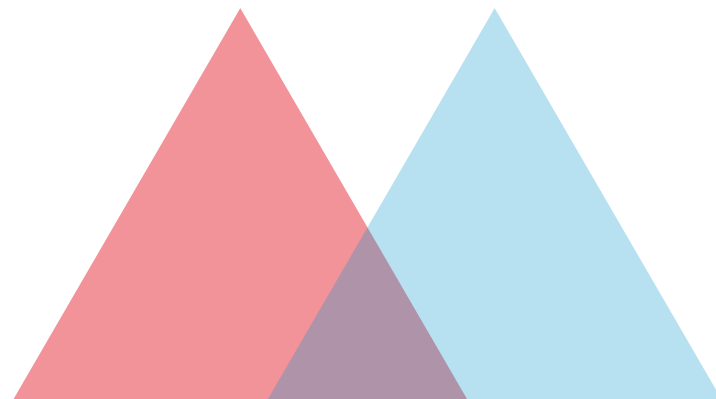


문구반점



Bbacx22

(고귀환, 임준우, 한승희)

“Expo go” 를
다운로드해주세요

Contents

001 프로젝트 동기

002 개발 과정

- Backend
- Machine-Learning
- Frontend

003 앱 소개

- 알고리즘 및
인공지능
- 앱 개발

004 차후 계획

001

프로젝트 동기





002-1

개발 과정





PostgreSQL



Prisma



GraphQL

+

A P O L L O

DOCS

SCHEMA

QUERIES

seePhotoComments(...): [Comment]!

seeLikedPhoto(...): [Photo]

seePhoto(...): [Photo]!

seeSeenPhoto(...): [Photo]

seeUploadedPhoto(...): [Photo]

getAdjacents: [Int]!

me: User

seeProfile(...): User

MUTATIONS

createComment(...): createCommentResult!

deleteComment(...): deleteCommentResult!

editComment(...): editCommentResult!

deletePhoto(...): deletePhotoResult!

editPhoto(...): editPhotoResult!

markSeen(...): markSeenResult!

toggleLike(...): toggleLikeResult!

uploadMany(...): [uploadManyResult!]

uploadPhoto(...): Photo

createAccount(...): createAccountResult!

editProfile(...): editProfileResult!

login(...): LoginResult!

DB
Models

All Models

User	27
Photo	135
Like	71
Seen	170
Comment	0

Seen X User X Photo X Like X +

Filters None Fields All Showing 100 of 135

id #

1

2

3

4

5

6

<input checked="" type="checkbox"/>	Search	
<input checked="" type="checkbox"/>	id	Int
<input checked="" type="checkbox"/>	user	User
<input checked="" type="checkbox"/>	userId	Int
<input checked="" type="checkbox"/>	file	String
<input checked="" type="checkbox"/>	caption	String?
<input checked="" type="checkbox"/>	seen	Seen[]
<input checked="" type="checkbox"/>	likes	Like[]
<input checked="" type="checkbox"/>	comments	Comment[]
<input checked="" type="checkbox"/>	createdAt	DateTime
<input checked="" type="checkbox"/>	updatedAt	DateTime

