

캡스톤 디자인(1)

# 머신러닝을 이용한 영화 탐색 웹

최종보고서

## 목차

<b>1. 개요 및 목적</b>	<b>1</b>
1.1 프로젝트 개요	
1.2 개발 동기	
<b>2. 요구사항</b>	<b>2</b>
2.1 User Requirements	
2.2 System Requirements	
<b>3. 설계 및 구현</b>	<b>3</b>
3.1 소프트웨어 아키텍처	
3.2 시퀀스 다이어그램	
3.3 클래스 다이어그램	
3.4 주요기능 function 설명	
<b>4. 실행 결과</b>	<b>4</b>
4.1 웹 실행결과	
<b>5. 개선 사항 및 향후 계획</b>	<b>5</b>
5.1 개선 사항	
5.2 향후 계획	

## 1. 개요 및 목적

### 1.1 프로젝트 개요

본 프로젝트는 다양한 영화를 검색하고 로그인 하여 리뷰를 남기는 커뮤니티 웹, 그리고 사용자의 리뷰내용을 바탕으로 각 사용자에게 적합한 영화를 추천해주는 웹사이트 개발을 목적으로 한다. 사용자는 원하는 영화를 검색할 수 있고, 그 영화에 대한 평가를 남긴다. 정보들은 db에 저장되어 그 정보들을 기계학습하여 평가가 높을 것으로 추정되는 영화를 알려주는 api를 개발한다.

### 1.2 개발동기

웹 개발의 백엔드, 프론트엔드 개발능력 그리고 다양한 api를 사용하고 최종적으로는 머신러닝 그리고 알고리즘에 대한 기본적인 이해까지 잘 표현 할 수 있는 주제라고 생각하여 계획하였다.

## 2. 요구사항

### 2.1 User Requirements

1. 사용자가 원하는 영화를 검색하여 찾고 상세페이지를 확인할 수 있다.
2. 상세페이지에서 영화의 올바른 정보를 얻고 리뷰, 평점을 작성할 수 있다.
3. 로그인 하여 사용자가 작성한 코멘트를 관리할 수 있다.
4. 작성한 평점을 바탕으로 사용자에게 알맞은 영화를 추천 받을 수 있다.

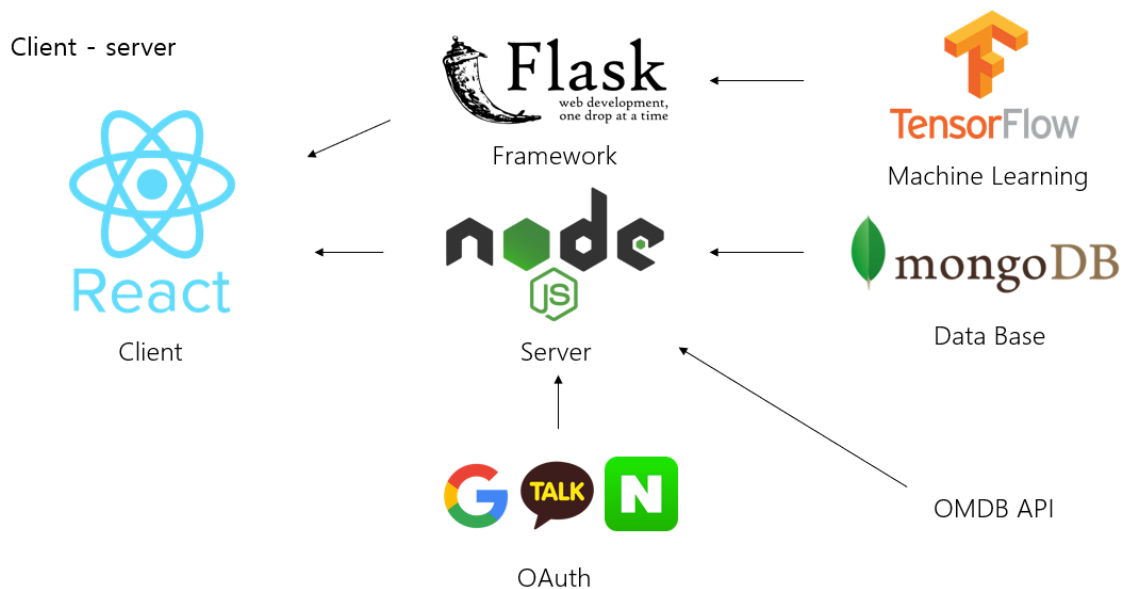
### 2.2 System Requirements

1. 각 URL에 알맞은 페이지로 올바르게 Route 한다.
2. 유저에게 전달받은 값을 올바르게 Api에 전송하고 해당하는 정보를 받아서 웹페이지에 나타낸다.

