2조 최태원 이재민 이재훈 이승완

SeSAC MovieInfoAPP

TMDB API를 이용한 영화 정보 제공 서비스

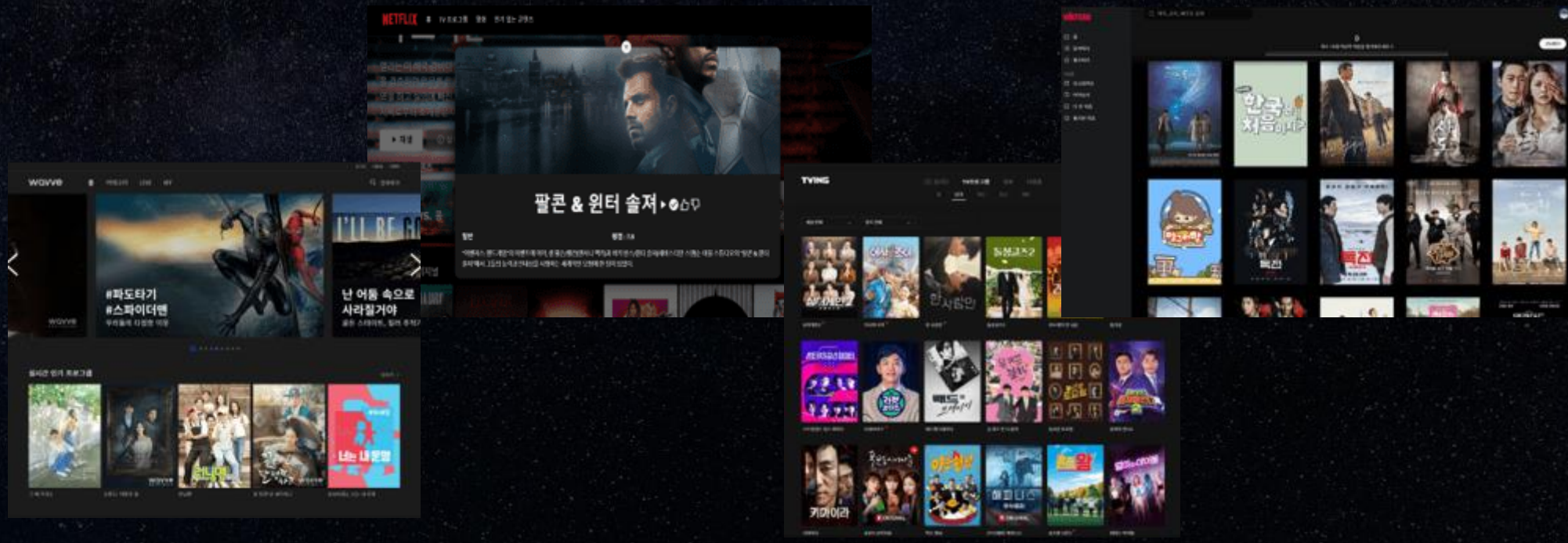
SeSSAC MovieInfoAPP

목차

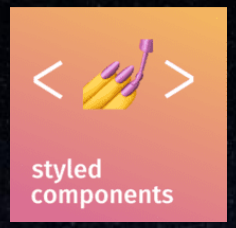
기획 의도

