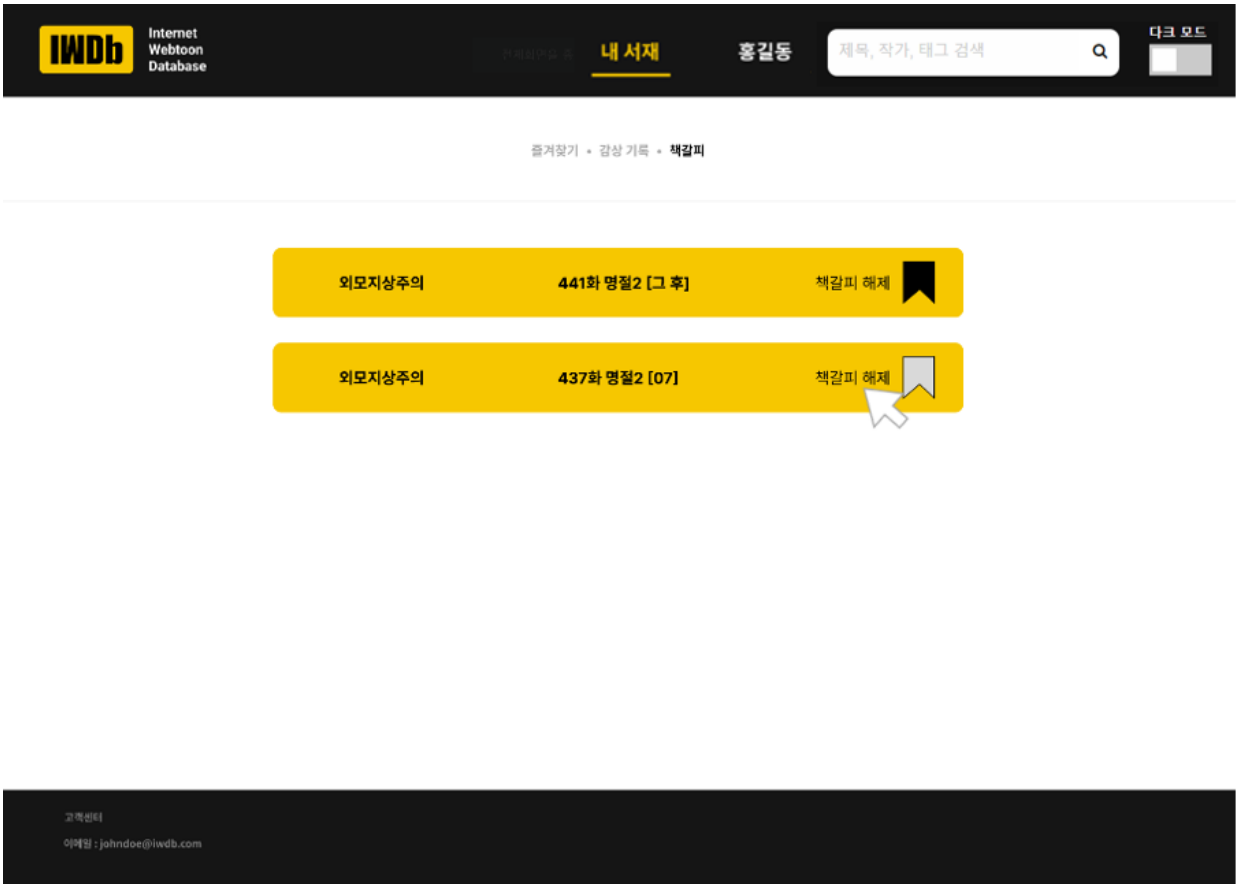
8) 내 서재 페이지 – (2) 감상기록



웹툰 정보 페이지에서 클릭된 에피소드 카드가 자동으로 저장되어있는 페이지이다. 특징은 다음과 같다.

- 웹툰 에피소드 카드 클릭, 해당 플랫폼의 에피소드 링크로 연결된다.
- 감상 기록 삭제 글자 및 아이콘에 호버링하면 붉은색 X자 표시가 생겨난다.
- 감상 기록 삭제 글자 및 아이콘을 클릭 시, 해당 카드가 우측으로 이동하는 애니메이션과 함께 목록에서 사라진다.