3. 로그인, 회원가입, 리뷰 등의 DB 에 저장에 필요한 정보들을 올바르게 작성하고 불러올 수 있다.
4. 사용자 평점기반으로 영화를 추천하는 모델을 적합하게 학습한다.

### 3. 설계 및 구현

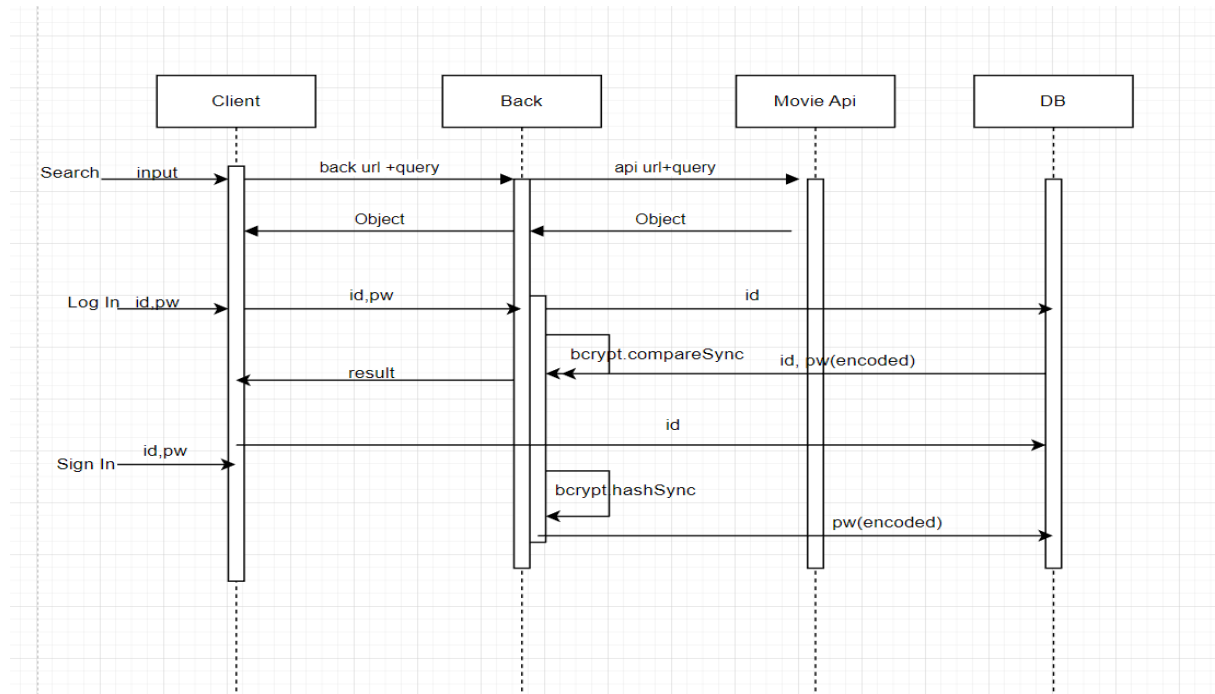
#### 3.1 소프트웨어 아키텍처



Client – server 아키텍처이다

프론트로 기본적으로 HTML, CSS, JavaScript, React 를 사용하고 서버로는 Nodejs framework 를 사용한다. DB 는 MongoDB 를 사용하고 카카오, 네이버, 구글 소셜로그인 api 를 사용할 예정이다. 영화를 불러오는데 사용한 OMDB API 는 추후 더욱 많은 기능을 요구할 수 있는 TMDb API 를 사용할 예정이다. 영화추천 모델은 TensorFlow 를 사용하여 학습할 예정이며 Flask Framework 를 통하여 api 를 서버 혹은 클라이언트에 전달 할 예정이다.

