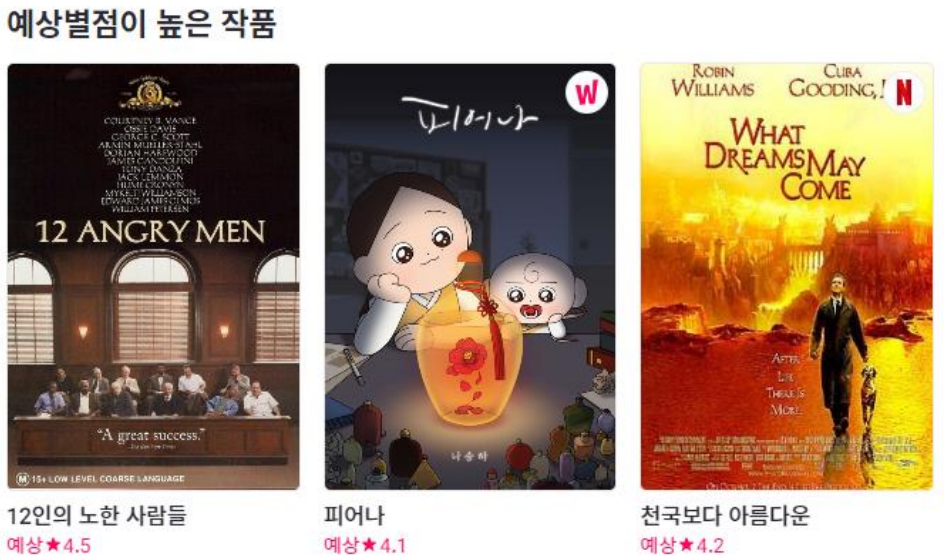
캡스톤 디자인(2)
최종발표

머신러닝을 이용한 영화 탐색 및 추천 웹

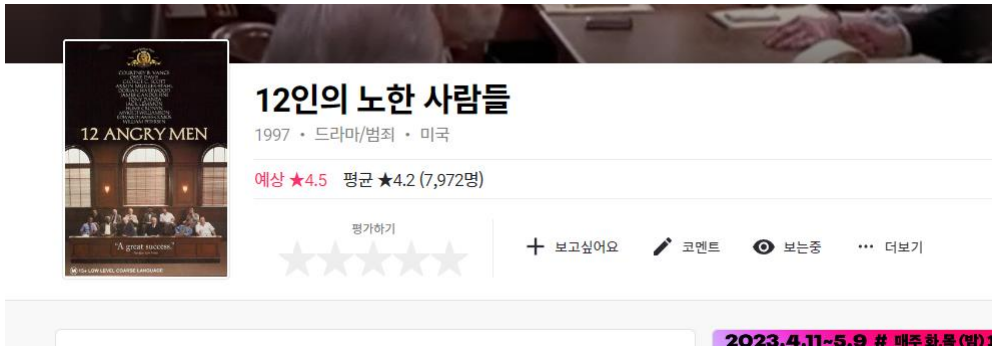
목차

개요 및 목적

영화 검색



예상 별점이 높은 작품 추천



영화 평가, 코멘트 남기기, 커뮤니티

요구 명세

User Requirement

1. 사용자가 원하는 영화를 검색하여 찾을 수 있다.
2. 로그인이 가능하고 평점을 작성하여 저장 할 수 있다.
3. 작성한 평점을 바탕으로 사용자에게 맞는 영화를 추천해줄 수 있다.