강의로 배운 내용을 토대로 현재 서비스 중인 국내외 OTT 웹사이트 직접 구현해 보면서

어떤 기술을 사용하고 어떻게 구현했는지 알아보고자 이 프로젝트를 기획함



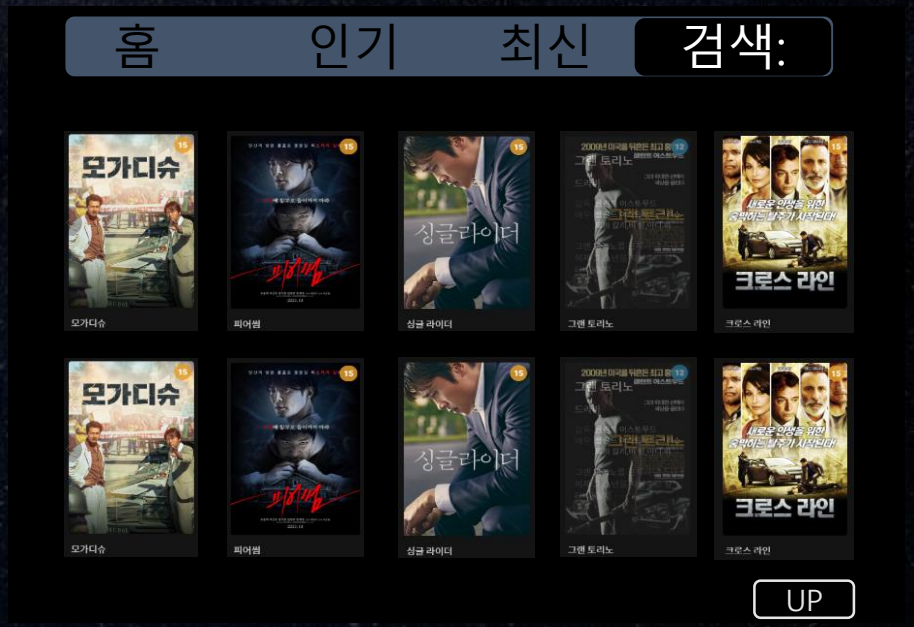
사용 기술 및 도구



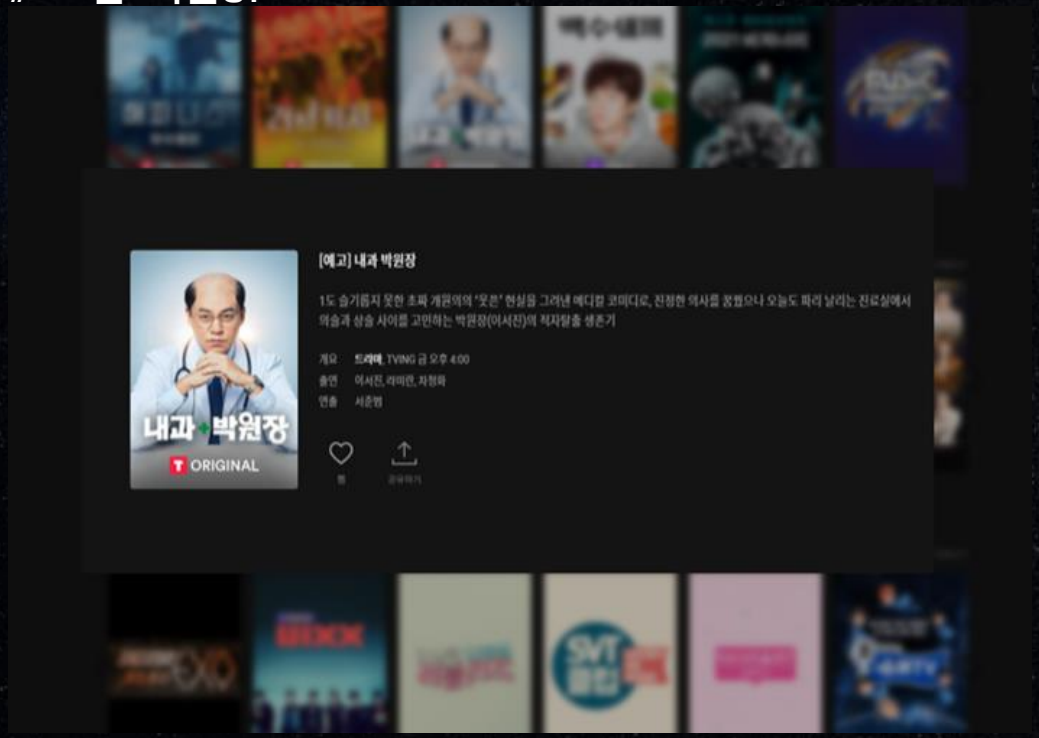
일정 계획

주요 화면 구성(UI 설계)