### 3.2 시퀀스 다이어그램

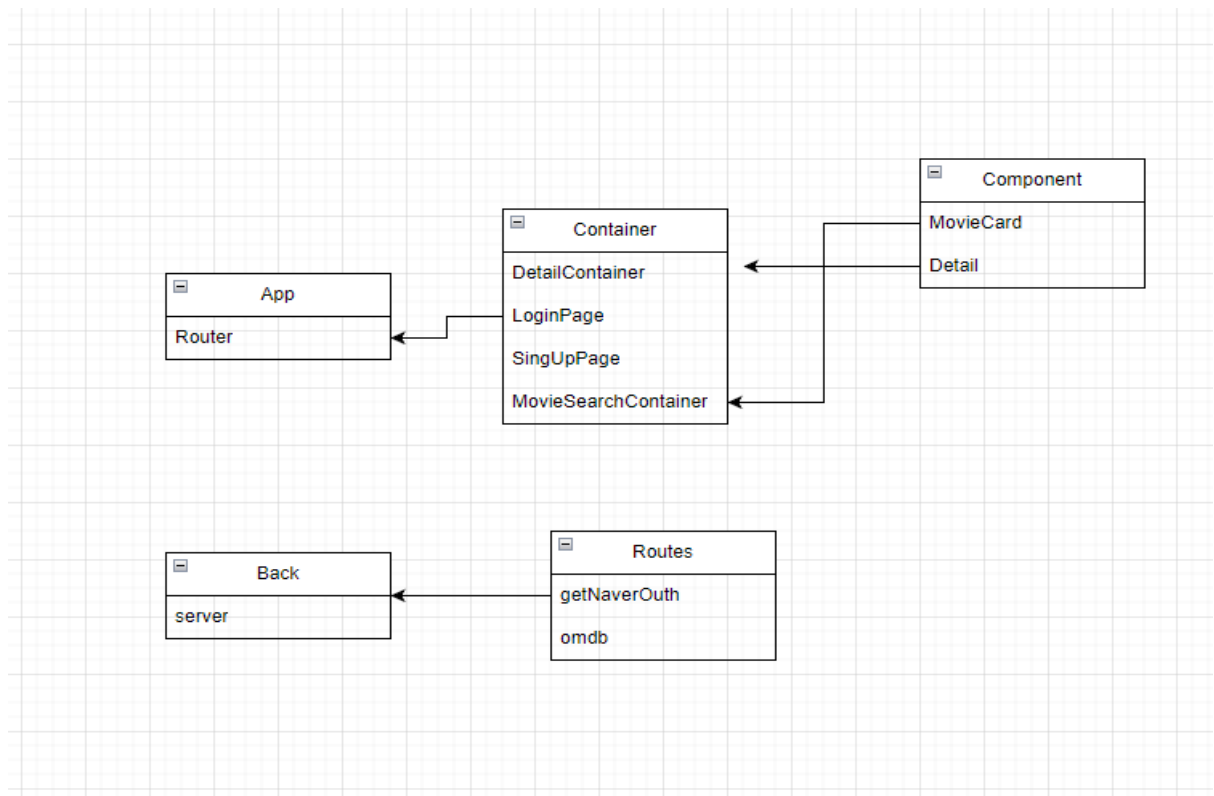


검색창에서 사용자가 input 을 입력하면 Client 에서 query 로 받아 서버에 전달한다. 서버에서 Client 에서 받은 query 를 api 로 전달하여 영화 정보 Object 를 받아낸뒤 Client 에 전달한다.

로그인에서 사용자가 id 와 password 를 입력하면 서버에서 db 에 일치하는 아이디가 있는지 확인한다. 일치하는 아이디가 있으면 db 에서 가져온 아이디에 해당하는 비밀번호를 입력한 비밀번호와 대조하여 아이디가 없음, 비밀번호가 틀림, 로그인 성공 등의 결과를 Client 에 전송한다. 이때 암호화된 비밀번호를 대조하기 위하여 bcrypt 라이브러리 내에 있는 함수를 사용한다.

