

About Me:



CONTACT

최종 학력	전남대학교 심리학과
Phone	010-2295-3824
E-mail	gom_55@naver.com
Github	https://github.com/maxman999

1

빅데이터 분석 서비스 개발자 과정 수료

- K-디지털 핵심 실무인재 양성사업 - 빅데이터 분석서비스 개발자 과정
- 2020. 11. 02 ~ 2021.04.12 (928h)
- (사)스마트인재개발원

Team
project

2

CNN 모델 전이학습을 활용한 반려식물 종합 관리 서비스

- 빅데이터 분석서비스 개발자 과정 최종 융합 프로젝트
- 프로젝트 기간 : 2021.03.16 ~ 2021.04.12

Team
project

3

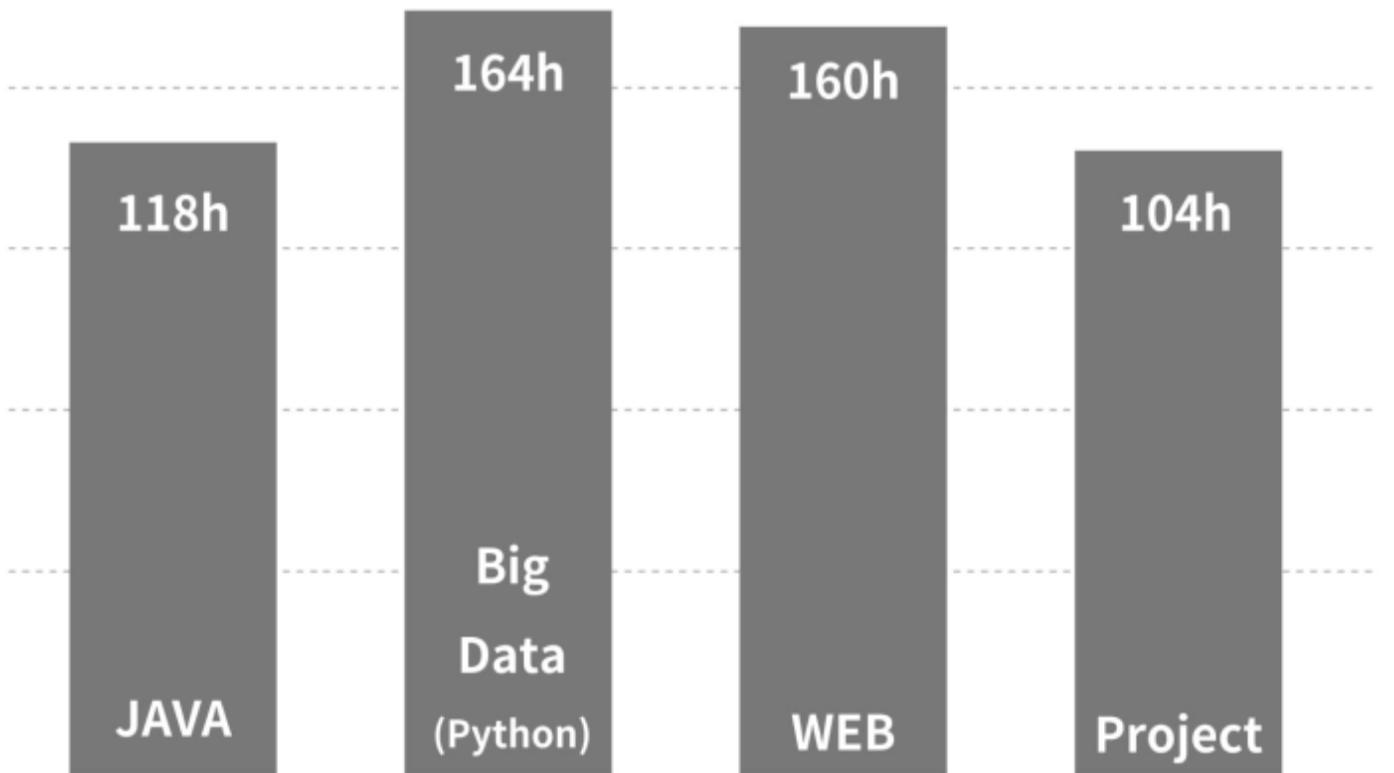
텍스트 마이닝을 활용한 임산부 정보 제공 및 육아수첩 제작 서비스

- 빅데이터 분석서비스 개발자 과정 핵심역량 강화 프로젝트
- 프로젝트 기간 : 2021.01.27 ~ 2021.02.08

빅데이터 분석 서비 스 개발자 과정 수료

소감 :