Seen X User X Photo X Like X + in X User X Photo X Like X +

Filters None

Fields All

Showing 100 of 1

Filters None Fields All Showing 27 of 27

#

id #

105

106

107

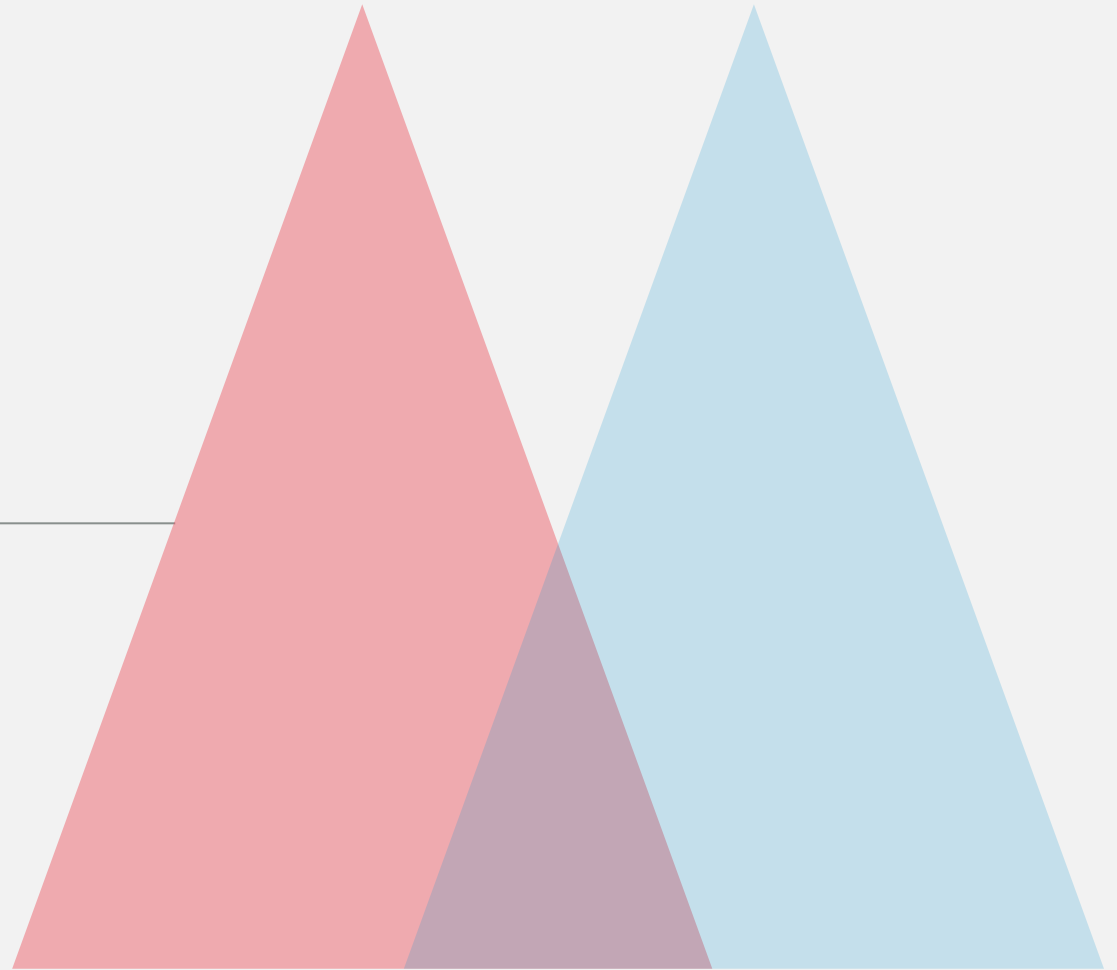
108

<input checked="" type="checkbox"/>	Search	
<input checked="" type="checkbox"/>	id	Int
<input checked="" type="checkbox"/>	photo	Photo
<input checked="" type="checkbox"/>	user	User
<input checked="" type="checkbox"/>	photoId	Int
<input checked="" type="checkbox"/>	userId	Int
<input checked="" type="checkbox"/>	createdAt	DateTime
<input checked="" type="checkbox"/>	updatedAt	DateTime

<input checked="" type="checkbox"/>	Search	
<input checked="" type="checkbox"/>	id	Int
<input checked="" type="checkbox"/>	firstName	String
<input checked="" type="checkbox"/>	lastName	String?
<input checked="" type="checkbox"/>	username	String
<input checked="" type="checkbox"/>	email	String
<input checked="" type="checkbox"/>	password	String
<input checked="" type="checkbox"/>	bio	String?
<input checked="" type="checkbox"/>	avatar	String?
<input checked="" type="checkbox"/>	photos	Photo[]
<input checked="" type="checkbox"/>	seen	Seen[]
<input checked="" type="checkbox"/>	likes	Like[]
<input checked="" type="checkbox"/>	comments	Comment[]
<input checked="" type="checkbox"/>	createdAt	DateTime
<input checked="" type="checkbox"/>	updatedAt	DateTime

002-2

데이터 수집 및 처리



```
def scroll_down(driver):  
    SCROLL_PAUSE_TIME = 5  
  
    while True:  
        time.sleep(SCROLL_PAUSE_TIME)  
  
        last_height = driver.execute_script("return document.body.scrollHeight")  
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")  
        time.sleep(SCROLL_PAUSE_TIME)  
        new_height = driver.execute_script("return document.body.scrollHeight")  
  
        if new_height == last_height:  
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")  
            time.sleep(SCROLL_PAUSE_TIME)  
            new_height = driver.execute_script("return document.body.scrollHeight")  
  
            if new_height == last_height:  
                break  
  
        else:  
            last_height = new_height  
            continue
```

```
chrome_options = Options()
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')

driver = webdriver.Chrome(options=chrome_options)

print('open chrome')

search_words = ['감성문구', '예쁜 문구', '예쁜 글귀', '프사하기 좋은 글귀', '행복한 문구', '짧은 문구', '워트있는 문구', '예쁜 시', '명언 글귀', '멋진 글귀']

daum_data_lst = pickle.load(open('daum_data_lst.p', 'rb'))

print('load pickle file')
```

```

for i in range(1, len(search_words)):
    print()
    print(f'{i}번째')
    pre_word = search_words[i-1]
    search_word = search_words[i]
    url = 'https://search.daum.net/search?w=img&m=&q=' +
        search_word +
        '&nzq=' +
        pre_word +
        'DA=NSJ'
    driver.get(url)
    print('url get')

    scroll_down(driver)
    print('scroll_down')

    req = driver.page_source
    soup = BeautifulSoup(req, 'html.parser')

    thumbnails = soup.select('#imgList > div > a > img')

    print('select soup')

    lst_len = len(daum_data_lst)

    for idx, thumbnail in enumerate(thumbnails):
        # src가 이미지 url임!
        src = thumbnail['src']
        a = {}
        a['id_num'] = lst_len + idx
        a['from'] = 'daum'
        a['data'] = src
        a['type'] = 'img'
        daum_data_lst.append(a)

    print('data extract')