사용자가 회원가입시 아이디와 비밀번호를 서버에 전달하고 서버에서 비밀번호를 암호화하여 DB 에 저장한다.

### 3.3 클래스 다이어그램



클라이언트 웹페이지를 담당하는 클래스는 크게 각 페이지를 Route 하기위한 App 그리고 각 페이지를 담은 Container 그리고 페이지안의 각각의 카드들을 나타내는 Component 세개의 구조로 설계하였다. 서버를 담당하는 Class 는 클라이언트에서 호출할 정보들을 불러오는 Routes 로만 구분해놓았다.

### 3.4 주요기능 function 설명

```
/* GET home page. */
router.get('/getOmdbMovie', async function(req,res){
  let query = req.query.query;
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Credentials', 'true'); //쿠키 주고 받기
  try{
    let movieRes= await Axios.get(
      `http://www.omdbapi.com/?i=tt3896198&apikey=79033114&s=${query}`
    );
    return res.json(movieRes.data);
  } catch (e) {
    return res.json({
      status : 400,
      message : e
    });
  }
});

module.exports = router;
```

검색된 영화들을 api 에서 불러오는 server 코드이다. client 에서 server/getOmdbMovie 에 query 값을 붙여 호출하면 그 query 값을 api 에 전달하여 값을 받은뒤 다시 client 에 값을 전달해 준다

```
/* GET home page. */
router.get('/getNaverOuth', async function(req,res){
  let id = req.query.id;

  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Credentials', 'true'); //쿠키 주고 받기 허용
  try{
    let naverOuth= await Axios.get(
      ('https://nid.naver.com/oauth2.0/authorize?response_type=code&client_id=' + process.env.REACT_APP_NAVER_CLIENT_ID + '&redirect_uri=' +
        encodeURIComponent(process.env.REACT_APP_NAVER_CALLBACK_URI) + '&state=' + Math.random().toString(36).substr(3, 14))
    );
    return res.json(naverOuth);
  } catch (e) {
    return res.json({
      status : 400,
      message : e
    });
  }
});
```

네이버 소셜로그인 코드이다. 프론트에서 호출하면 미리 발급해둔 api key 를 api url 에 넣어 완성된 네이버 로그인 창을 띄워준다.

```
const createSalt = async () => {
  const buf = await randomBytesPromise(64);

  return buf.toString("base64");
};
```

검증에 필요한 해시 key 인 salt 값을 생성하는 함수이다.

```
export const createHashedPassword = async (password) => {
  const salt = await createSalt();
  const key = await pbkdf2Promise(password, salt, 104906, 64, "sha512");
  const hashedPassword = key.toString("base64");

  return { hashedPassword, salt };
};
```