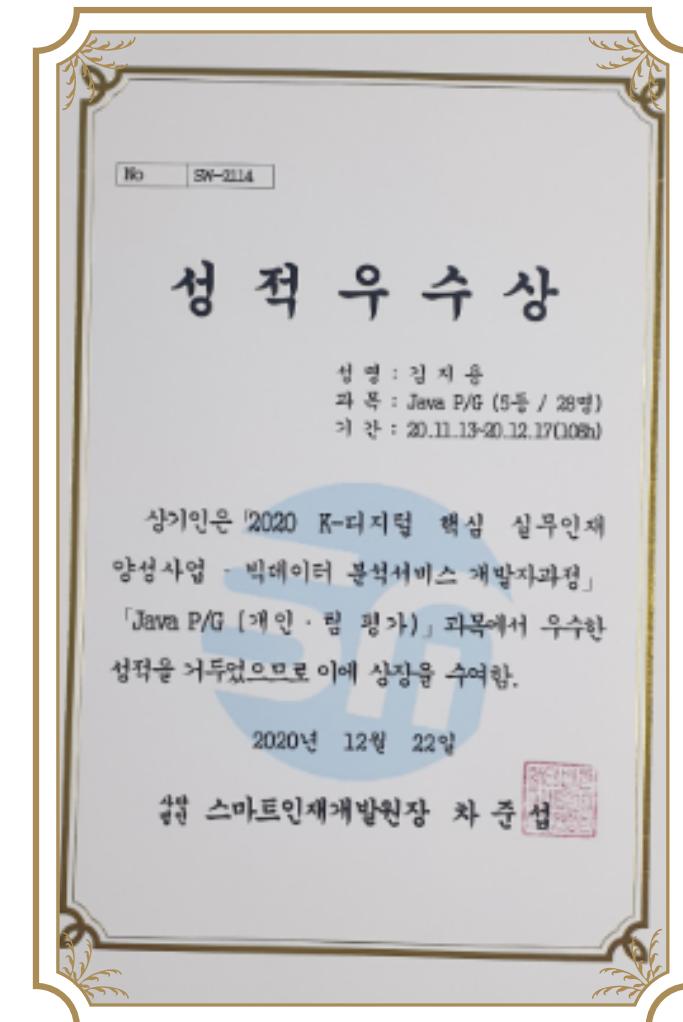
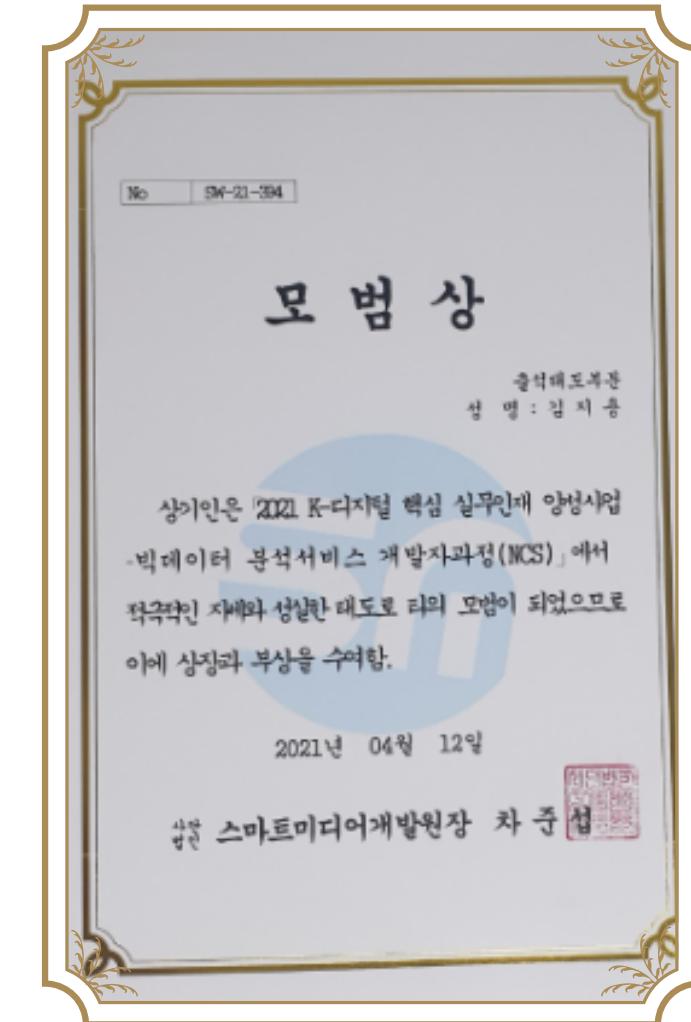
2020.11.02 ~ 2021.04.12



Passion

노력은 배신하지 않는다!

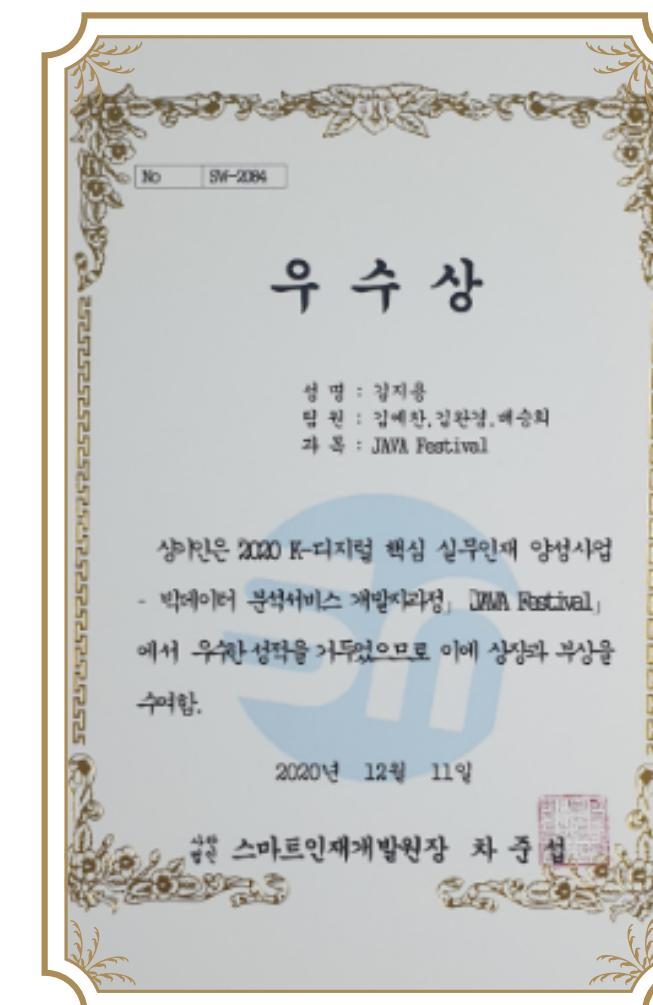
소프트웨어 개발이라는 생소한 분야에 처음 도전했기 때문에 처음에는 두렵기도 했지만, 노력은 배신하지 않는다는 신념에 따라 5개월간 쉬지 않고 달려왔습니다. 시작은 어리숙했지만, 끝에 가서는 손에 꼽히는 좋은 성적을 받았습니다. 개발의 세계에서 끊임없이 정진하는 것이 얼마나 중요한지 배울 수 있었습니다.



Communication

협력과 소통의 중요성

세 차례의 크고 작은 팀 프로젝트를 진행하면서 협력과 소통의 중요성을 배울 수 있었습니다. 팀원들과 함께 고생하며 만족할 만한 성과를 올렸을 때의 기쁨은 이루 말할 수가 없었습니다. 단순히 스킬만 좋은 개발자가 아닌 팀에 필요한 개발자가 되어야겠다고 다짐할 수 있었습니다.