System Requirement

1. URL에 알맞은 페이지를 전송한다.
2. 로그인, 리뷰 등의 정보를 DB에 올바르게 작성하고 불러 올 수 있다.
3. 목적에 맞는 머신러닝 Api 를 작성하고 활용 할 수 있다.

소프트웨어 아키텍쳐

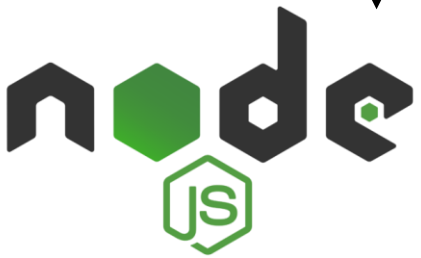
Client - server

Framework

TensorFlow

Machine Learning

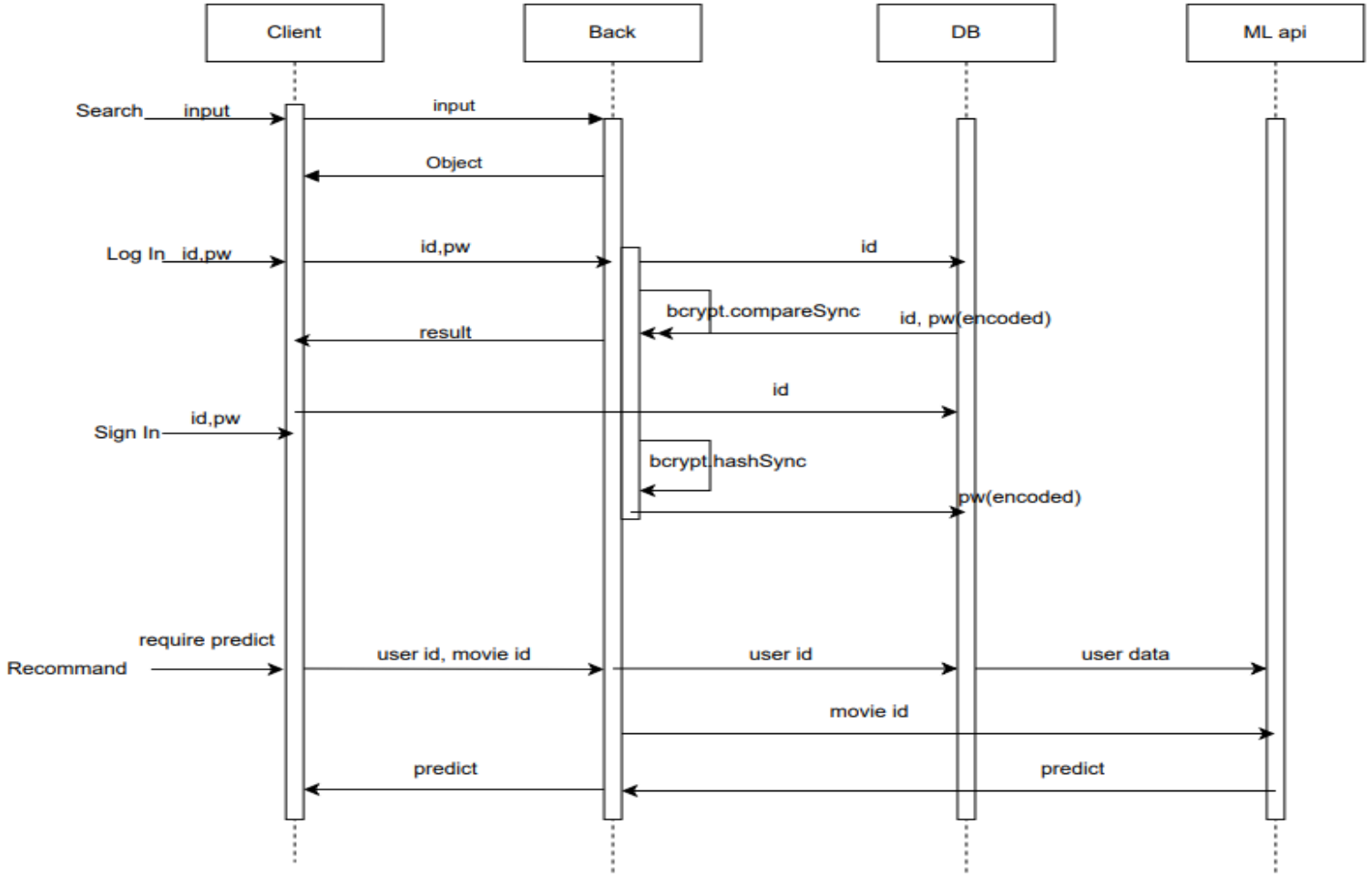
Client



Server

Data Base

Sequence Diagram



머신러닝

R

	Movie1	Movie2	Movie3	Movie4	Movie5
User1	4			2	
User2		5		3	
User3			3	4	4
User4	5	2	1	2	

R =~ P X Q.T

	Factor1	Factor2
User1	0.94	0.96
User2	2.14	0.08
User3	1.79	1.76
User4	0.58	1.59

P

X

=

	Movie1	Movie2	Movie3	Movie4	Movie5
Factor1	1.7	2.3	1.41	1.36	0.41
Factor1	2.49	0.41	0.14	0.75	1.77

Q.T

머신러닝

SGD를 이용하여 P,Q행렬을 학습.

5-fold cross validation을 통해 learning_rate, lambda 조절

```python
def matrix_factorization(R, K, steps=200, learning_rate=0.01, r_lambda=0.01):
    num_users, num_items = R.shape
    np.random.seed(1)
    P = np.random.normal(scale=1./K, size=(num_users, K))
    Q = np.random.normal(scale=1./K, size=(num_items, K))

    break_count = 0
    non_zeros = [(i, j, R[i, j]) for i in range(num_users) for j in range(num_items) if R[i, j] > 0]

    for step in range(steps):
        for i, j, r in non_zeros:
            eij = r - np.dot(P[i, :], Q[j, :].T)
            P[i, :] = P[i, :] + learning_rate * (eij * Q[j, :] - r_lambda * P[i, :])
            Q[j, :] = Q[j, :] + learning_rate * (eij * P[i, :] - r_lambda * Q[j, :])

        rmse = get_rmse(R, P, Q, non_zeros)
        if (step%10) == 0:
            print("### iteration step : ", step, " rmse : ", rmse)

    return P, Q
```