```

Noise 데이터 제거

```
noise = [14, 16, 17, 24, 26, 27, 28, 30, 31, 34, 35, 36, 38, 44, 46, 47, 48, 49, 50, 53, 55, 56, 57,
109, 111, 115, 117, 118, 119, 120, 125, 131, 132, 134, 138, 139, 143, 145, 146, 147, 150,
193, 194, 196, 197, 200, 201, 203, 204, 208, 209, 210, 211, 212, 214, 215, 217, 219, 220,
252, 254, 255, 256, 259, 264, 266, 267, 277, 280, 284, 293, 294, 302, 303, 305, 306, 310,
366, 373, 380, 383, 386, 387, 388, 391, 392, 393, 404, 409, 410, 411, 415, 418, 419, 420,
460, 461, 462, 463, 464, 466, 468, 470, 473, 475, 477, 480, 481, 482, 483, 484, 485, 487,
514, 516, 517, 518, 519, 520, 522, 523, 525, 527, 529, 530, 531, 532, 534, 535, 536, 537,
563, 564, 565, 566, 567, 568, 570, 571, 572, 573, 574, 577, 579, 580, 581, 582, 583, 584,
619, 620, 621, 622, 623, 625, 628, 629, 630, 631, 632, 633, 634, 636, 637, 638, 639, 640,
685, 687, 688, 689, 690, 691, 693, 694, 696, 697, 698, 700, 711, 763, 773, 797, 801, 812,
1023, 1024, 1025, 1026, 1033, 1040, 1041, 1042, 1045, 1047, 1048, 1049, 1050, 1051, 1052,
1095, 1096, 1102, 1106, 1110, 1115, 1120, 1126, 1128, 1133, 1134, 1136, 1137, 1139, 1142,
1256, 1262, 1265, 1269, 1274, 1275, 1279, 1280, 1282, 1283, 1284, 1285, 1286, 1287, 1288,
1305, 1306, 1307, 1308, 1311, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322,
1344, 1345, 1346, 1347, 1348, 1351, 1352, 1353, 1355, 1357, 1358, 1359, 1360, 1361, 1362,
1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1394, 1396, 1397, 1398, 1399, 1400,
1417, 1418, 1419, 1420, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432,
1450, 1451, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1467,
```

```
plus = [n for n in range(2101, 2401)]
print(plus[0], plus[-1])
start = 2849
print(len(noise))
noise += plus
noise.sort()
len(noise)
```

Noise 데이터 제거

```
for i in range(start, len(full_data_lst)):
    url = full_data_lst[i]['data']
    res = requests.get(url)
    request_get_img = Image.open(BytesIO(res.content))
    display.clear_output(wait=True)
    plt.imshow(request_get_img)
    plt.title(f'step: {i}')
    plt.show()
    time.sleep(2)
```

```
for i in range(len(full_data_lst)):
    full_data_lst[i]['url'] = full_data_lst[i]['data']
    del full_data_lst[i]['data']
    del full_data_lst[i]['from']
```

```
dic = {}
for i in noise:
    dic[i] = True
for i in range(len(full_data_lst)):
    if dic.get(i, False):
        full_data_lst[i] = None
full_data_lst[10:20]
```

png파일로 변환해서 저장

```
for i in range(len(full_data_lst)):
    try:
        url = full_data_lst[i]['url']
        res = requests.get(url)
        request_get_img = Image.open(BytesIO(res.content))
        resize_img = request_get_img.resize(size=(750 // 4, 1334 // 4))
        display.clear_output(wait=True)
        plt.imshow(resize_img)
        plt.title(f'step: {i}')
        plt.show()
        resize_img.save('drive/MyDrive/img_data/img_{:0>4}'.format(i), 'png')
    except:
        continue
```

AutoEncoder (실패)

```
def get_hr_and_lr(image_path):  
    img = tf.io.read_file(image_path)  
    img = tf.image.decode_jpeg(img, channels=3)  
    img = tf.image.convert_image_dtype(img, tf.float32)  
  
    hr = tf.image.random_crop(img, [80, 80, 3])  
    lr = tf.image.resize(hr, [20, 20])  
    lr = tf.image.resize(lr, [80, 80])  
    return hr, lr
```

- hr : 고화질 원본 데이터
- lr : 임의로 화질을 저하시킨 데이터


```

def REDNet(num_layers):
    conv_layers = []
    deconv_layers = []
    residual_layers = []

    inputs = tf.keras.layers.Input(shape=(None, None, 3))
    conv_layers.append(tf.keras.layers.Conv2D(3, kernel_size=3, padding='same', activation='relu'))

    for i in range(num_layers-1):
        conv_layers.append(tf.keras.layers.Conv2D(64, kernel_size=3, padding='same', activation='relu'))
        deconv_layers.append(tf.keras.layers.Conv2DTranspose(64, kernel_size=3, padding='same', activation='relu'))

    deconv_layers.append(tf.keras.layers.Conv2DTranspose(3, kernel_size=3, padding='same'))

    # 인코더 시작
    x = conv_layers[0](inputs)

    for i in range(num_layers-1):
        x = conv_layers[i+1](x)
        if i % 2 == 0:
            residual_layers.append(x)

    # 디코더 시작
    for i in range(num_layers-1):
        if i % 2 == 1:
            x = tf.keras.layers.Add()([x, residual_layers.pop()])
            x = tf.keras.layers.Activation('relu')(x)
        x = deconv_layers[i](x)

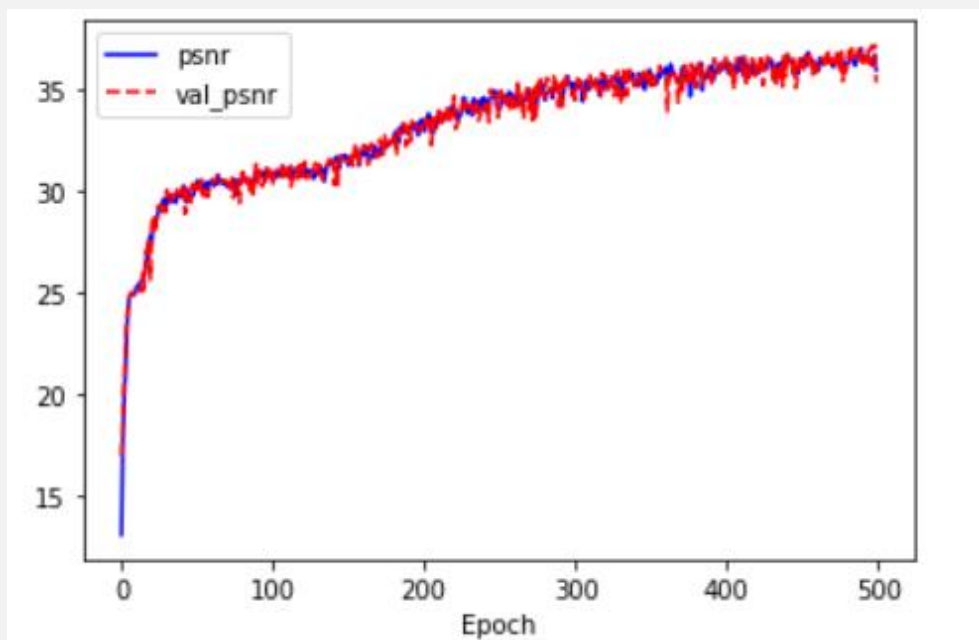
    x = deconv_layers[-1](x)

    model = tf.keras.Model(inputs=inputs, outputs=x)