CNN 모델 전이학습을 활용한 반려식물 종합 관리 서비스 :

2021.03.16 ~ 2021.04.12

김지용(팀장), 박서진, 권익주, 김혜선, 정우현

https://github.com/maxman999/smhrd_3rdProject

프로젝트 소개

코로나19의 영향과 홈 인테리어 열풍이 맞물려 현재 가드닝 시장은 급격하게 성장하고 있다. 초보 가드너들이 늘고 있는 지금. 화초 식별(가드닝 정보 제공)과 질병 진단 기능을 통해 초보 가드너들의 취미생활을 지원하고자 한다.

역할

프로젝트 기획 및 총괄
스프링 MVC 프레임워크를 활용한 회원 CRUD 구현
농사로 OPEN API를 활용한 가드닝 정보 제공 기능 구현
반려식물 식별 모델 구축
Git/GitHub를 활용한 형상관리 및 모듈통합

The screenshot shows a mobile application interface for plant identification. At the top, there's a navigation bar with icons for search, service introduction, login, and sign up. The main header reads "SeekSick". Below the header, a question "당신의 화초 얼마나 건강한가요?" (How healthy is your plant?) is displayed. Underneath it, a sub-question "AI로 화초의 건강을 체크하세요." (Check the health of your plant with AI) is shown. Two buttons are present: "식물 식별" (Plant Identification) and "질병 진단" (Disease Diagnosis). The background features a blurred image of a potted plant.

This screenshot shows a detailed view of a plant profile. On the left is a photograph of a Sansevieria trifasciata (Snake Plant) in a white pot, with a circular icon overlaid showing a camera and a checkmark. To the right of the photo is the plant's name "산세베리아" (Sansevieria trifasciata) and its scientific name "Sansevieria trifasciata". Below this is a section titled "가드닝 TIP!" (Gardening Tip!) containing a table with the following information:

생육온도	21~25°C
겨울 최저 온도	13°C 이상
비료정보	비료를 거의 요구하지 않음
물주기	토양 표면이 말랐을 때 충분히 관수함

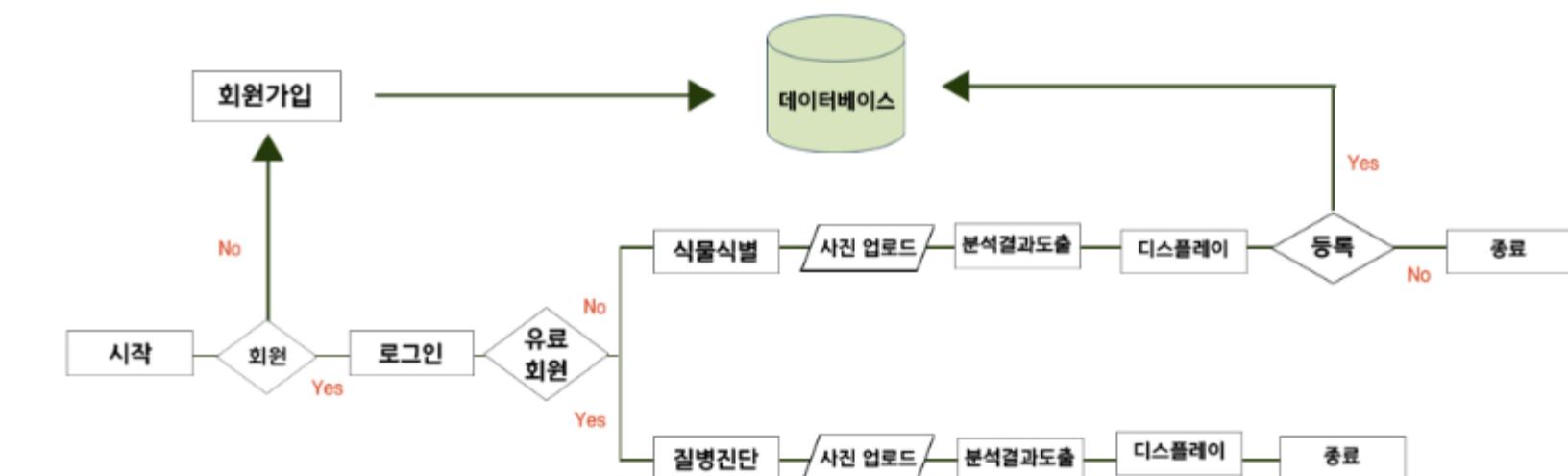
At the bottom of the screen are two buttons: "홈으로" (Home) and "등록" (Register).

1. Briefing

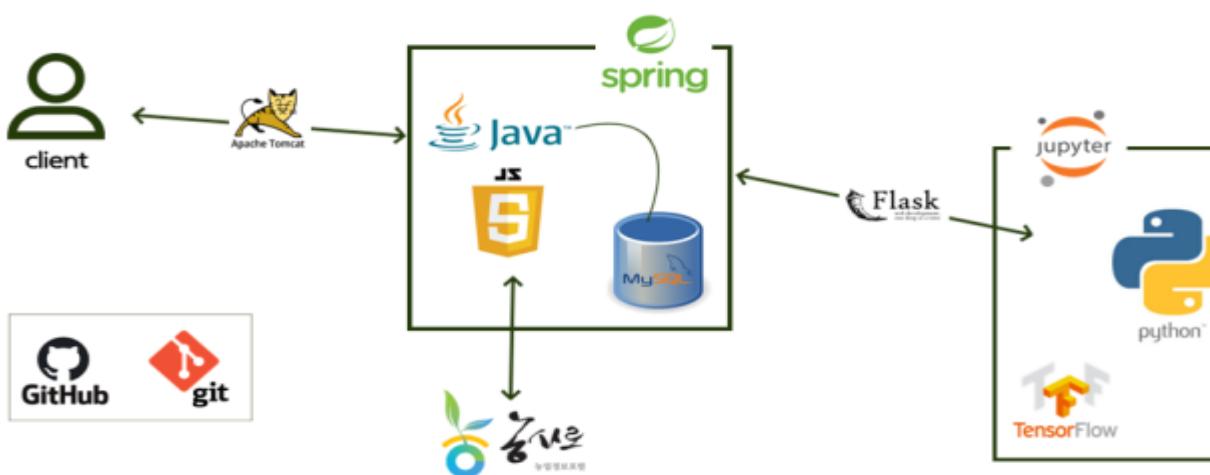
개발 일정

추진 일정	3월														4월												
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12
회의 및 주제 선정																											
기획안작성																											
주요기능 설계																											
데이터 수집																											
주요기능 구현																											
발표 준비																											