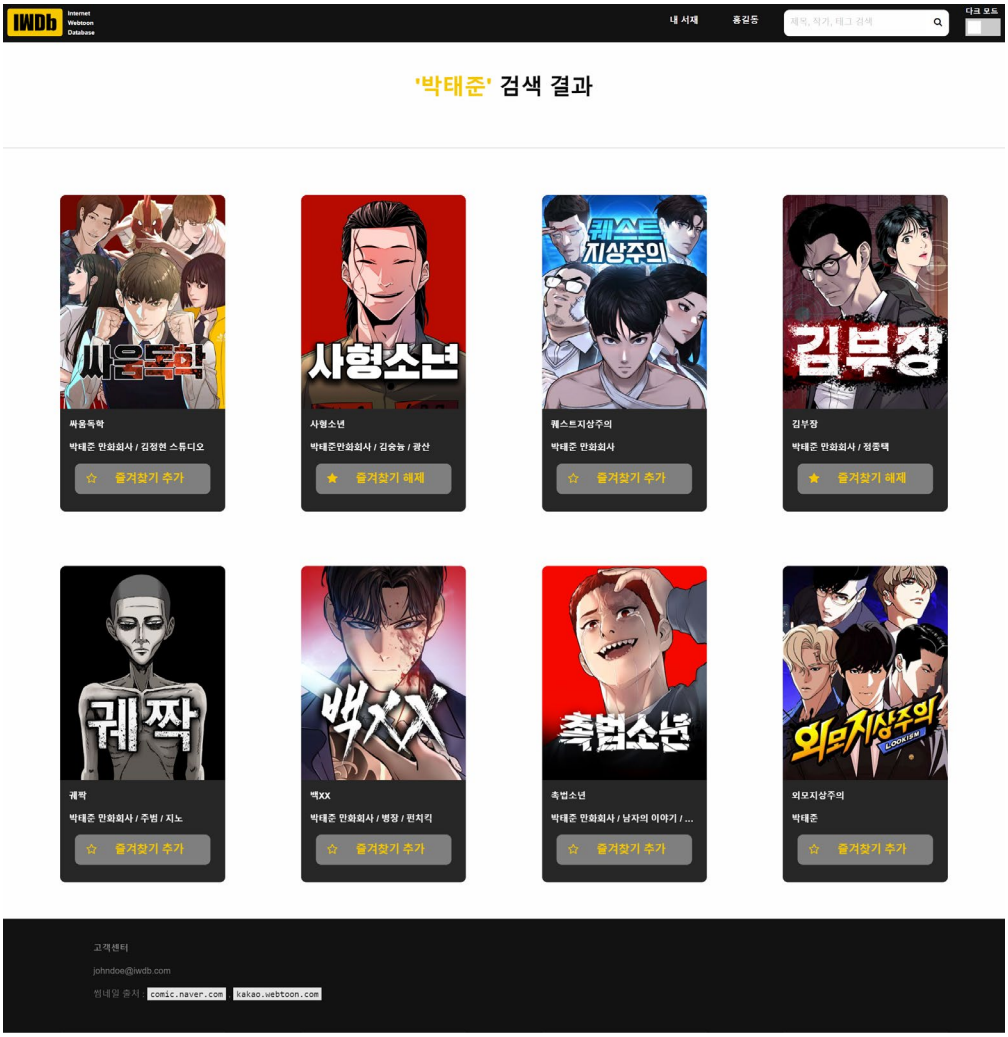
9) 내 서재 페이지 – (3) 책갈피



웹툰 정보 페이지에서 책갈피 추가를 한 에피소드 카드를 확인할 수 있다. 특징은 다음과 같다.