Main화면 UI



모달 화면UI



일정 계획

WBS

담당 기능

이재민 - 웹 기능 구현, API정보 받기, 화면 출력

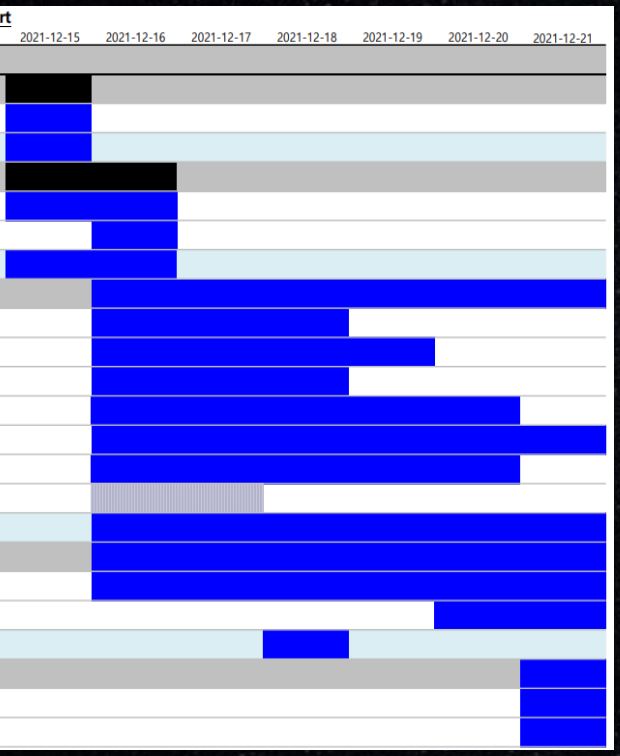
최태원 - 필터, 검색

이재훈 - 화면 출력,모달

이승완 - 버튼

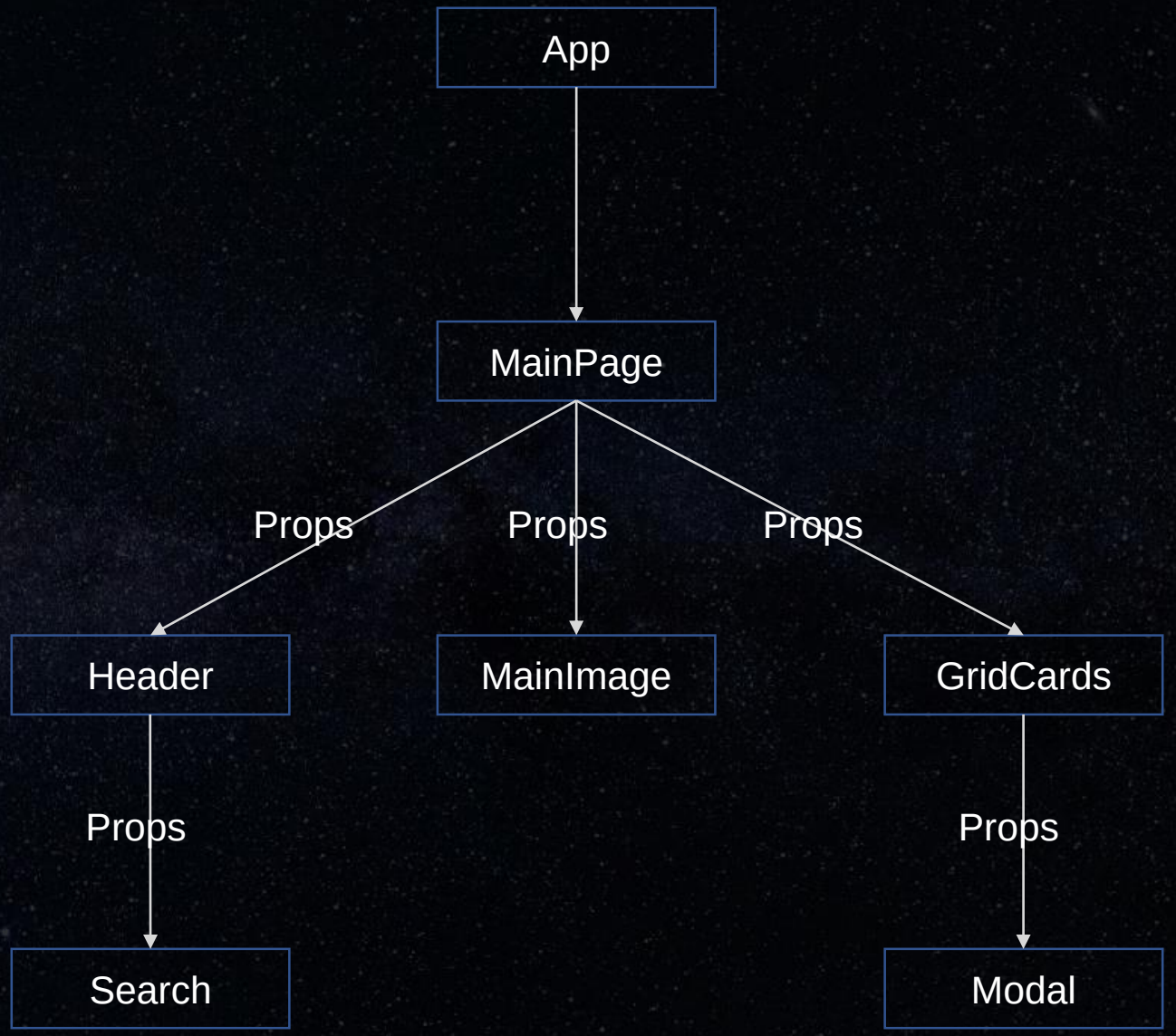
API를 이용한 영화 정보 제공 웹 서비스 개발 — **Gantt Chart**