서비스 흐름도



프로그래밍 구성도



DB 모델링

member		회원		plant		식물	
사용자	id	Domain	varchar(30)	식물	plantid	Domain	number
비밀번호	password	notnull	varchar(20)	사용자	id	Domain	varchar(30)
별명	nickname	notnull,unique	varchar(20)	식물이름	plantname	notnull	varchar2(20)
이메일	email	notnull	varchar(50)	등록일자	regidate	notnull	date
전화번호	phone	notnull	varchar(20)	사진	picture	notnull	varchar(500)
프리미엄여부	premium	notnull	number	병역	sick	Domain	varchar(30)
				식물별명	plantnick	Domain	varchar(20)

2. 회원관리 기능 구현 - URL 흐름

SeekSick

회원가입

아이디 아이디를 입력해주세요 중복확인

비밀번호 비밀번호를 입력해주세요

닉네임 닉네임을 입력해주세요

이메일 이메일을 입력해주세요

전화번호 전화번호를 입력해주세요

회원가입 (/myapp/join)

SeekSick

localhost:8081 내용:
사용할 수 없는 아이디입니다.

회원가입

아이디 rlawldyd 중복확인

ID중복확인(/myapp/idCheck)

SeekSick

당신의 화초 얼마나 건강한가요?

AI로 화초의 건강을 체크하세요.

식물 식별 질병 진단

메인화면(/myapp/main)

로그인

아이디 rlawldyd

비밀번호 ****

로그인 (/myapp/login)

SeekSick

마이페이지

아이디 rlawldyd

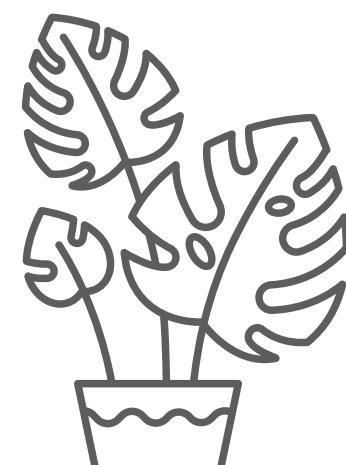
비밀번호* 123

닉네임 김지웅

이메일 ggm_10@gmail.com

전화번호 01022353834

수정/탈퇴 (myapp/mypage)



2. 회원관리 기능 구현 - 코드 예시

코드는 회원 관리 기능 중 다소 까다로웠다고 생각한 중복 확인 기능만 예시로 들었습니다. 전체 코드는 프로젝트 소개 페이지의 깃허브 주소에서 확인하실 수 있습니다.

join.jsp

```
<script>
window.onload = function(){
    /* 중복 확인을 위한 idCheck 함수 선언 */
    let idCheck = id => {
        /* 비동기 방식으로 서버와 통신하여 중복 체크 */
        const xhr = new XMLHttpRequest();
        xhr.open('GET', "/myapp/memberIdCheck.do?id=" + id);
        xhr.setRequestHeader('Content-Type', 'application/xml');
        xhr.getResponseHeader('Access-Control-Allow-Methods: POST, GET, OPTIONS');
        xhr.send();
        xhr.onload = () => {
            if (xhr.status === 200) {
                let result = xhr.response;
                if (result === "1") {
                    alert("사용할 수 없는 아이디입니다.");
                    /* 중복될 경우 가입이 안되도록 submit 버튼 잠금 */
                    document.getElementById('join_submit').setAttribute('disabled', 'disabled');
                } else {
                    alert("사용할 수 있습니다.");
                    document.getElementById('join_submit').removeAttribute('disabled');
                }
            } else {
                console.error('Error', xhr.status, xhr.statusText);
            }
        };
    };
    /* 중복확인 버튼을 클릭하면 idCheck함수 실행 */
    document.getElementById("idCheck").addEventListener('click', function(){
        let id = document.getElementById("id4check").value;
        idCheck(id);
    });
};
</script>
```

사용자가 버튼을 눌러 중복확인을 요청하면 XMLHttpRequest 객체를 이용해 서버와 비동기 통신을 시작한다. 통신 결과에 따라 사용자에게 중복여부를 알려준 뒤, 사용할 수 있는 아이디인 경우 가입 버튼을 활성화 해 가입 할 수 있도록 설정했다.

AJAX

MemberController

```
@ResponseBody
@RequestMapping("/memberIdCheck.do")
public int memberIdCheck(String id) {
    System.out.println(id);
    int chk = dao.memberIdCheck(id);
    System.out.println(chk);

    return chk;
}
```

컨트롤러에서 memberIdCheck.do에 해당하는 요청을 받으면 DAO의 memberIdCheck 메소드를 호출한다. 해당 요청에 대해서는 @ResponseBody 어노테이션을 이용해 화면전환을 하지 않고 반환값을 HTTP의 본문(body)부분에 담아서 보내주었다.