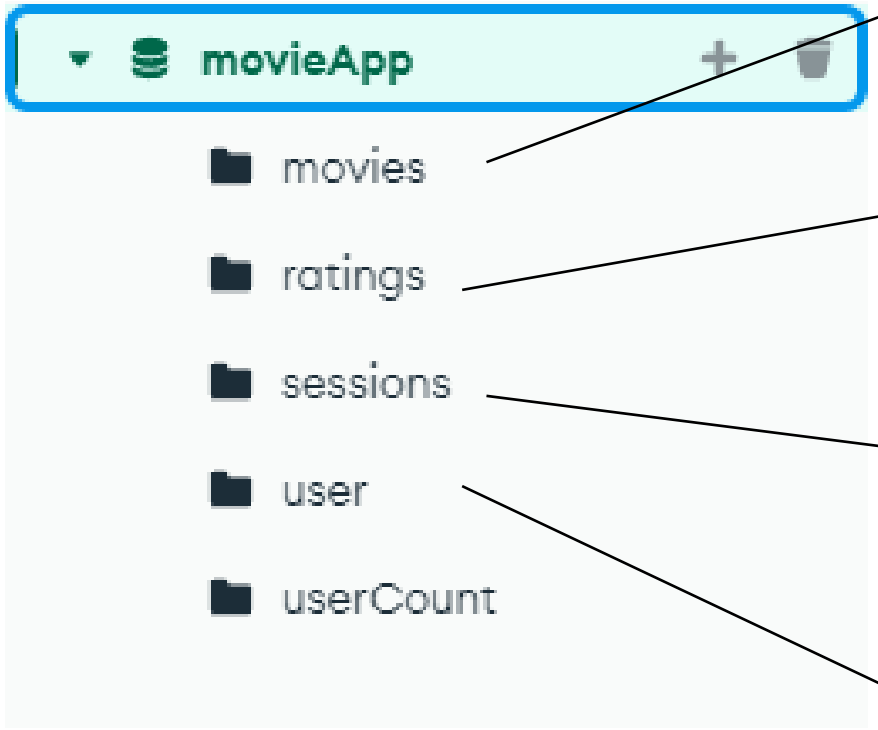
```
### iteration step :  110  rmse :  0.13587915240890538
### iteration step :  120  rmse :  0.1309988056679569
### iteration step :  130  rmse :  0.12700752742900484
### iteration step :  140  rmse :  0.12368883727233916
### iteration step :  150  rmse :  0.12089057936741691
### iteration step :  160  rmse :  0.11850272024515557
### iteration step :  170  rmse :  0.11644375567735363
### iteration step :  180  rmse :  0.11465207071295917
### iteration step :  190  rmse :  0.11308026280230041
Average RMSE across 5-fold cross-validation: 0.1585975552506638
```

Flask

학습된 pred_matrix 결과값을
웹서버에서 사용

```python
import numpy as np
from flask import Flask, jsonify, request

app = Flask(__name__)

pred_matrix=np.genfromtxt("predict.csv", delimiter=",", encoding='UTF8')

# 예측 행렬을 활용하여 사용자가 가진 평점이 높은 상위 9개의 영화를 추천하는 함수
def get_top_recommendations(user_ratings, top_n=9):
    # 사용자가 가진 평점을 내림차순으로 정렬하고, 상위 top_n개를 추출
    top_indices = np.argsort(user_ratings)[::-1][:top_n]
    return top_indices

@app.route('/get_prediction', methods=['POST'])
def get_prediction():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자와 영화 ID를 받아서 예측 평점 반환
        user_id = data.get('userId')
        movie_id = data.get('movieId')

        # 예측 행렬에서 해당 위치의 값을 가져옴
        predicted_rating = pred_matrix[user_id, movie_id]

        # JSON 형태로 응답
        response = {'predictedRating': predicted_rating}
        return jsonify(response)

@app.route('/get_recommendations', methods=['POST'])
def get_recommendations():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자 ID를 받아서 해당 사용자의 예측 평점 반환
        user_id = data.get('userId')
        user_ratings = pred_matrix[user_id]

        # 사용자가 가진 평점이 높은 상위 9개의 영화 추천
        top_recommendations = get_top_recommendations(user_ratings)

        # JSON 형태로 응답
        response = {'topRecommendations': top_recommendations.tolist()}
        return jsonify(response)

if __name__ == '__main__':
    # 서버 실행
    app.run(debug=True)
```

DB 구조



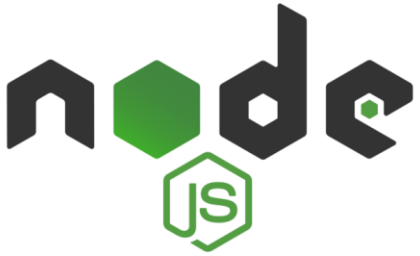

```
_id: ObjectId('6567ccad58e170ce08a27a0b')
movieId: 21
title: "Get Shorty (1995)"
content: ""
```

```
_id: ObjectId('6567cd8b58e170ce08a2a010')
userId: 1
movieId: 163
rating: 5
```

```
_id: "6-AgwGXCJfdPUhK65cMbFvYvLXpjWtX3"
expires: 2023-11-30T21:15:01.591+00:00
session: "{"cookie":{"originalMaxAge":3600000,"expires":"2023-11-30T21:14:23.868…"
```

```
_id: ObjectId('6568ed1058500b5cd410b31d')
userId: 1
username: "test"
password: "$2b$10$Oj3ll6BkT6XVRihsI9nIm.SIgim0HUtPTXYyAFgahtdCZAmtDITiy"
```

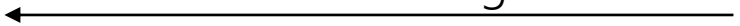
DB 구조



POST /get_predict

userId , movieId

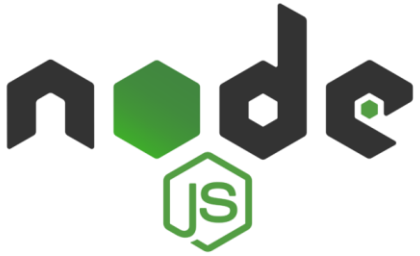
Predict rating



해당되는 유저, 영화의
예측평점을 전송

```python
@app.route('/get_prediction', methods=['POST'])
def get_prediction():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자와 영화 ID를 받아서 예측 평점 반환
        user_id = data.get('userId')
        movie_id = data.get('movieId')

        # 예측 행렬에서 해당 위치의 값을 가져옴
        predicted_rating = pred_matrix[user_id, movie_id]

        # JSON 형태로 응답
        response = {'predictedRating': predicted_rating}
        return jsonify(response)
```