WBS	테스크	작업자	상태	시작일	종료일	기간	진척도
1	**분석**	-	-	**2021-12-15**	**2021-12-15**	**1**	**100%**
1.1	요구사항 분석	전원	Completed	21-12-15	21-12-15	1	100%
1.2	진행 업무 점검 및 이슈 해결	전원	Milestone	21-12-15	21-12-15	1	100%
2	**설계**	-	-	**2021-12-15**	**2021-12-16**	**2**	**100%**
2.1	UI 설계	전원	Completed	21-12-15	21-12-16	2	100%
2.2	컴포넌트 설계	전원	Completed	21-12-16	21-12-16	1	100%
2.3	진행 업무 점검 및 이슈 해결	전원	Milestone	21-12-15	21-12-16	2	100%
3	**구현**	-	-	**2021-12-16**	**2021-12-21**	**4**	**100%**
3.1	웹 기능 구현	이재민	Completed	21-12-16	21-12-18	2	100%
3.1.1	API정보 받기	이재민, 최태원	Completed	21-12-16	21-12-19	2	100%
3.1.2	화면 출력	이재민, 이재훈	Completed	21-12-16	21-12-18	2	100%
3.1.3	필터	최태원	Completed	21-12-16	21-12-20	3	100%
3.1.4	검색	최태원	Completed	21-12-16	21-12-21	4	100%
3.1.5	모달	이재훈, 이재민	Completed	21-12-16	21-12-20	3	100%
3.1.6	버튼	이승완	Completed	21-12-16	21-12-17	2	100%
3.2	진행 업무 점검 및 이슈 해결	전원	Milestone	21-12-16	21-12-21	4	100%
4	**테스트**	-	-	**2021-12-16**	**2021-12-21**	**4**	**100%**
4.1	단위 테스트(웹)	전원	Not Started	21-12-16	21-12-21	4	100%
4.2	통합 테스트(웹)	전원	Not Started	21-12-20	21-12-21	2	100%
4.3	진행 업무 점검 및 이슈 해결	전원	Milestone	21-12-18	21-12-18	0	100%
5	**결과발표**	-	-	**2021-12-21**	**2021-12-21**	**1**	**100%**
5.1	최종 보고서 작성	전원	Not Started	21-12-21	21-12-21	1	100%
5.2	산출물 업로드	전원	Not Started	21-12-21	21-12-21	1	100%