- 책갈피 해제 글자 및 아이콘에 호버링하면 아이콘 색상이 밝아진다.
- 책갈피 해제 글자 및 아이콘을 클릭 시, 해당 카드가 사라진다.
- 비로그인 시: 책갈피 추가 버튼이 비활성화 되어있다.
- 로그인 시: 책갈피 추가 버튼에 호버링하거나 클릭한 경우 검은색으로 칠해지며 책갈피 해제로 글자가 변경된다.

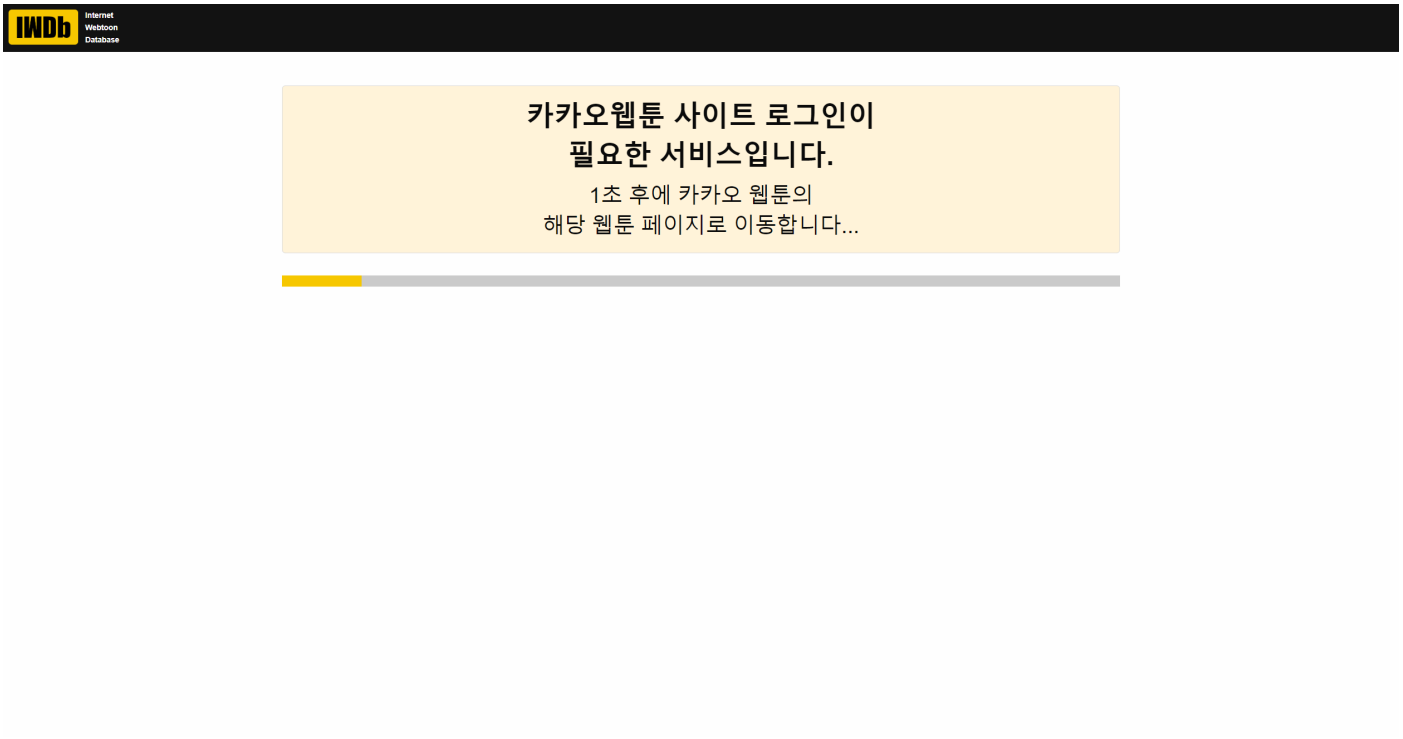
10) 검색 페이지



특징은 다음과 같다

- 메인 페이지와 마찬가지로 비로그인 시 즐겨찾기 추가 버튼 클릭하면 로그인 하라는 알림이 뜨고 로그인 시 제대로 작동한다.
- 웹툰 카드를 클릭하면 웹툰 정보 페이지가 보여진다.
- 제목, 작가, 태그 정보를 검색할 수 있다.
- 검색 키워드가 제목, 작가, 태그와 정확히 일치해야만 되는 것이 아니라 포함 관계여도 검색 가능하다.
- 검색된 웹툰 카드는 제목, 작가, 태그의 일치 순서이며, 그 항목이 중복되지 않는다. 예시) 제목과 태그에 모두 로맨스가 들어가는 웹툰이 존재하더라도 검색에서는 중복되어 나오지 않는다.