DB 구조



POST /get_recommandation

userId

recommendation

해당되는 유저의 예측평점 상위 9개 영화를 전송

```python
@app.route('/get_recommendations', methods=['POST'])
def get_recommendations():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자 ID를 받아서 해당 사용자의 예측 평점 반환
        user_id = data.get('userId')
        user_ratings = pred_matrix[user_id]

        # 사용자가 가진 평점이 높은 상위 9개의 영화 추천
        top_recommendations = get_top_recommendations(user_ratings)

        # JSON 형태로 응답
        response = {'topRecommendations': top_recommendations.tolist()}
        return jsonify(response)
```

웹 구현

회원가입

```
app.post('/signIn', async (req, res) => {
  let hash = await bcrypt.hash(req.body.password, 10)
  let count = await db.collection('userCount').findOne({_id : new ObjectId('6567b2c09398c21fd4fc4849')})
  await db.collection('userCount').updateOne({ _id : new ObjectId('6567b2c09398c21fd4fc4849') },
    {$inc : { count : 1 }
  })
  await db.collection('user').insertOne({
    userId : (count.count+1),
    username : req.body.username,
    password : hash
  })
  res.redirect('/')
})
```

Bcrypt 라이브러리를 이용하여 비밀번호 암호화 후 db.user collection에 저장.
관리를 용이하게 하기위하여 가입순서대로 userId 부여.

웹 구현

로그인

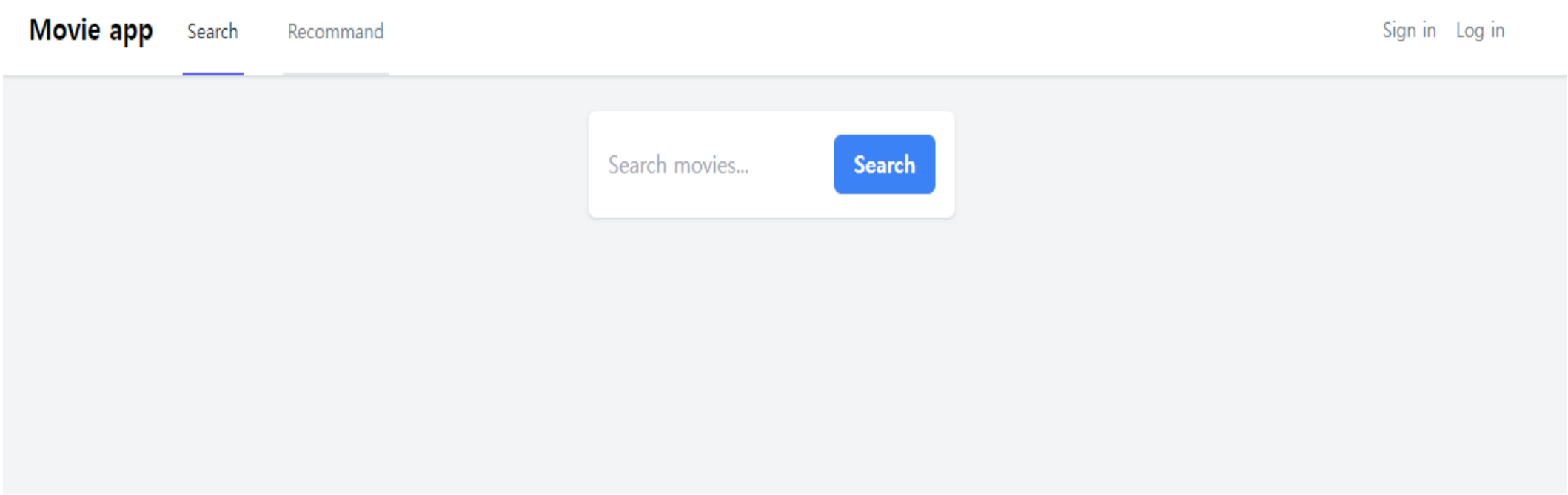
```
app.post('/login', async (req, res, next) => {

  passport.authenticate('local', (error, user, info) => {
      if (error) return res.status(500).json(error)
      if (!user) return res.status(401).json(info.message)
      req.logIn(user, (err) => {
        if (err) return next(err)
        res.redirect('/')
      })
  })(req, res, next)

})
```

```
app.use(passport.initialize())
app.use(session({
  secret: 'wjsghk4043',
  resave : false,
  saveUninitialized : false,
  cookie : { maxAge : 60 * 60 * 1000 },
  store : MongoStore.create({
    mongoUrl : process.env.DB_URL ,
    dbName: 'forum',
  })
}))
app.use(passport.session())
```

Passport 라이브러리를 이용.
암호화된 비밀번호와 비교하여 일치하면 로그인 성공.
로그인 성공시 1시간 유지되는 쿠키 생성.