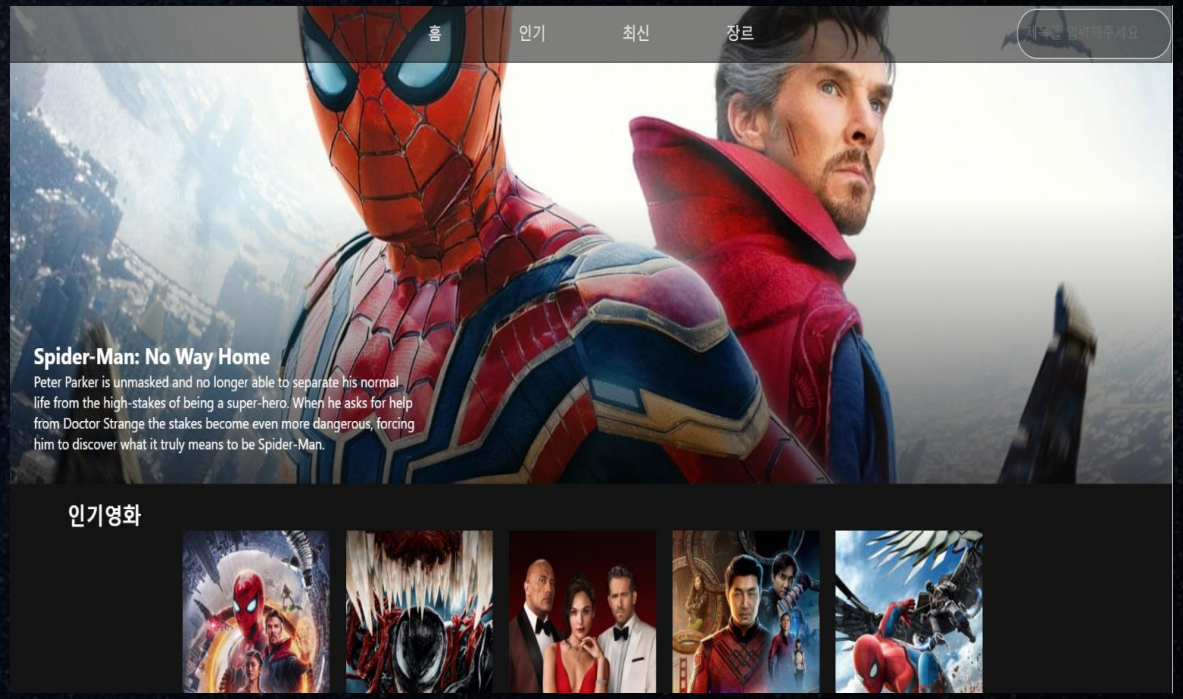
일정 계획

컴포넌트 설계

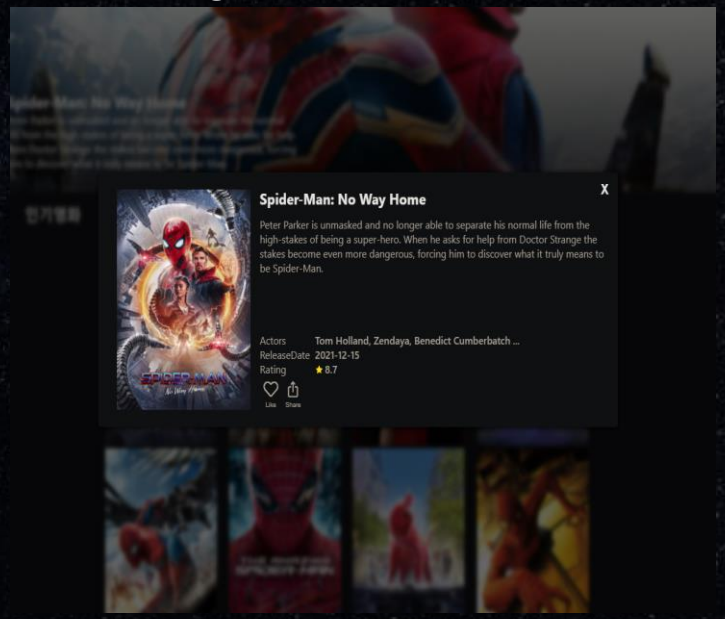

App
MainPage
Props
Props
Props
Header
MainImage
GridCards
Props
Props
Search
Modal

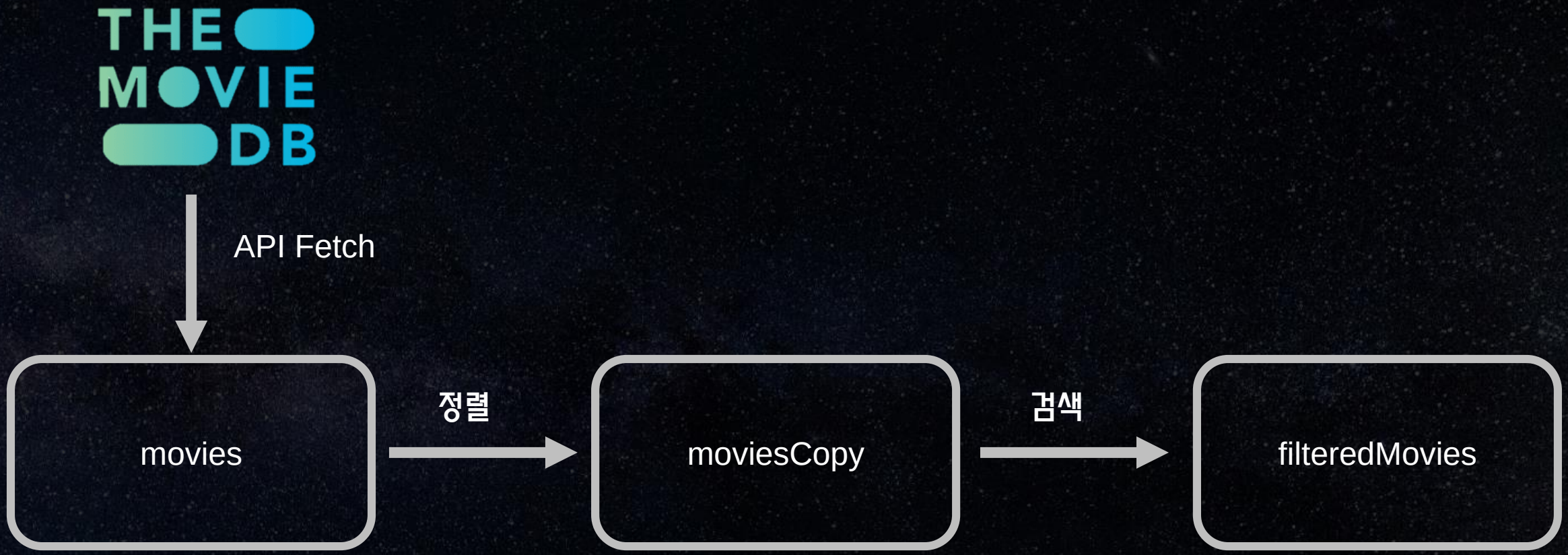
완성된 주요 화면

Main□ □ UI



□□ □□UI




THE MOVIE DB
API Fetch
movies
정렬
moviesCopy
검색
filteredMovies

Main화면 구조

Header

```
<Header
    containerRef={containerRef}
    nameChange={nameChange}
    filter={filter}
    movies={movies}
    moviesCopy={moviesCopy}
    setMoviesCopy={setMoviesCopy}
    setFilteredMovies={setFilteredMovies}
    reset={reset}
/>
```

Body

```
{mainMovieImage && (
    <MainImage
        image={`${IMAGE_URL}w1280${mainMovieImage.backdrop_path}`}
        title={mainMovieImage.original_title}
        overview={mainMovieImage.overview}
    />
)}
```