```

오토인코더 모델중 REDNet 모델을 이용

```
plt.plot(history.history['psnr_metric'], 'b-', label='psnr')  
plt.plot(history.history['val_psnr_metric'], 'r--', label='val_psnr')  
plt.xlabel('Epoch')  
plt.legend()  
plt.show()
```



Psnr 점수 계산

```
img = tf.io.read_file(test_path[0])
img = tf.image.decode_jpeg(img, channels=3)
hr = tf.image.convert_image_dtype(img, tf.float32)

lr = tf.image.resize(hr, [hr.shape[0] // 2, hr.shape[1] // 2])
lr = tf.image.resize(lr, [hr.shape[0], hr.shape[1]])
predict_hr = model.predict(np.expand_dims(lr, axis=0))

print('sr_psnr:', tf.image.psnr(np.squeeze(predict_hr, axis=0), hr, max_val=1.0))
print('lr_psnr:', tf.image.psnr(lr, hr, max_val=1.0))
```

```
sr_psnr: tf.Tensor(24.14708, shape=(), dtype=float32)
lr_psnr: tf.Tensor(27.810328, shape=(), dtype=float32)
```

```
plt.figure(figsize=(16, 4))

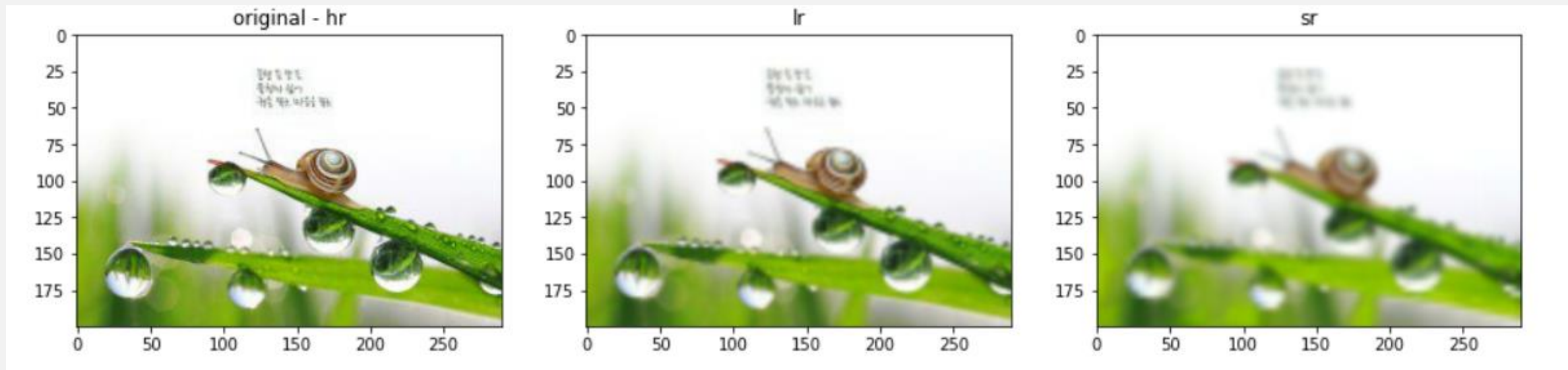
plt.subplot(1, 3, 1)
plt.imshow(hr)
plt.title('original - hr')

plt.subplot(1, 3, 2)
plt.imshow(lr)
plt.title('lr')

plt.subplot(1, 3, 3)
plt.imshow(np.squeeze(predict_hr, axis=0))
plt.title('sr')

plt.show()
```

실제 사진 비교



002-3

머신러닝 : 협업 필터링



협업 필터링 이란?