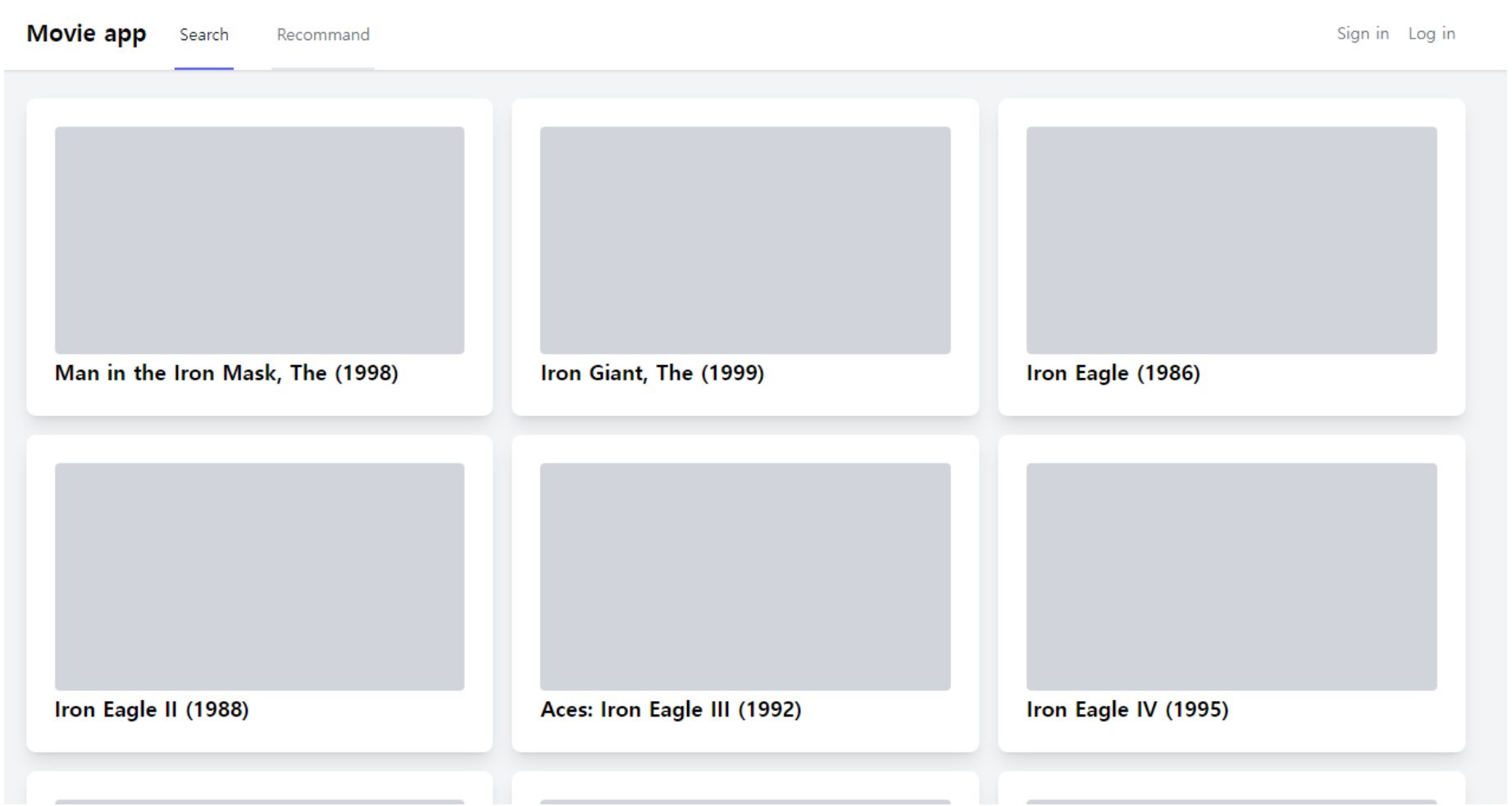
웹 구현

검색


Movie app
Search
Recommand
Sign in
Log in
Search movies...
Search

웹 구현

검색


Movie app
Search
Recommand
Sign in
Log in
Man in the Iron Mask, The (1998)
Iron Giant, The (1999)
Iron Eagle (1986)
Iron Eagle II (1988)
Aces: Iron Eagle III (1992)
Iron Eagle IV (1995)

웹 구현

검색

```
<script>
    document.querySelector('#search-send').addEventListener('click', function(){
      let searchInput = document.querySelector('#search').value
      location.href = '/searchResult?val=' + searchInput
    })
</script>
```