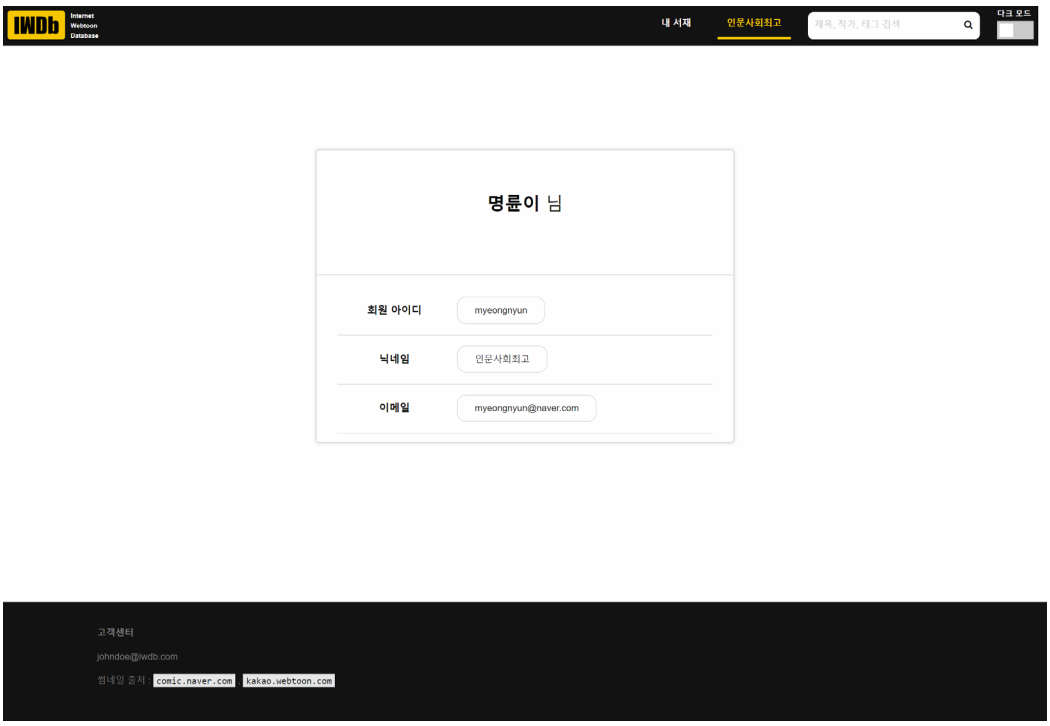
11) 카카오 웹툰 이동 페이지



카카오 웹툰으로 연결되기 전 안내 페이지이다. 특징은 다음과 같다

- 3초 후에 카카오 웹툰 플랫폼의 해당 카카오 웹툰 페이지로 연결된다.