MemberDAO

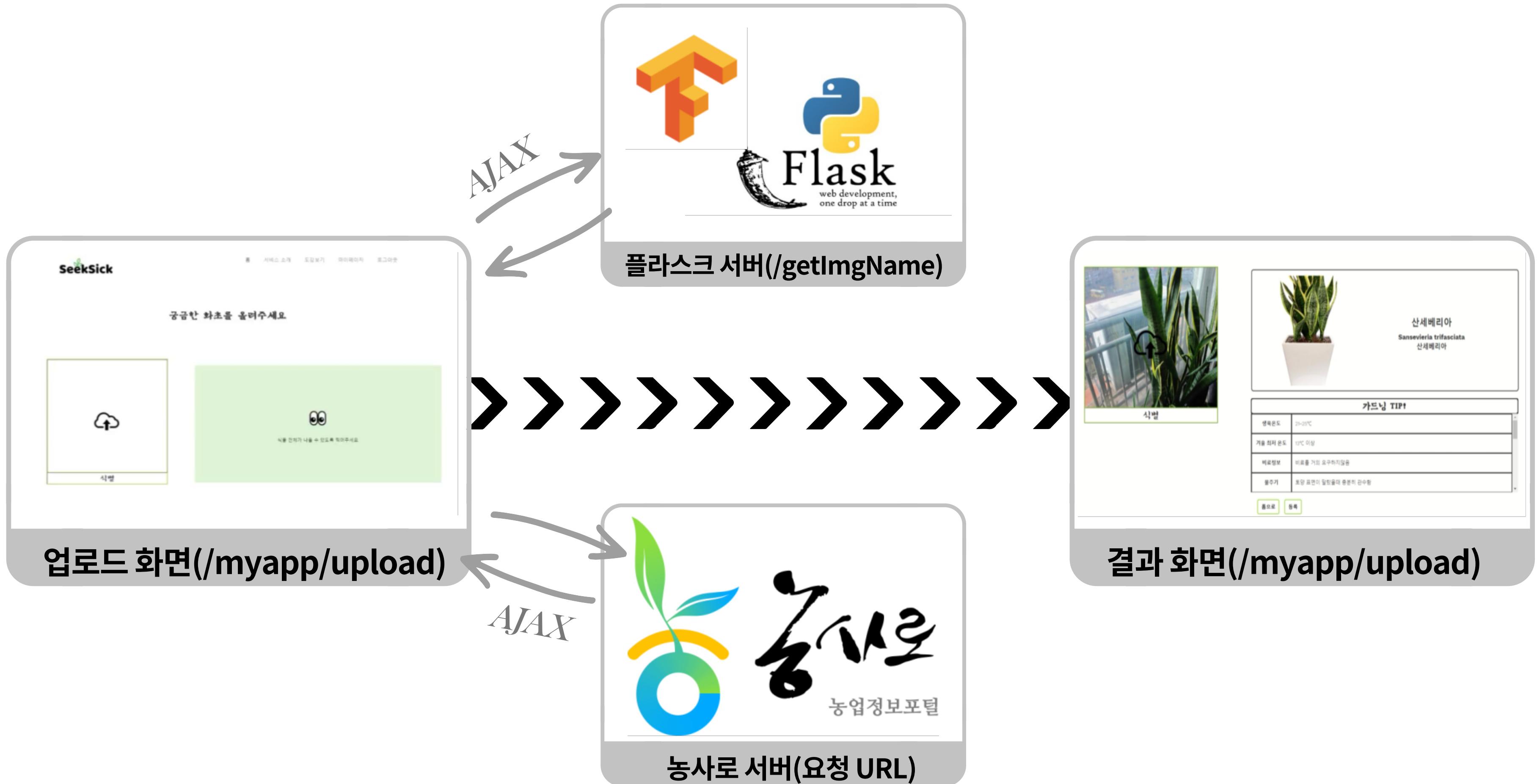
```
@Override
public int memberIdCheck(String id) {
    SqlSession session = sqlSessionFactory.openSession();
    int chk = session.selectOne("memberIdCheck", id);
    System.out.println(chk);
    session.close();
    return chk;
}
```

해당 메소드가 호출 되면, 스프링 컨테이너에 등록해 놓던 sqlSessionFactory 객체를 이용해 jdbc를 수행한다.

MemberMapper.xml

```
<select id="memberIdCheck" parameterType="String" resultType="int">
    select COUNT(*) from member where id = #{id}
</select>
```

2. 가드닝 정보제공 기능 구현 - URL 흐름



2. 가드닝 정보제공 기능 구현 - 코드 예시

upload.jsp

```
$('#btnUpload').on('click', function(event) {
    event.preventDefault();
    /* 유저의 사용성을 위한 로딩중 화면 처리 */
    loadingStart();
    /* 사용자가 올린 사진 데이터 변수에 저장 */
    var form = $('#uploadForm')[0];
    var data = new FormData(form);
    $('#btnUpload').prop('disabled', true);
    /* getName함수를 통해 플라스크 서버에서 받아온 식물번호(이름)을 저장 */
    const targetNum = getName();
    targetNum.then(pname => {
        /* 식별이 끝난 식물의 이름을 화면에 출력 */
        document.getElementById("plant_name").innerText = pname.plantName;
        /* 농사로에서 받아온 정보 중 화면에 띄울 것만 배열에 저장 */
        const infoarr = ["plntbneNm", "distbNm", "grwhTpCodeNm", "winterLwetTpCodeNm", "frtlzrInfo", "watercycleSprngCodeNm", "lightDemanddoCodeNm", "posti"];
        const infoarrk = ["유동명", "식물학명", "생육온도", "겨울최저온도", "비료정보", "물주기", "광요구도", "배치장소", "특별관리 정보", "기능성 정보", "조언 정보"];
        /* getInfo 함수를 통해 식별이 끝난 식물의 가드닝 정보를 농사로 서버에 요청 */
        const result = getInfo('http://api.nongsaro.go.kr/service/garden/gardenDtl?apiKey=20210325ZS1OCEZBQCK8HV5TOYGQUQ&cntntsNo=' + pname.plantNum);
        result.then(pdata => {
            /* 받아온 가드닝 정보를 화면에 출력 */
            for (var i = 0; i < infoarr.length; i++) {
                plantInfo = pdata.getElementsByTagName(infoarr[i])[0].innerHTML;
                plantInfo = plantInfo.replace("<![CDATA[", "").replace("]]>", "");
                document.getElementById("info"+i).innerText = plantInfo;
                document.getElementById("info_title"+i).innerText = infoarrk[i];
            }
            /* 샘플사진 변경 */
            document.getElementById("ex_img").src = "./resources/images/img_samples/" + pname.plantNum + "/" + pname.plantNum + "(1).jpg";
            document.getElementById("autoKeyword").style.display = "none";
        })
    })
})
```

upload.jsp에서 사용자가 버튼을 눌러 화초 식별을 요청함으로써 기능이 시작된다. getName 함수를 호출하여 플라스크 서버와 연동한 뒤, 플라스크 서버에서 분석한 결과값을 getInfo 함수에 전달하여 농사로 서버에서 가드닝 정보를 가져온다.