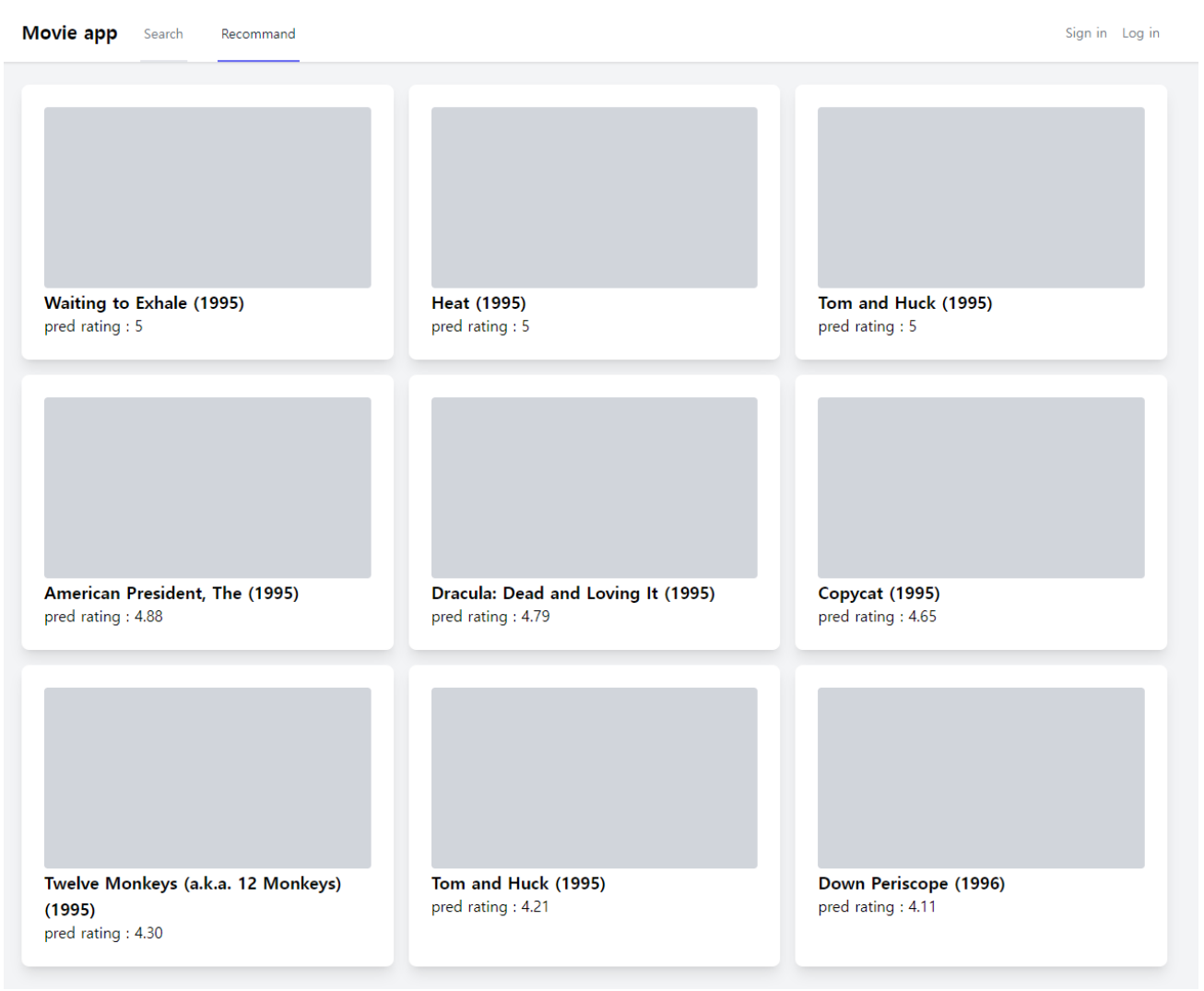
```
app.get('/searchResult', async(req, res) => {
  let result = await db.collection('movies').find( {title :{$regex : req.query.val}} ).limit(21).toArray()
  res.render('searchResult.ejs', { movieList : result })
})
```

제목을 입력후 검색시 해당 텍스트가 포함된 영화를 나열해주는 /searchResult?=val 사이트로 연결

웹 구현

추천영화 목록

로그인 되어있는 user의
예측평점이 높은 상위 9개의
영화를 추천



웹 구현

추천영화 목록

```
app.get('/recommend', async(req, res)=>{
  let movieList= await db.collection('movies').find().toArray()
  let predict = await axios.post("http:://localhost:5000/get_recommendations",{userId : req.user.id})
  res.render('recommend.ejs',{movieList : movieList, predictList : predict})
})
```

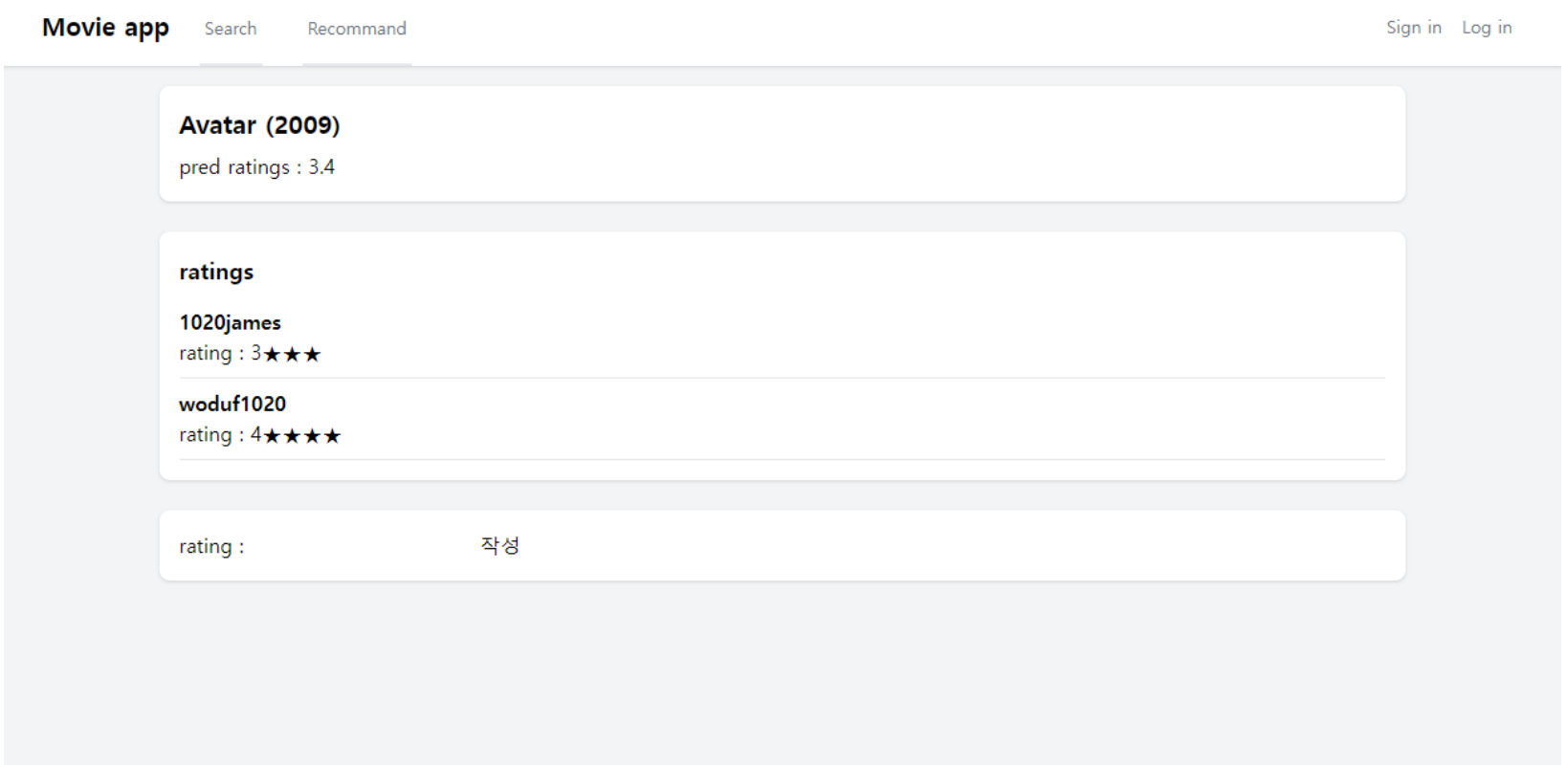
로그인 된 userId값을 post
flask에서 추천영화 목록을 반환

```
@app.route('/get_recommendations', methods=['POST'])
def get_recommendations():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자 ID를 받아서 해당 사용자의 예측 평점 반환
        user_id = data.get('userId')
        user_ratings = pred_matrix[user_id]

        # 사용자가 가진 평점이 높은 상위 9개의 영화 추천
        top_recommendations = get_top_recommendations(user_ratings)

        # JSON 형태로 응답
        response = {'topRecommendations': top_recommendations.tolist()}
        return jsonify(response)
```