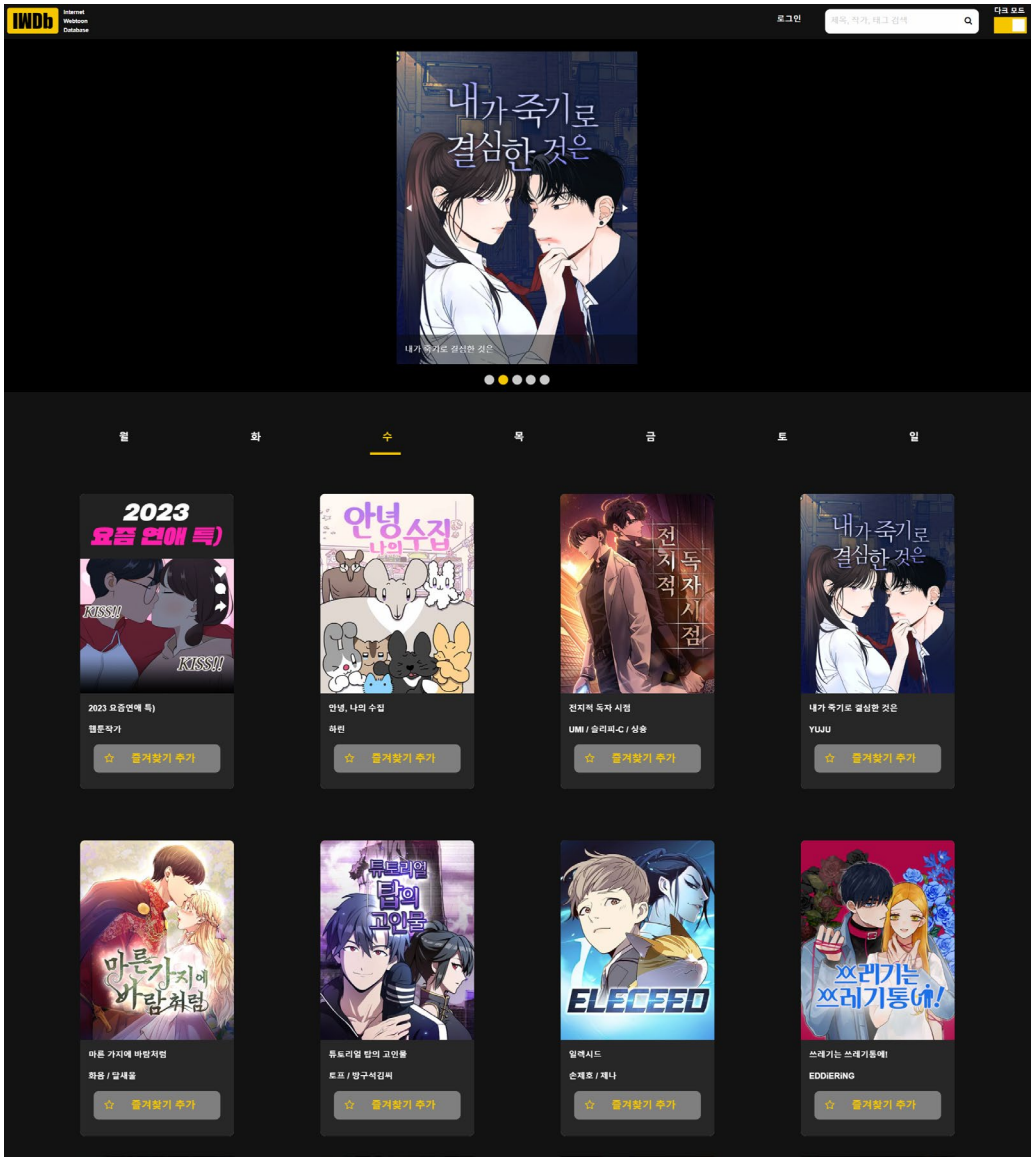
12) 회원 정보 페이지

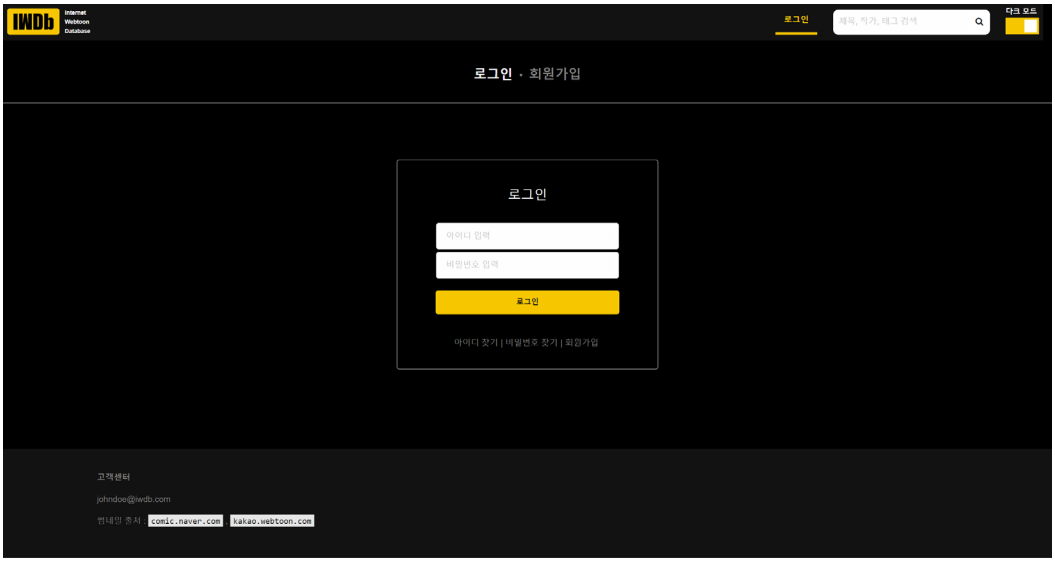


특징은 다음과 같다

- 로그인 후에만 나타나는 탭이다. 우측 상단 노란색 닉네임 혹은 회원정보 항목을 클릭하면 회원 정보 페이지로 연결된다.
- 로그인 된 회원의 이름, 아이디, 닉네임, 이메일 정보가 나타난다.