getName 함수

```
/* 사용자가 업로드한 사진을 딥러닝 모델에 보내주기 위한 함수 */
let getName = () => {
    /* 클백 지옥을 예방하기 위해 Promise 객체를 이용해 비동기 처리 */
    return new Promise((resolve, reject) => {
        $.ajax({
            type: "POST",
            enctype: "multipart/form-data",
            url: "/myapp/fileUpload.do",
            data: data,
            processData: false,
            contentType: false,
            cache: false,
            timeout: 600000,
            success: function (data) {
                $('#btnUpload').prop('disabled', false);
                console.log(data.plantName);
                console.log(data.plantNum);
                /* 딥러닝으로 분석한 결과를 저장 */
                resolve(data);
            },
            error: function (e) {
                $('#btnUpload').prop('disabled', false);
                alert('fail');
            }
        });
    });
}
```

getInfo 함수

```
/* 분석이 끝난 사진 속 화초의 가드닝 정보를 농사로 서버로부터 받아오는 함수 */
let getInfo = url => {
    /* 클백 지옥을 예방하기 위해 Promise 객체를 이용해 비동기 처리 */
    return new Promise((resolve, reject) => {
        const xhr = new XMLHttpRequest();
        xhr.open('GET', url);
        xhr.setRequestHeader('Content-Type', 'application/xml');
        xhr.getResponseHeader('Access-Control-Allow-Methods: POST, GET, OPTIONS');
        xhr.send();
        xhr.onload = () => {
            if (xhr.status === 200) {
                let xstring = xhr.response;
                var parser = new DOMParser();
                var xmlDoc = parser.parseFromString(xstring, "text/xml");
                resolve(xmlDoc);
            } else {
                console.error('Error', xhr.status, xhr.statusText);
            }
        };
    });
}
```

2. 가드닝 정보제공 기능 구현 - 코드 예시



PlantController

```
@ResponseBody  
@RequestMapping("/fileUpload.do")  
public HashMap<String, String> fileUpload(@RequestParam("uploadFile") MultipartFile file,  
    HttpServletRequest request) throws IllegalStateException, IOException, Exception {  
    if (!file.getOriginalFilename().isEmpty()) {  
        // 업로드한 식물 사진 저장  
        file.transferTo(new File(FILE_SERVER_PATH, file.getOriginalFilename()));  
    } else {  
    }  
    // 플라스크 서버에 사용자가 업로드한 사진 정보 전송  
    String result = excutePost("http://127.0.0.1:5000/getImgName", file.getOriginalFilename());  
    // 결과값을 HashMap에 담아서 return  
    System.out.println("받은 결과 : " + result);  
    HashMap<String, String> map = new HashMap<String, String>();  
    String[] resultArr = result.split("/");  
    map.put("plantName", resultArr[0]);  
    map.put("plantNum", resultArr[1]);  
    return map;  
}
```

컨트롤러에서는 upload.jsp에서 보내온 사진데이터의 경로를 추적하여 플라스크 서버로 보내준다. 플라스크 서버와 통신이 끝나면 분석한 결과를 키-값 쌍으로 map에 담아서 다시 upload.jsp에 돌려보낸다.

Flask Server

```
app = Flask(__name__)  
@app.route('/getImgName', methods=['GET', 'POST'])  
def getImgName():  
    result = ""  
  
    if request.method == 'POST':  
        name = request.form['imgName']  
        decoded_data = urllib.parse.unquote(name, encoding='utf-8')  
  
        # 전송받은 이미지 이름을 통해 사진 데이터 불러오기  
        img = image.load_img("C:/eGovFrame-3.9.0/sbs/project_sbs/src/main/webapp/resources/images/{}".format(decoded_data),  
                             target_size=(targetx,targety))  
        # 딥러닝 모델에 입력하기 위한 전처리 작업  
        img = img.convert("RGB")  
        img = img.resize((targetx,targety))  
        data = np.asarray(img)  
        X = np.array(data)  
        X = X.astype("float") / 256  
        X = X.reshape(-1, targetx, targety, 3)  
        # 딥러닝 모델에게 예측  
        pred = model.predict(X)  
        result = np.argmax(pred)  
        # 딥러닝 모델이 분석한 결과값 저장  
        target_name = cls_index[result]  
        print(target_name)  
        # 농사로 API 활용에 필요한 식물 고유 번호를 식물 이름과 함께 return  
        result = plantDic[target_name]  
        myPlant = target_name+"/"+str(result)  
  
    # elif request.method == 'GET':  
    #     print(request.args)  
  
    return myPlant
```

플라스크 서버에서는 미리 학습시킨 모델을 불러와 요청에 대한 분석을 진행한 뒤 그 결과값을 반환해준다.

3. 화초 식별 모델 구축 - 데이터 수집 및 전처리



데이터 수집

```
# 구글 크롤링

searchName = "디펜바카리아" # 수집할 식물 이름
crawl_num = 1000 # 수집할 데이터 개수

# 사진 데이터를 저장할 경로 설정
savePath = "./구글 {}".format(searchName)
try:
    if not os.path.exists(savePath):
        os.makedirs(savePath)
except OSError:
    print ('Error: Creating directory. ' + savePath)

# selenium 라이브러리를 이용한 웹 크롤링
driver = webdriver.Chrome()
driver.get("https://www.google.com/search?q={}&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjM39a06aLwAhULJzQIHcQB_gQ_AUoAXoECAEQAw&t")
time.sleep(3)
for i in range(20):
    driver.find_element_by_css_selector('body').send_keys(Keys.PAGE_DOWN)
    time.sleep(1)
html = driver.page_source
soup = bs(html, 'lxml')
n = 1
# 이미지 저장
for i in soup.select(".rg_i"):
    imgUrl = i['src']
    with urllib.request.urlopen(imgUrl) as f:
        with open(savePath + '/' + searchName + str(n)+'.jpg','wb') as h: # w - write b - binary
            img = f.read()
            h.write(img)
    n += 1
    if n > crawl_num:
        break
```

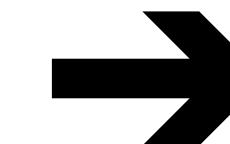
구글을 포함한 여러 사이트에서 웹 크롤링을 실시해 식물 20개 클래스에 대하여 약 14,000개의 사진 데이터를 확보 할 수 있었다.