MainImage

```
<MovieCardBlock>
    <h2 className="text">{`${name[number]}영화`}</h2>
    {/* Movie Grid Cards */}
    <MovieCardBody ref={containerRef}>
        {/* <MovieCardBody> */}
        {filteredMovies &&
            filteredMovies.map((movie, index) =>
                index < page ? (
                    <div key={index}>
                        <GridCards
                            posterPath={
                                movie.poster_path
                                    ? `${IMAGE_URL}w500${movie.poster_path}`
                                    : null
                            }
                            movie={movie}
                            modalPosterPath={
                                movie.poster_path
                                    ? `${IMAGE_URL}original${movie.poster_path}`
                                    : null
                            }
                        />
                    </div>
                ) : (
                    ''
                )
            )}
    </MovieCardBody>
</MovieCardBlock>
```

API관련 정보 상수로 선언

```
1   //TMDB api를 상수로 선언
2   export const API_URL = 'https://api.themoviedb.org/3/';
3   export const API_KEY = [redacted]
4   export const IMAGE_URL = 'http://image.tmdb.org/t/p/';
```

TMDB API Fetch

```
// 최초 브라우저 redering되었을 때 실행
useEffect(() => {
    const response = `${API_URL}movie/popular?api_key=${API_KEY}&language=en-US&page=`;
    callMovie(response);
}, []);

const callMovie = async response => {
    let movieList = [];
    for (let i = 1; i < 11; i++) {
        movieList = movieList.concat(
            await fetch(response + i)
                .then(response => response.json())
                .then(response => response.results)
        );
    }
    setMovies(movieList.concat());
    setMainMovieImage(movieList[0]);
};
```

##useRef로 특정 DOM 선택

```
// body, div 위로 이동
const goTop = () => {
    // div
    containerRef.current.scrollTo(0, 0);

    // 전역객체
    window.scrollTo({
        top: 0,
        behavior: 'smooth',
    });
    setScrollY(0);
    setBtnStatus(false); // invisible
};
```

CSS in JS 기술 사용하기 위해 styled-components 라이브러리를 사용

```
import styled from 'styled-components';

const MainPageBlock = styled.div`
    width: 100%;
    margin: 0px;
`;

const MovieCardBlock = styled.div`
    width: 90%;
    /* height: ; */
    margin: 1rem auto;
    .text {
        color: white;
    }
    box-sizing: border-box;
`;

const MovieCardBody = styled.div`
    height: 100vh;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin: 0 auto;
    width: 90%;
    box-sizing: border-box;
    /* 이미지가 div영역 밖으로 나가는거 방지 */
    overflow: auto;
    /* overflow의 스크롤바 없애고 기능은 살림*/
    -ms-overflow-style: none;
    ::-webkit-scrollbar {
        display: none;
    }
`;
```

인기순 정렬

```
setMoviesCopy(
  nextMovies.sort((a, b) => (a.popularity > b.popularity ? -1 : 1))
);
```

최신순 정렬

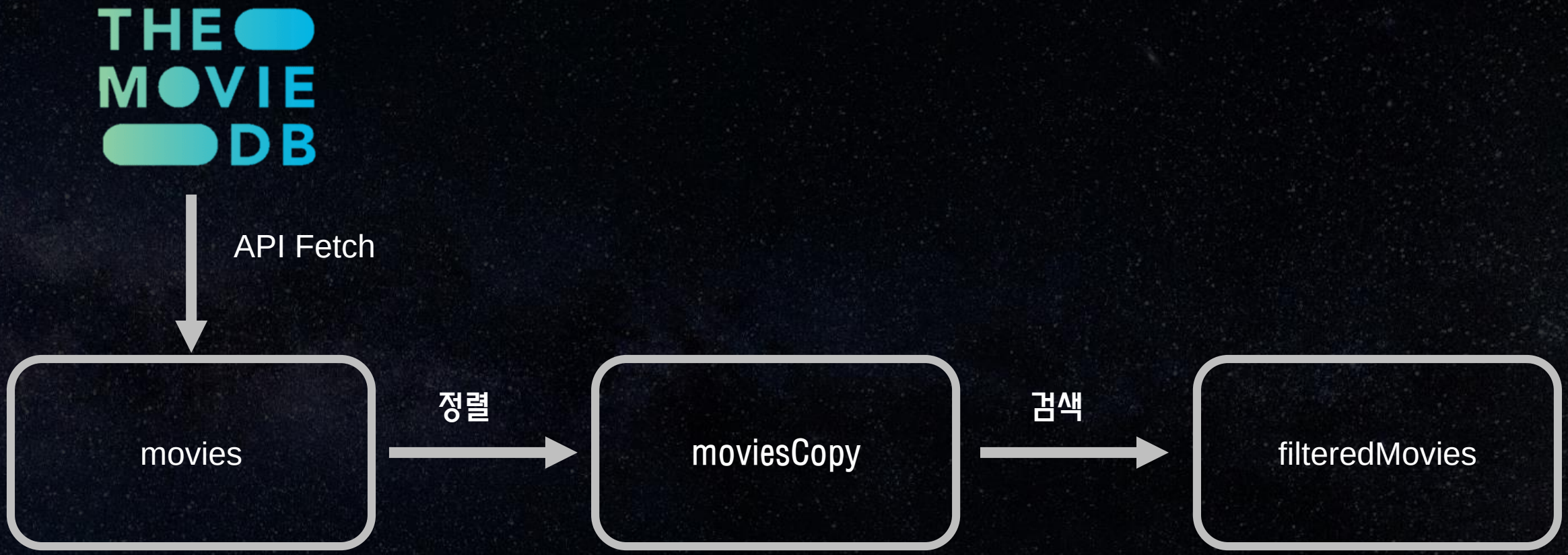
```
setMoviesCopy(
  nextMovies.sort((a, b) =>
    new Date(a.release_date) > new Date(b.release_date) ? -1 : 1
  )
);
```