pbkdf2 모듈은 비밀번호, salt, 암호화횟수, 길이, 알고리즘을 인자로 가져 암호화된 비밀번호를 생성한다.

```
export const verifyPassword = async (password, userSalt, userPassword) => {
  const key = await pbkdf2Promise(password, userSalt, 99999, 64, "sha512");
  const hashedPassword = key.toString("base64");

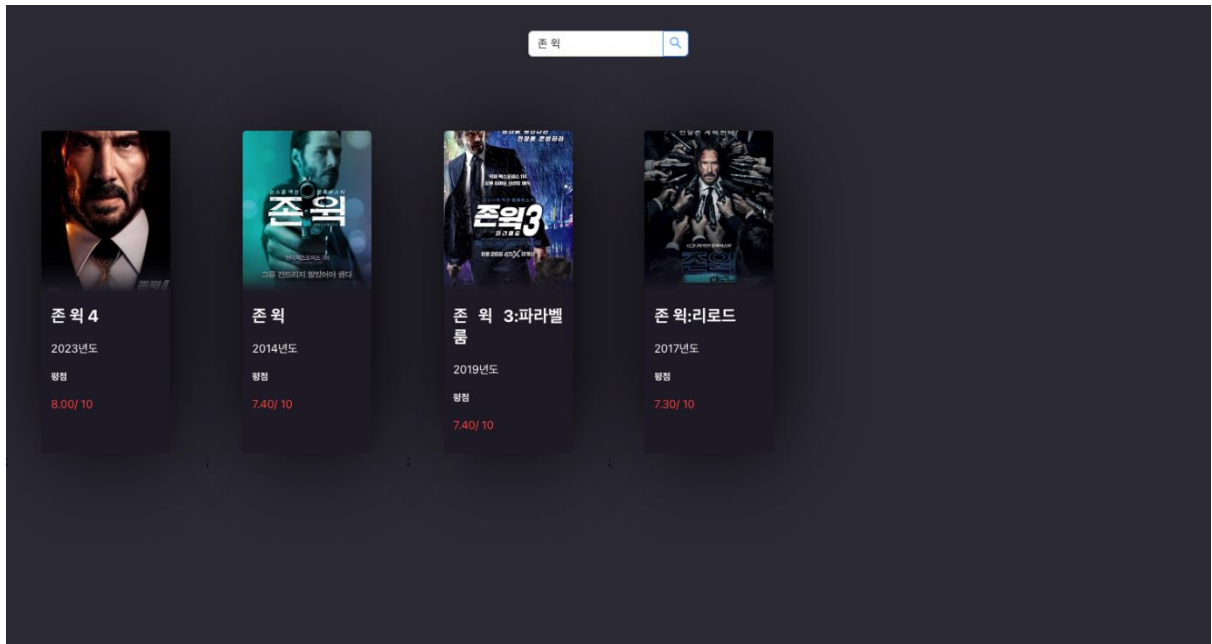
  if (hashedPassword === userPassword) return true;
  return false;
};
module.exports = app;
```

회원가입시 발행한 password 와 salt 값을 통해 비밀번호 일치여부를 확인하는 함수이다.

## 4. 실행 결과

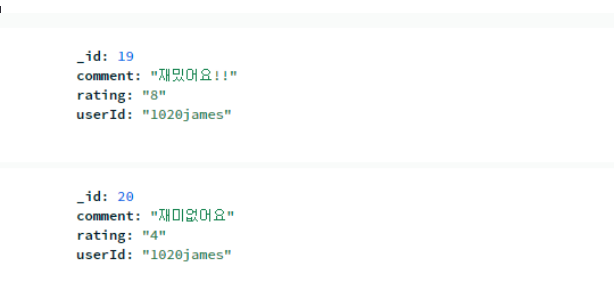
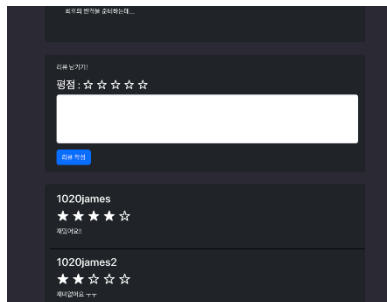
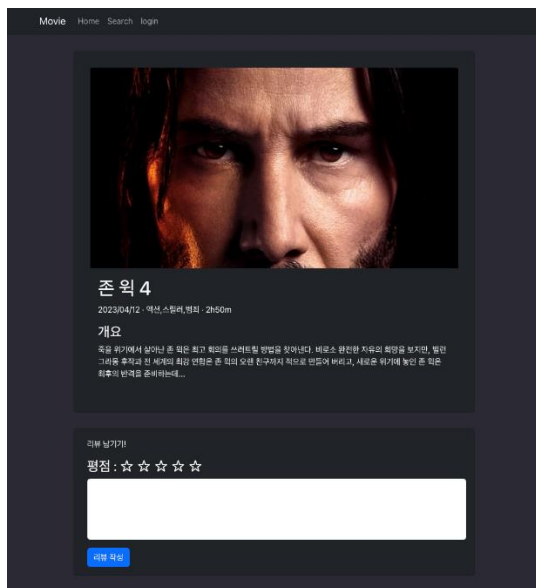
### 4.1 영화 검색