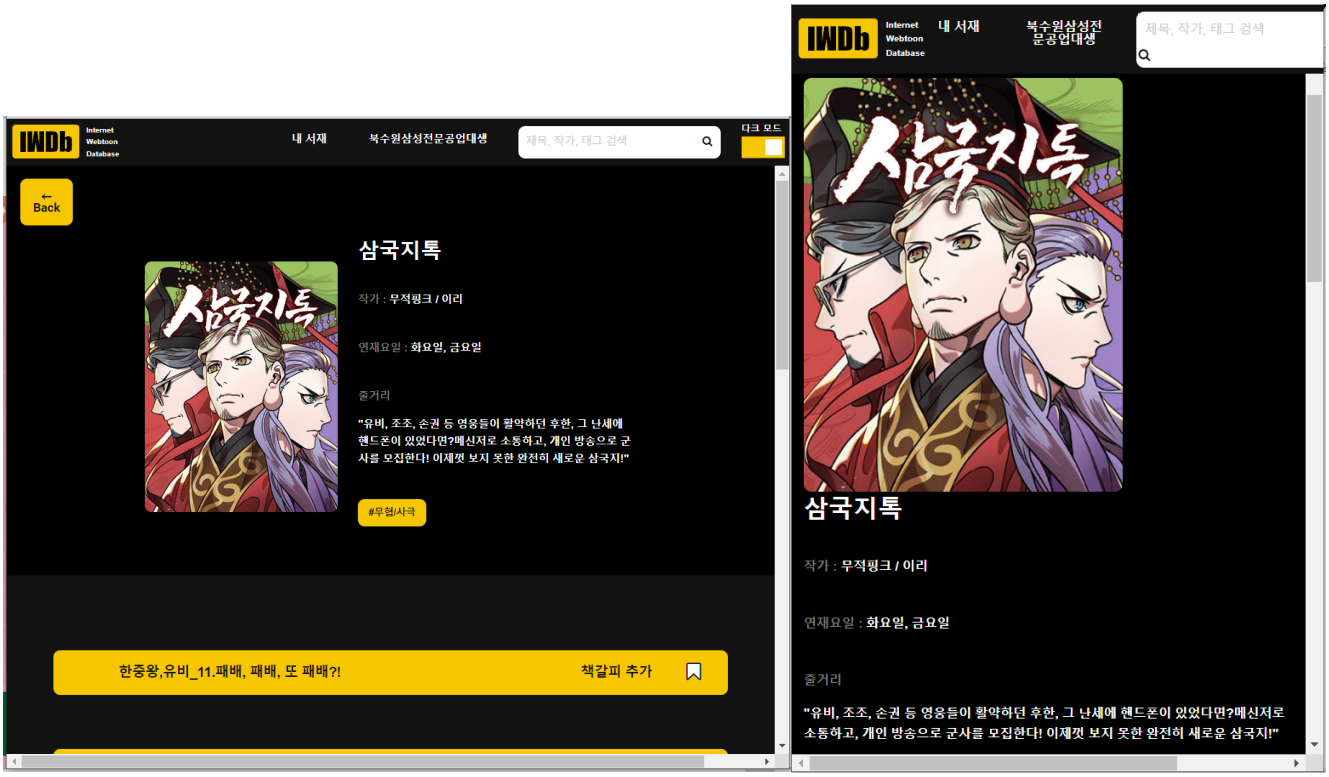
13) 다크모드



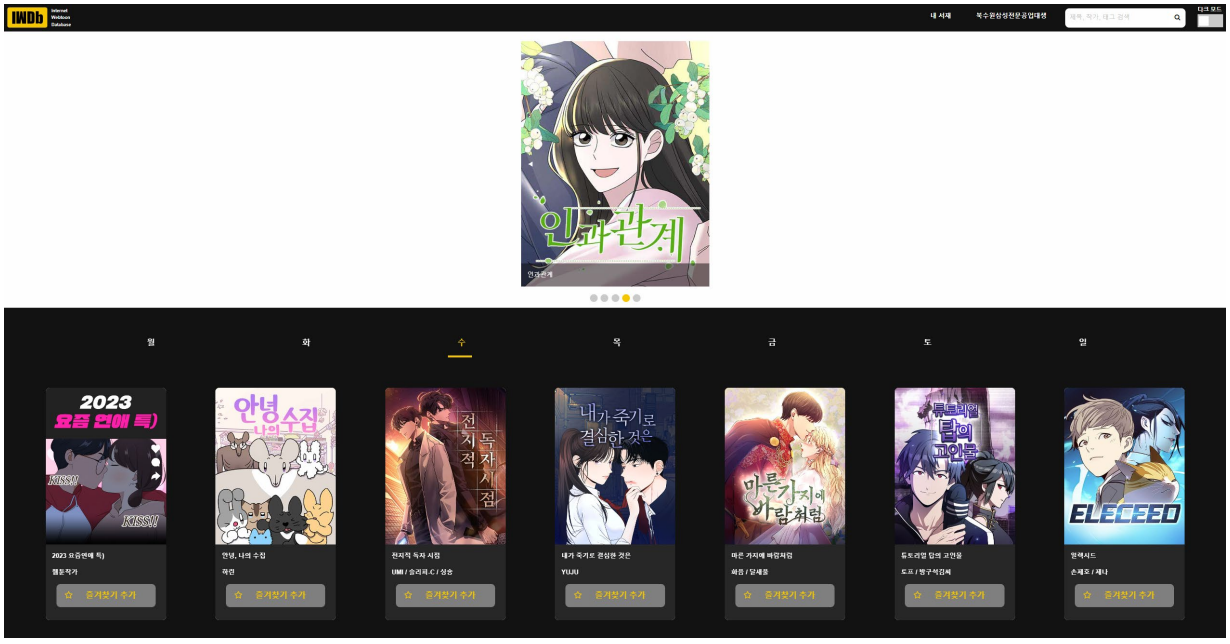


다크모드 토글 스위치를 누르면 배경이 어두워진다. 모든 페이지에 다크모드를 구현하였으며 특정 페이지에서 다크모드를 설정하면 이와 같은 설정이 다른 페이지에서도 유지된다.

14) Responsive Web



브라우저 스크린의 크기에 실시간으로 반응하여 UI의 레이아웃이 변하도록 하였다.



화면 크기에 맞추어 웹툰 카드의 열 개수가 변경된다.

15) 파비콘

IWDB 로고를 파비콘으로 설정하였다. PC, 안드로이드, 애플 모든 기기 환경을 대비해놓았다.

7. 개인 의견(option)