- 사용자와 항목의 유사성을 동시에 고려해 추천
- 기존에 내 관심사가 아닌 항목이라도 추천 가능
- 학습과정에 나오지 않은 항목은 추천불가

머신러닝 알고리즘

```
for (let user_idx=0; user_idx<allUserId.length; user_idx++) {  
  matrix[user_idx] = new Array(allPhotoId.length).fill(0);  
  const likes = await client.like.findMany({  
    where : {  
      userId : allUserId[user_idx]  
    },  
    select : {  
      photoId : true,  
    }  
  })  
  const likeIdList = likes.map(obj=>obj.photoId);  
  for (let idx=0; idx<likeIdList.length; idx++){  
    matrix[user_idx][allPhotoId.indexOf(likeIdList[idx])] = 1;  
  }  
}  
// make matrix
```



```
const me_idx = allUserId.indexOf(loggedInUser.id);
let result = [];
for (let user_idx=0; user_idx<allUserId.length; user_idx++){

    if (user_idx!==me_idx){
        //where algorithm used
        result[user_idx] = [allUserId[user_idx], cosinesim(matrix[me_idx], matrix[user_idx]) || 0];
    }
}
result = result.filter(Boolean)
result.sort(function(b, a) {
    return a[1] - b[1];
}); //result = adjacent users sorted by cosine similarity
console.log(result);
```

```
1  export function cosinesim(A,B){
2      var dotproduct=0;
3      var mA=0;
4      var mB=0;
5      for(let i = 0; i < A.length; i++){
6          dotproduct += (A[i] * B[i]);
7          mA += (A[i]*A[i]);
8          mB += (B[i]*B[i]);
9      }
10     mA = Math.sqrt(mA);
11     mB = Math.sqrt(mB);
12     var similarity = (dotproduct)/((mA)*(mB))
13     return similarity;
14 }
```

```
{ 10, 1, username: 'manager' }
[
  [ 10, 0.6708203932499369 ],
  [ 32, 0.6708203932499369 ],
  [ 11, 0.5976143046671968 ],
  [ 2, 0.5656854249492379 ],
  [ 3, 0.5163977794943223 ],
  [ 9, 0.5163977794943223 ],
  [ 14, 0.4743416490252569 ],
  [ 13, 0.3651483716701107 ],
  [ 12, 0.31622776601683794 ],
  [ 16, 0.31622776601683794 ],
  [ 30, 0.22360679774997896 ],
  [ 31, 0.22360679774997896 ],
  [ 15, 0 ],
  [ 17, 0 ],
  [ 18, 0 ],
  [ 19, 0 ],
  [ 20, 0 ],
  [ 21, 0 ],
  [ 22, 0 ],
  [ 23, 0 ],
  [ 24, 0 ],
  [ 25, 0 ],
  [ 26, 0 ],
  [ 27, 0 ],
  [ 28, 0 ],
  [ 29, 0 ]
]
```

```
export default {
  Query: {
    seePhoto: protectedResolver(async(_, {list, offset}, props) => {
      if (list.length < 1) {
        return null
      }
      const photolist = await client.photo.findMany({
        take: 1,
        skip: offset,
        where: {
          id: { in: list }
        }
      })
      return photolist
    })
  }
}
```

```
[
  14, 22, 23, 24, 25, 26, 27, 28,
  29, 30, 31, 32, 33, 34, 35, 36,
  37, 1342, 1343, 1344, 1345, 1346, 1347, 1348,
  1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356,
  1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364,
  1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372,
  1373, 1374
]
```

머신러닝 알고리즘

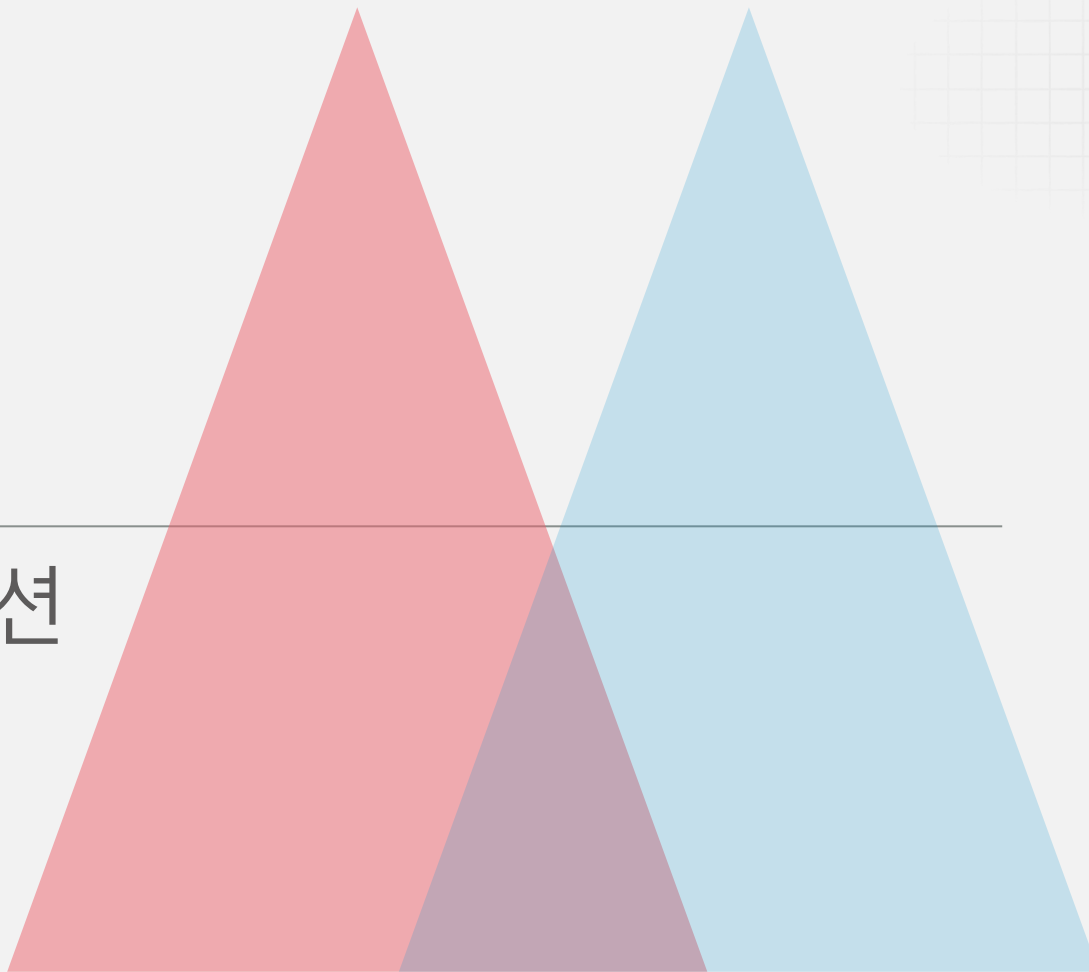
```
const PHOTO_RECOMMEND = gql`
  query getAdjacents{
    getAdjacents
  }
`;

const PHOTO_QUERY = gql`
  query seePhoto( $list : [Int], $offset: Int!) {
    seePhoto(list: $list, offset: $offset) {
      id
      user{
        id
        username
      }
      file
      comments{
        id
      }
      commentNumber
      likesNumber
      isLiked
    }
  }
`;
```

```
const { data : recommendData, loading } = useQuery(PHOTO_RECOMMEND)
const photoList = recommendData?.getAdjacents
const { data : userData } = useMe();
const { data, loading : renderingPhoto, refetch, fetchMore } = useQuery(PHOTO_QUERY, {
  variables : {
    offset : 0,
    list : photoList,
  }
})
```