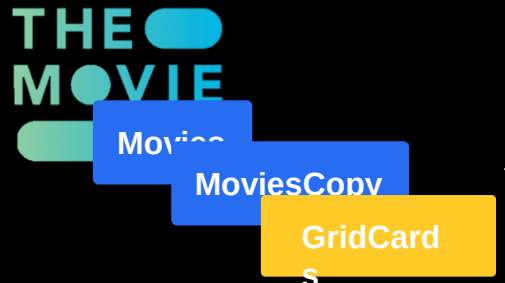
장르별 정렬

```
const genreNo = parseInt(id);
setMoviesCopy(
  nextMovies.filter((movie) => movie.genre_ids.includes(genreNo))
);
```

검색

```
useEffect(() => {
  const queryText = value.toLowerCase().replaceAll(" ", "");
  if (queryText.trim() == "") {
    reset();
    return;
  }
  setFilteredMovies(
    moviesCopy
      .concat()
      .filter((movie) =>
        movie.title.toLowerCase().replaceAll(" ", "").includes(queryText)
      )
  );
}, [value]);
```


THE MOVIE DB
API Fetch
movies
정렬
moviesCopy
검색
filteredMovies



Data


```
[0 … 99]
▼0:
    adult: false
    backdrop_path: "/V1Ht27nCqOuTnuX6bku8QZapzO.jpg"
  ▶ genre_ids: (3) [28, 12, 878]
    id: 634649
    original_language: "en"
    original_title: "Spider-Man: No Way Home"
    overview: "Peter Parker is unmasked and no longer able to separate his normal life from the high-stakes of being a super-hero. W…
    popularity: 21848.81
    poster_path: "/1g0dhYtq4irTY1GPXvft6k4YLjm.jpg"
    release_date: "2021-12-15"
    title: "Spider-Man: No Way Home"
    video: false
    vote_average: 8.7
    vote_count: 1414
  ▶ [[Prototype]]: Object
▶1: {adult: false, backdrop_path: '/eENEf62tMXDhyVvdcX1nQz2wcuT.jpg', genre_ids: Array(3), id: 580489, original_language: 'en', …}
▶2: {adult: false, backdrop_path: '/7ajHGIAYNMiIzejy1LJWdPrcAx8.jpg', genre_ids: Array(4), id: 512195, original_language: 'en', …}
```