데이터 전처리 및 증식

```
# 데이터 전처리 및 증식
train_datagen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=30,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    shear_range=0.2,
                                    zoom_range=0.2,
                                    horizontal_flip=True,
                                    fill_mode='nearest')
val_datagen = ImageDataGenerator(rescale=1./255)

trainGen = train_datagen.flow_from_directory(
    os.path.join(rootPath, 'train'),
    target_size = target_size,
    subset='training',
    shuffle=True,
    class_mode = 'categorical'
)
validationGen = val_datagen.flow_from_directory(
    os.path.join(rootPath, 'val'),
    target_size = target_size,
    shuffle=True,
    class_mode = 'categorical'
)
```

성능이 좋은 모델을 구축하기에는 데이터가 부족하다고 생각했기 때문에 ImageDataGenerator를 이용해 이미지 증식을 적용했다.



3. 화초 식별 모델 구축 - 학습 및 파라미터 튜닝

전이학습 및 미세조정

```
# imagenet으로 학습한 InceptionV3모델 불러오기
inceptionv3 = InceptionV3(include_top=False, weights='imagenet', input_shape=IMG_SIZE)
# 모델 동결
for layer in inceptionv3.layers[:289]:
    layer.trainable = False
# 미세조정
y = inceptionv3.output
y = GlobalAveragePooling2D()(y)
y = BatchNormalization()(y)
y = Dropout(0.5)(y)
y = Dense(1024, activation="relu")(y)
y = BatchNormalization()(y)
y = Dropout(0.5)(y)
y = Dense(512, activation="relu")(y)
predictions2 = Dense(20, activation="softmax")(y)
# 최종 모델 생성
model1 = Model(inputs=inceptionv3.input, outputs=predictions2)
model1.summary()
```

화초 식별에 사용된 모델로 이미지 처리에 특화된 CNN 기반의 모델인 VGG16, resnet50, inceptionV3를 사용했으며, 그 중 가장 높은 정확도를 보였던 InceptionV3를 최종 모델로 선정했다. 이미지넷에서 훈련된 가중치를 전이학습 시켜 데이터 부족 문제를 해소하고 모델의 성능을 개선시킬 수 있었다. 미세조정을 거쳐 최종적으로 20개 클래스에 대하여 약 90%의 정확도로 식물을 분류할 수 있는 모델을 구축했다.

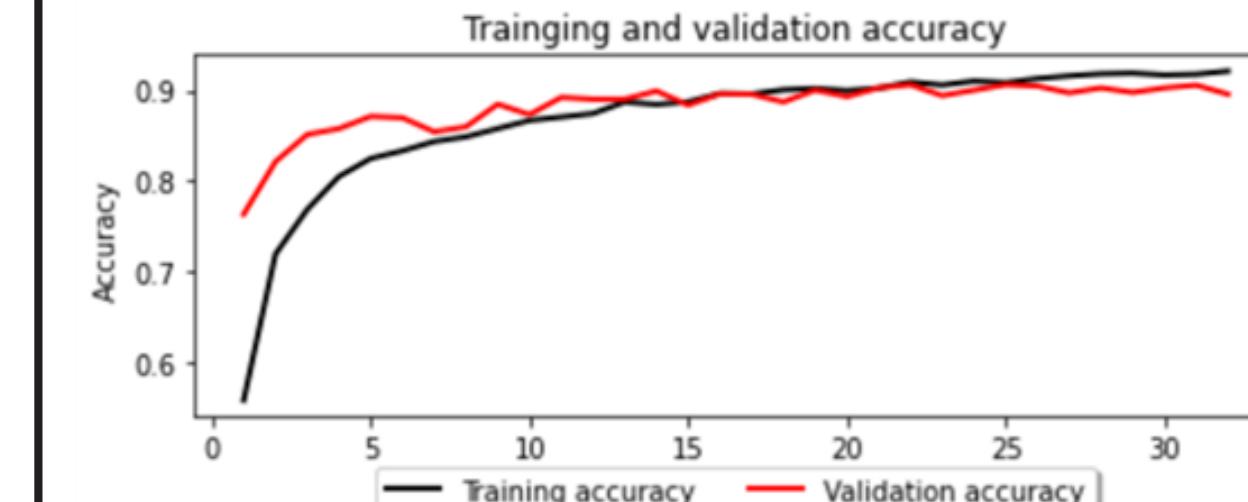
학습

```
# 모델 컴파일
model1.compile(optimizer='adam',
                loss='categorical_crossentropy',
                metrics=['acc'])

# ModelCheckpoint와 EarlyStopping 설정
if not os.path.exists(SAVE_DIR):
    os.mkdir(SAVE_DIR)
modelpath = SAVE_DIR + "sbs_{epoch:03d}_{acc:.3f}_{val_acc:.3f}.hdf5"
mc = ModelCheckpoint(filepath = modelpath,
                     monitor = "val_acc",
                     save_best_only = True)
es = EarlyStopping(monitor="val_acc", patience=10)

# 학습 시작
history = model1.fit_generator(
    trainGen,
    epochs=epochs,
    validation_data=validationGen,
    verbose=1,
    callbacks = [mc,es]
)
```

모델 정확도



텍스트 마이닝을 활용한 임산부 정보 제공 및 육아 수첩 제작 서비스 :