003

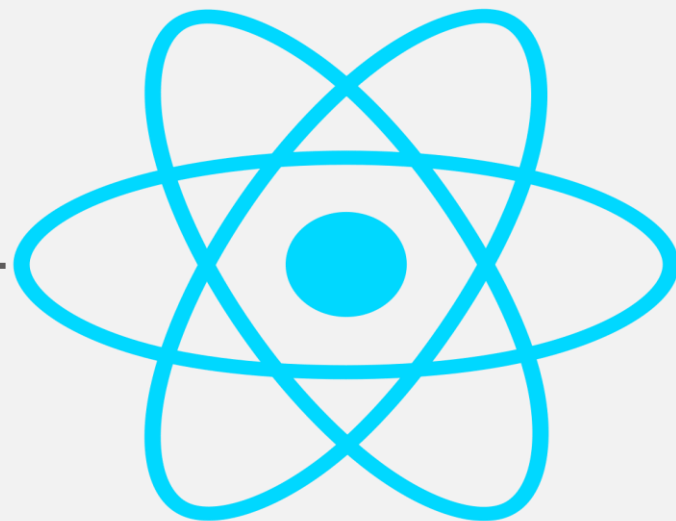
애플리케이션
소개



<exp://mr-hja.gwihwan-go.frontend.exp.direct:80>

The Apollo logo, featuring a stylized 'A' inside a circle followed by the word 'APOLLO' in a bold, sans-serif font.The GraphQL logo, featuring a pink geometric icon of a hexagon with internal lines and the word 'GraphQL' in a pink, sans-serif font.A Venn diagram consisting of three overlapping circles: a white circle on the left, a light blue circle on the top right, and a light orange circle on the bottom right. A yellow square with the text 'JS' is positioned in the center where all three circles overlap. Lines connect the Apollo logo to the white circle, the GraphQL logo to the orange circle, and the Expo logo to the orange circle. The React Native logo is positioned to the right of the blue circle.

JS



React Native

The Expo logo, featuring a blue and white icon of a stylized 'A' or book shape followed by the word 'Expo' in a bold, black, sans-serif font.