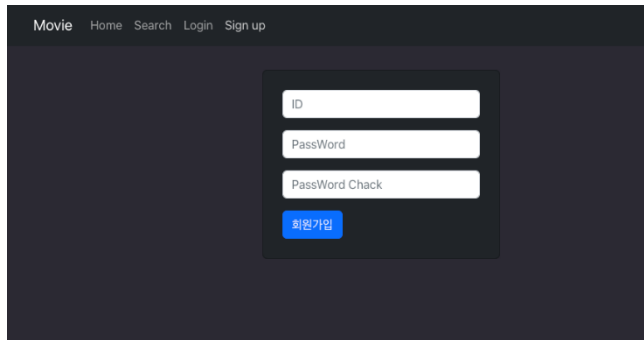
존 윅 이라는 영화를 검색하여 해당 내용 카드들을 아래에 불러온 모습이다.

## 4.2 디테일



출력된 카드를 클릭하면 영화의 상세정보들을 나타내준다. 아래에는 평점과 코멘트를 작성할 수 있다. 그리고 작성된 코멘트가 DB에 저장된 모습이다.

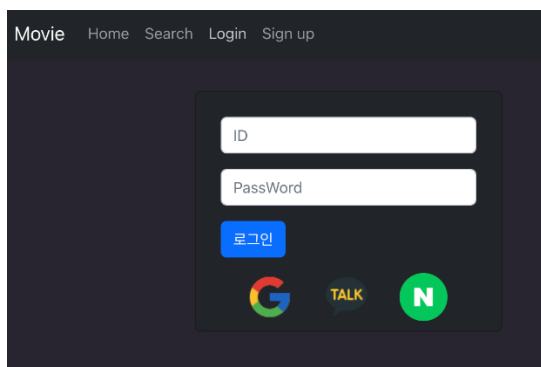
#### 4.3 회원가입

A screenshot of the 'Movie' app's registration page. The page has a dark background with a light gray header containing links: 'Movie', 'Home', 'Search', 'Login', and 'Sign up'. The main content area features a white registration form with three input fields: 'ID', 'PassWord', and 'PassWord Chack'. Below these fields is a blue button labeled '회원가입' (Sign up).

```
_id: ObjectId('646f670e2f96a3fd4f089c8c')
id: "1020james"
pw: "t43u9f4j3ff903jt43t4rhdsgfds94qnyq9aks93ajj83rea84u8385483u4543m5g943f..."
salt: "w8ngy40mq7379638a9wb78830g5321f4m32m89md3421d29mf853g68waf57c32e3f5435..."
```

아이디와 비밀번호, 비밀번호 확인 칸이 있다. 비밀번호와 비밀번호 확인칸이 일치하면 정상적으로 회원가입이 되며 DB에 id, 암호화된 비밀번호, salt 값이 저장된다.

#### 4.4 로그인

A screenshot of the 'Movie' app's login page. The page has a dark background with a light gray header containing links: 'Movie', 'Home', 'Search', 'Login', and 'Sign up'. The main content area features a white login form with two input fields: 'ID' and 'PassWord'. Below these fields is a blue button labeled '로그인' (Login). At the bottom of the form are three social login buttons: Google (G), TALK, and Naver (N).

아이디와 비밀번호를 입력하여 로그인하는 페이지이다. Db에 저장된 아이디 비밀번호가 일치하면 로그인에 성공하며 아래에는 소셜로그인 세가지 버튼이 존재한다.

## 5. 개선 사항 및 향후 계획

### 5.1 개선 사항

1. Client 의 디자인 조정 및 애니메이션등 삽입.
2. 회원 로그인 후 자기가 작성한 리뷰관리, 마이페이지 등과 자신이 작성한 글에만 보이는 Component 구현.
3. 네이버 소셜로그인 뿐만 아닌 카카오, 구글 소셜로그인 구현.
4. 검색된 영화들의 정렬기능을 구현하기위해 api 변경 고려.

### 5.2 향후 계획

- 6 월 ~7 월초 - 개선사항 구현, 최대한 영화추천기능을 제외한 기본 웹 요구사항 구현
- 7 월~8 월 - 영화추천 모델 data set 확보 후 모델학습
- 9 월~(2 학기) - 모델을 성공적으로 웹에서 이용, 알맞은 추천알고리즘 구현