웹 구현

상세페이지

영화제목, 예측평점, 유저의 평점 출력.
해당영화의 평점을 매길 수 있음



웹 구현

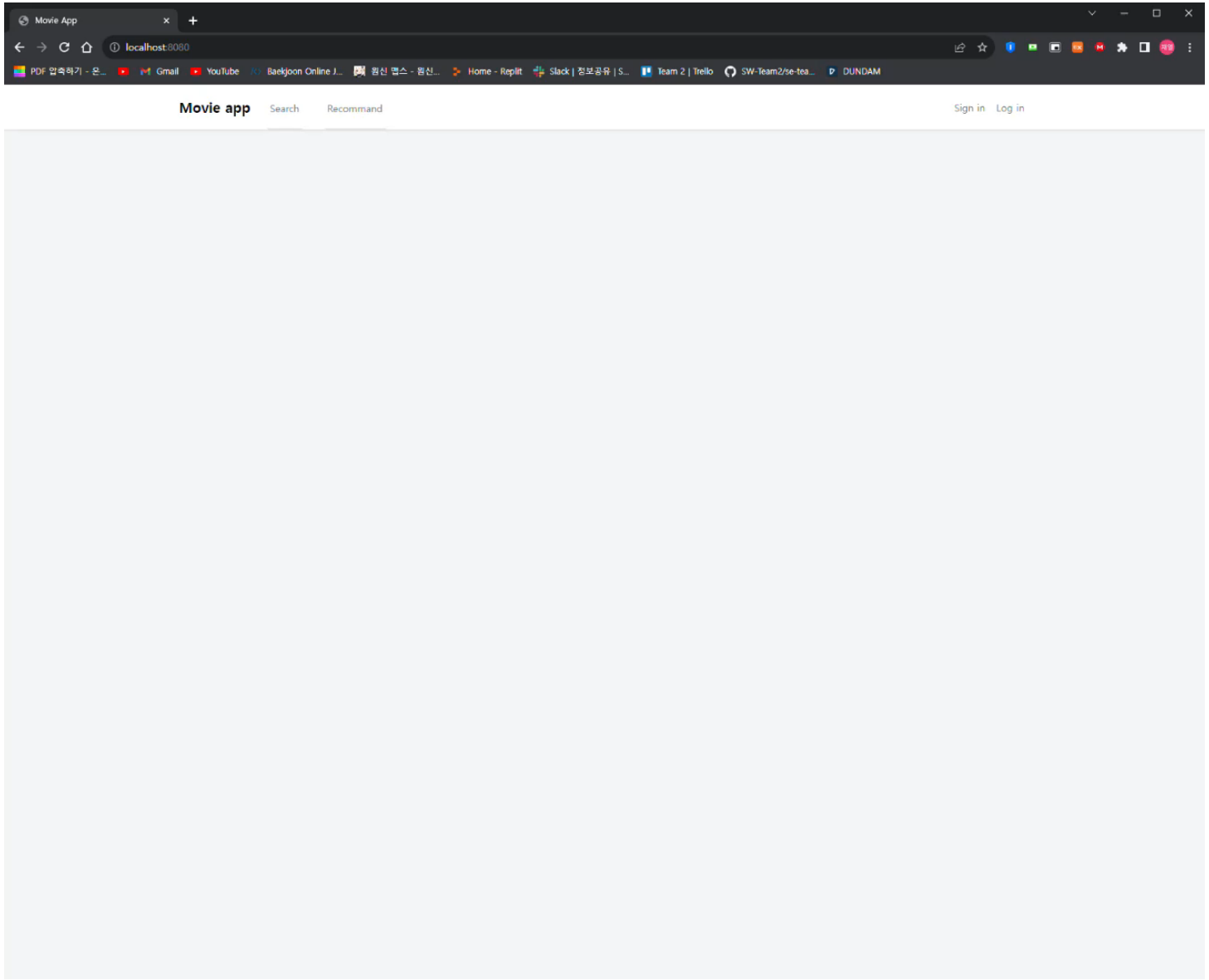
예측평점 코드

```python
@app.route('/get_prediction', methods=['POST'])
def get_prediction():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자와 영화 ID를 받아서 예측 평점 반환
        user_id = data.get('userId')
        movie_id = data.get('movieId')

        # 예측 행렬에서 해당 위치의 값을 가져옴
        predicted_rating = pred_matrix[user_id, movie_id]

        # JSON 형태로 응답
        response = {'predictedRating': predicted_rating}
        return jsonify(response)
```