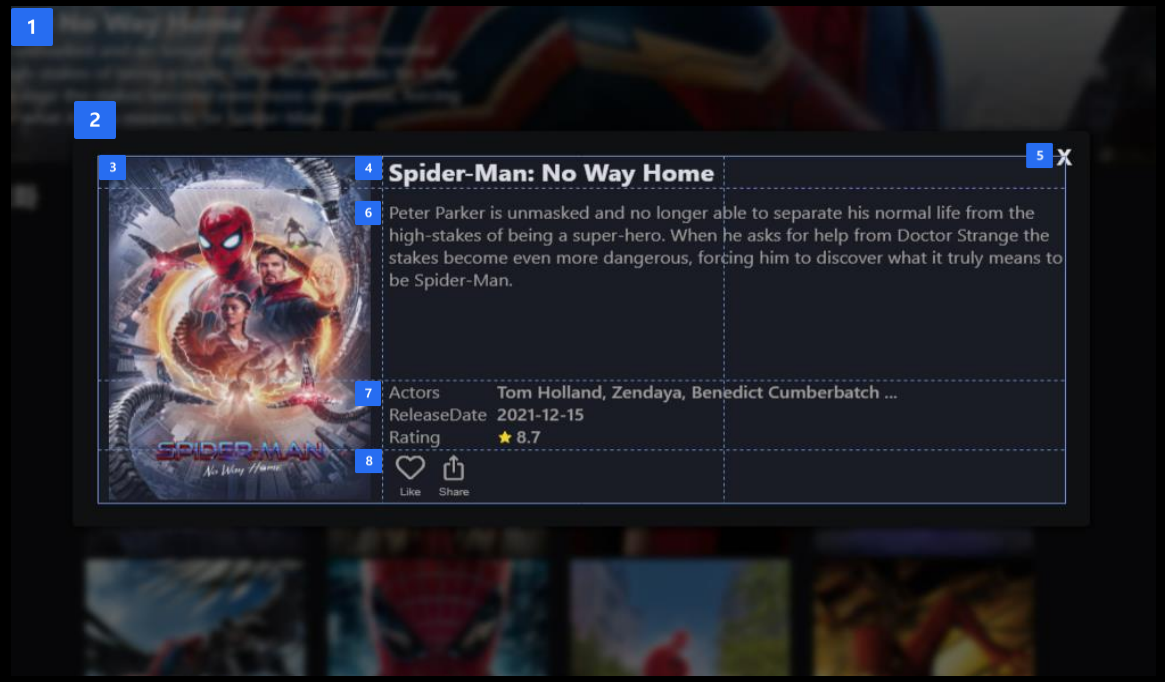

```
const getActors = async () => {
  let actorsList = [];

  const request = `https://api.themoviedb.org/3/movie/${movie.id}/credits?api_key=${API_KEY}`;

  actorsList = actorsList.concat(
    await fetch(request)
      .then((response) => response.json())
  );

  actorsList = actorsList[0].cas
    .filter((actor) => actor.order < 3)
    .map((actor) => actor.name);

  let actorsInfo = "";

  actorsList.forEach((actor, index) => {
    if (index !== actorsList.length - 1) {
      actorsInfo += actor + ", ";
    } else {
      actorsInfo += actor + " ...";
    }
  });

  setActors(actorsInfo);
};
```

Modal

Modal on/off

```
const handleModal = () => {
  if (!openModal) {
    getActors();
  }
  setOpenModal(!openModal);
};
```

Modal 레이아웃 및 설명



01 Modal_overlay : 화면 전체를 감싸는 레이아웃, 해당 오버레이 클릭 시 모달창을 닫음

02 Modal_Window : 모달창 화면

03 Poster_img : TMDB API를 통해 가져온 영화 포스터 이미지 출력 화면

04 Title : 영화 제목을 나타내는 화면

05 Close_Button : 모달창을 닫는 버튼

06 Summary : 영화 줄거리 출력 화면

07 SummaryInfo : 개요

- 07-1 Actor : 영화배우 출력 화면
- 07-2 ReleaseDate : 영화 개봉일 출력 화면
- 07-3 Rating : 영화 평점 출력 화면

08 Icons : 기타 Like,Share 아이콘 출력 화면

```jsx
export default function Modal({ movie,
modalPosterImg, actors,handleModal }) {
  return (
    //□□ blur □□□ overlay
    <div className="modal_overlay" id="modal">
      <div
        className="modal_window"
        id={movie.key}
        onClick={(e) => e.stopPropagation()}
      >
```

```jsx
{/* closeBtn*/}
<button className="close_btn" id="closeBtn" onClick={handleModal}>
  X
</button>
{/* □□ □□□ */}
<img className="poster_img" src={modalPosterImg}></img>
{/* □□ title*/}
<div className="title" id={movie.id}>
  <h3>{movie.original_title}</h3>
</div>
<div className="movieInfo">
  {/* □□ summary*/}
  <div className="content">
    <span className="summary">{movie.overview}</span>
  </div>
</div>
```

```jsx
<dl className="summaryInfo">
  {/* <dt>Genre</dt>
  <dd>{movie.genre}</dd> */}
  <dt>Actors</dt>
  <dd>{actors}</dd>
  <dt>ReleaseDate</dt>
  <dd>{movie.release_date}</dd>
  <dt>Rating</dt>
  <dd>
    <AiFillStar className="star" />
    {movie.vote_average}
  </dd>
</dl>
{/* icons*/}
<Icons />
</div>
</div>
```

프로젝트 회고

최태원

예상보다 실수가 더 많았고, 급하게 하느라 “돌아가기만 하면 된다” 라는 생각으로 진행해서
아쉽지만 끝나고 나니 꽤나 보람찬 프로젝트였습니다.

이재민

UI 구조를 만드는데 많은 시간을 소요하여 더 많은 기능을 추가하지 못했던 아쉬움과 짧은
시간에 가독성 좋은 소스코드, 최적화를 진행하지 못해 아쉽지만,
추후 더 나은 소스로 수정할 예정입니다.

이재훈

리액트를 강의로 처음 배우고 처음하는 프로젝트라 깃이라던지 리액트부분에서 굉장히 어려움이 많았는데
구글링과 조원분들의 도움으로 잘 끝낸거같습니다.
이해가 잘 안돼는 부분은계속해서 보고 고치면서 공부할 생각입니다

Q & A