문구반점

새 계정 만들기

로그인



문구반점

First Name

Last Name

Username

Email

Password

Check Your Password Again

Create Account

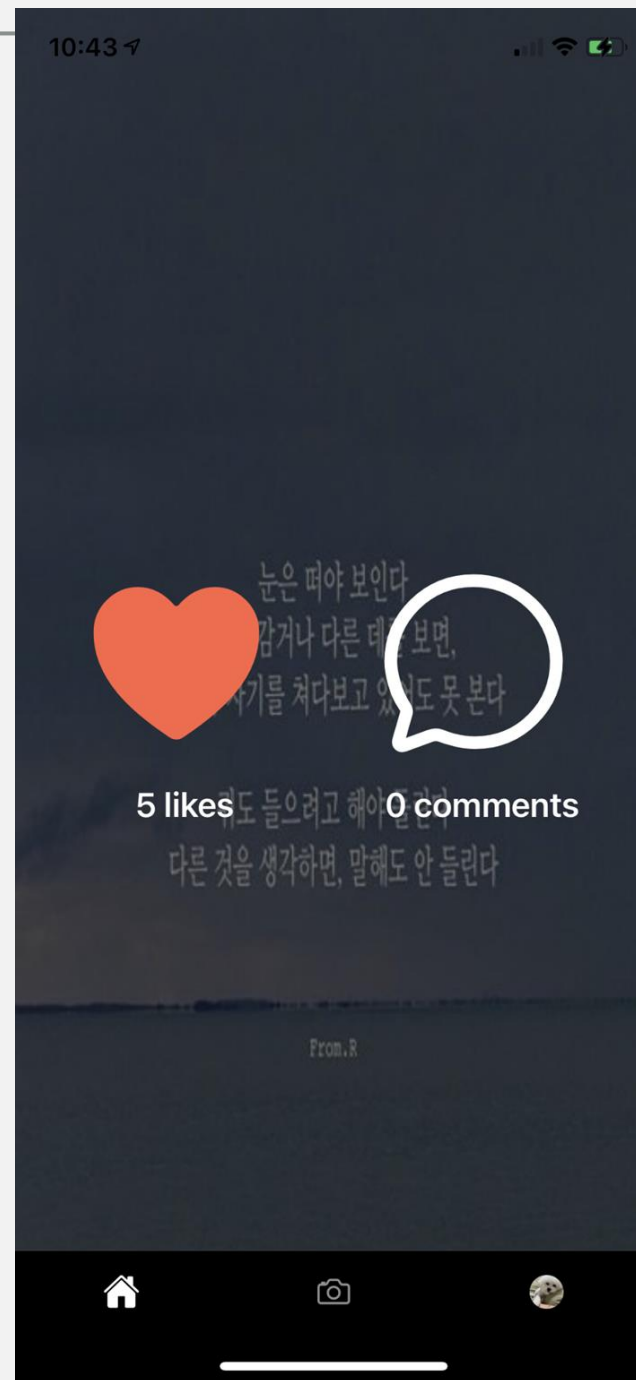
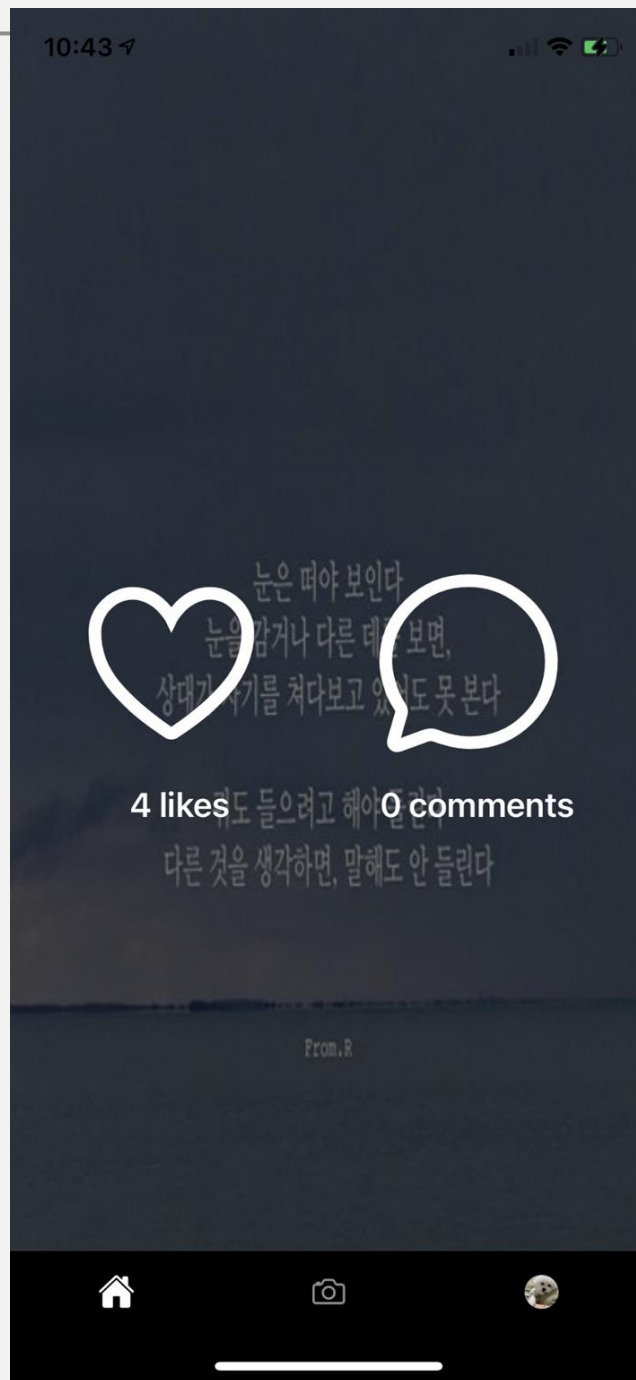
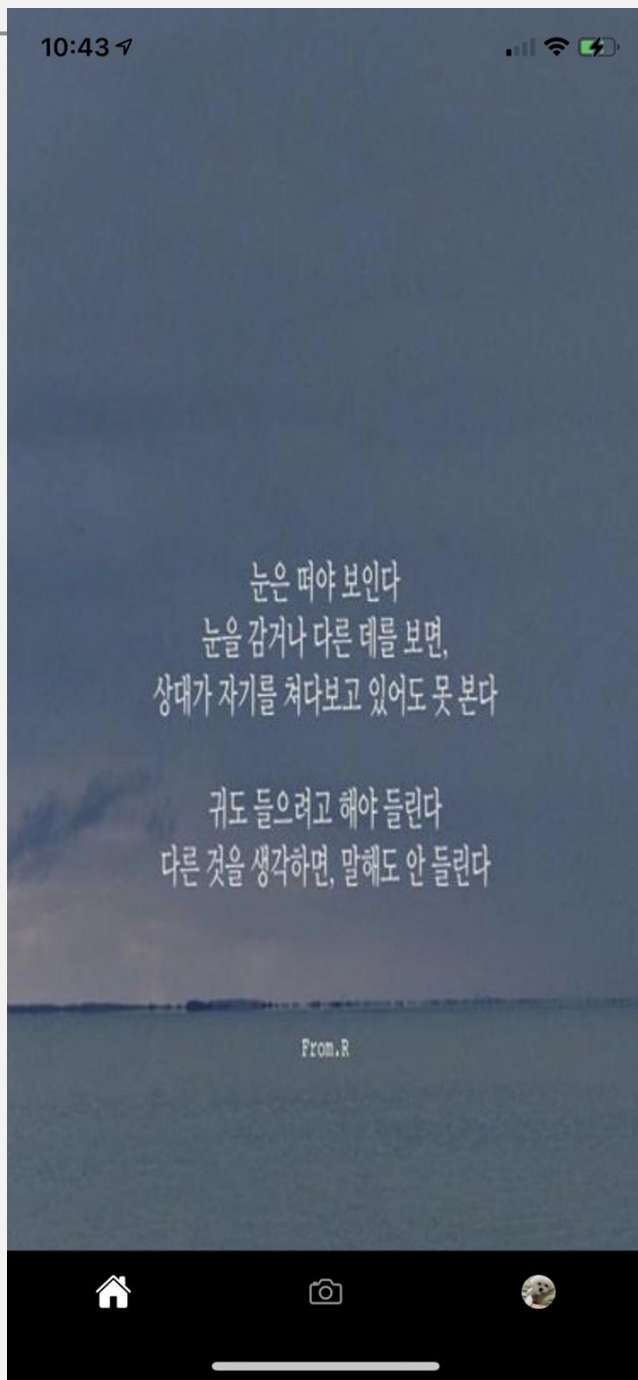


문구반점

아이디를 입력해주세요.