영화제목, 예측평점, 유저의 평점 출력.
해당영화의 평점을 매길 수 있음

```javascript
app.get('/get_predict/:id', async(req,res) => {
  let predict = await axios.post("http:://localhost:5000/get_predict", {userId: req.user.userId, movieId: req.params.id
  res.send(predict)
})
```

시연

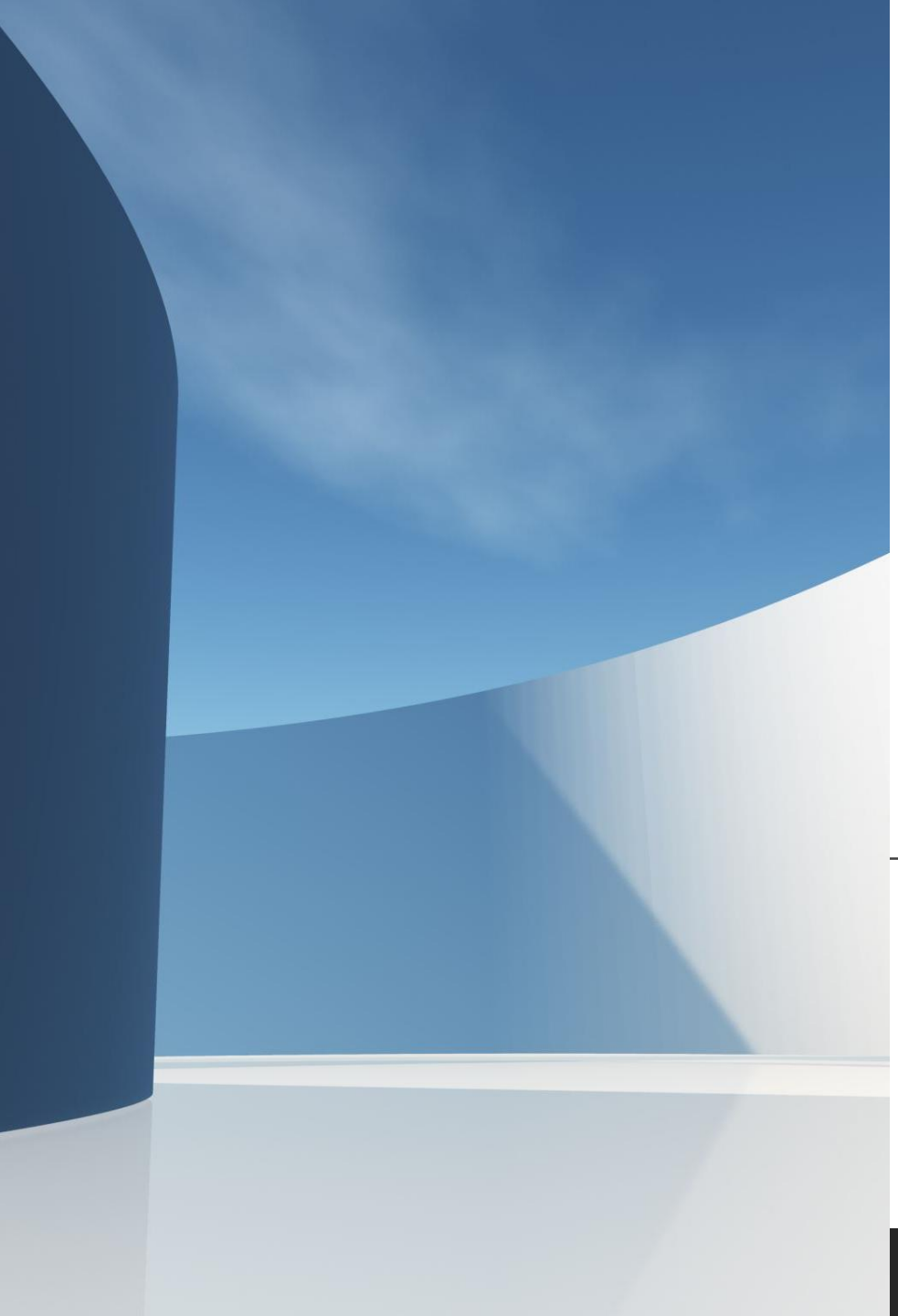

Movie App
localhost:8080
PDF 압축하기 - 온...
Gmail
YouTube
Baekjoon Online J...
원신 맵스 - 원신...
Home - Replit
Slack | 정보공유 | S...
Team 2 | Trello
SW-Team2/se-tea...
DUNDAM
Movie app
Search
Recommand
Sign in
Log in

개선 사항

- 모델 학습 자동화
- 상세페이지 이미지 첨부기능
- 웹 디자인 추가
- AWS에 서버배포

작품의 의미

- 기초적인 DB관리, 웹페이지 작성
- DB에서 가져온 데이터로 학습하는 기능
- 임의의 프로그램을 서버로 만들어 웹에서 활용



감사합니다