비밀번호를 입력해주세요.

로그인



문구반점

15
Seen11
Likes135
Upload

Gwihwan Go

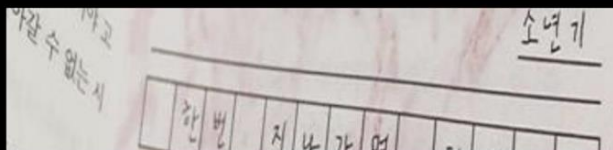
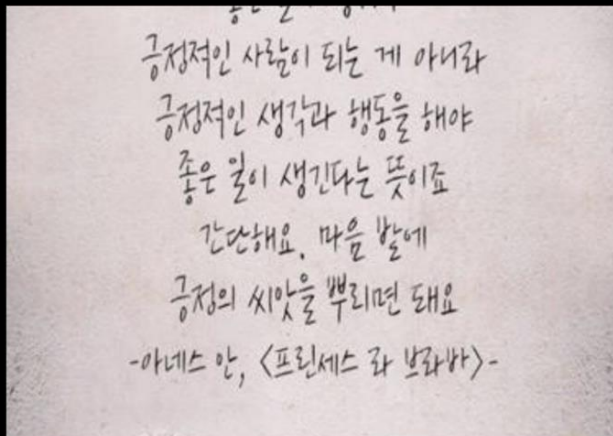
I'm the main developer of moon9banjeom project

Edit Profile(developing...)

SEEN

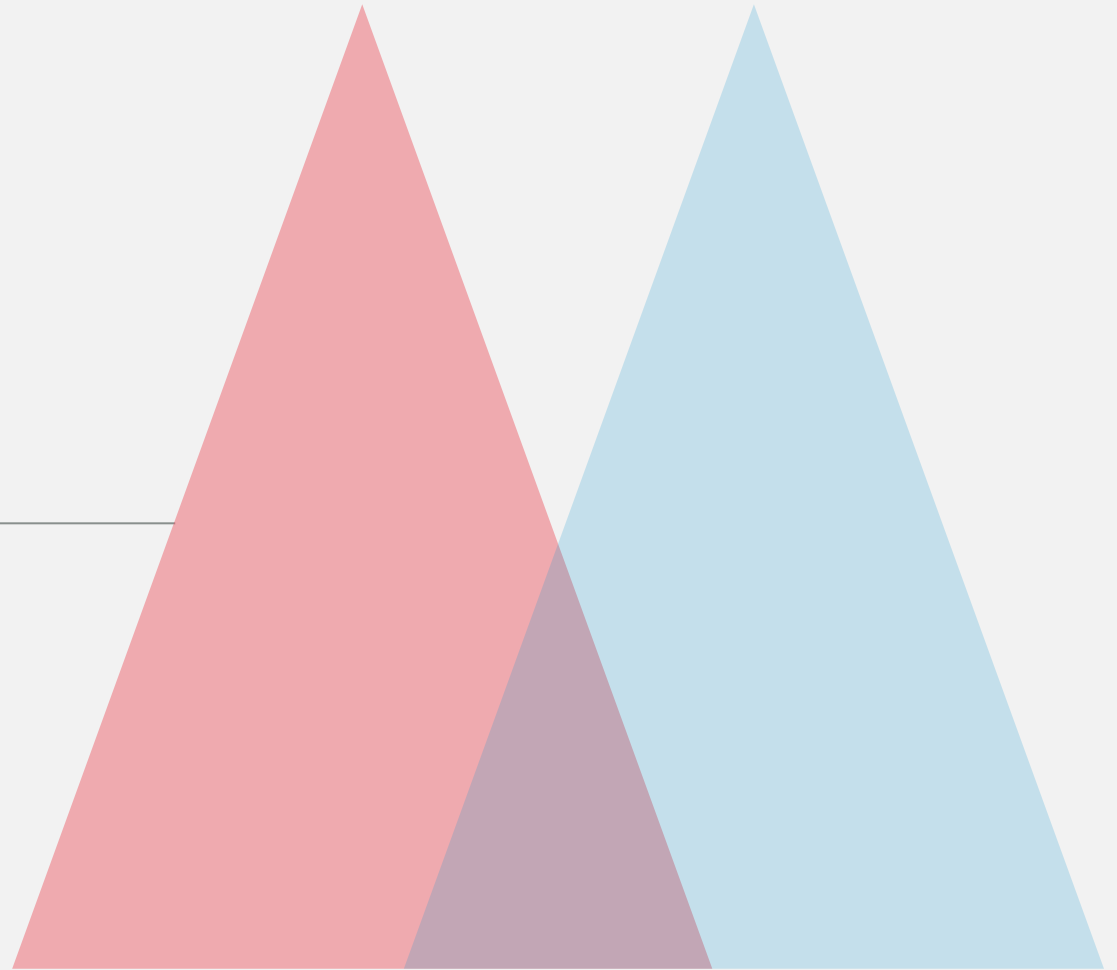
LIKES

UPLOAD



004

차후 계획



차후 계획

1. 저작권 문제 해결
2. 미완성된 기능 구현
 - a. 프로필 수정
 - b. 사진 업로드 기능
3. 유저의 자체적인 사진 업로드를 독려하고 유저 자체 share-approval 의 block-chain like 커뮤니티화가 목표
 - a. 유저에게 회원 등급 부여
 - b. 유저가 업로드한 사진은 다른 유저들의 승인을 받아야 하는데, 유저의 등급이 높으면 강한 승인 권한을 행사 가능함