2021.01.27 ~ 2021.02.08

박성수(팀장), 김지용, 권익주, 김현준, 정우현

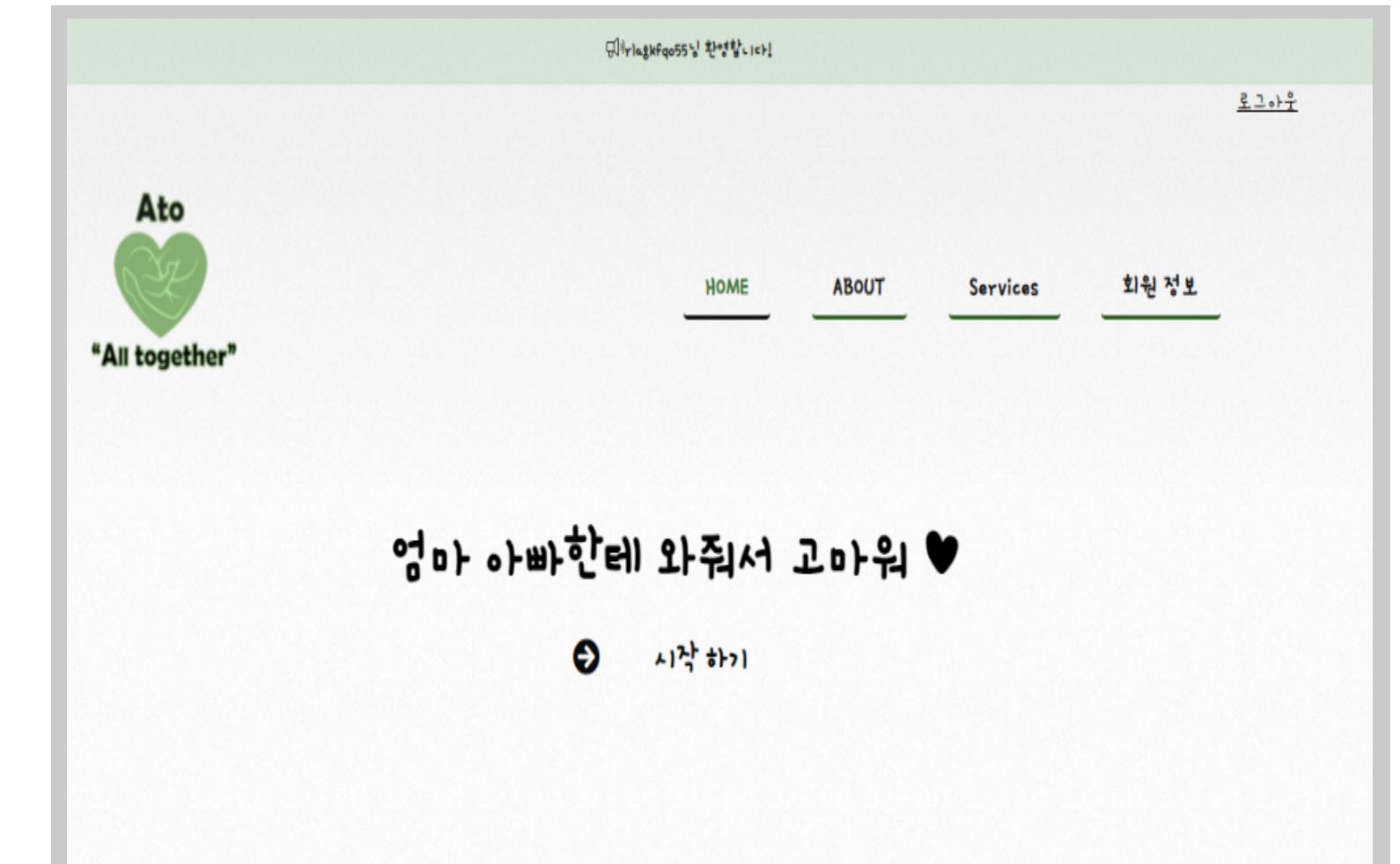
https://github.com/maxman999/smhrd_2ndProject

프로젝트 소개

모든 것이 두렵고 궁금한 임산부들을 위한 맞춤형 정보 제공 서비스
10개월의 소중한 시간을 추억할 수 있는 육아 수첩을 제작해주는 웹 서비스

역할

토픽 모델링을 활용한 키워드 추천 기능 구현
JDBC를 이용한 회원관리 기능 구현
빈도수 기반의 워드클라우드 제작

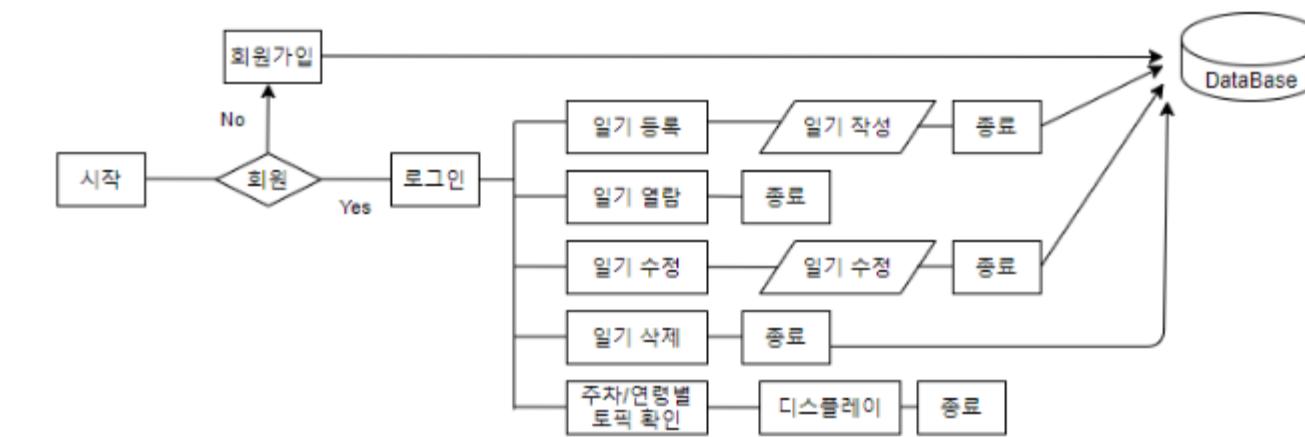


1. Briefing

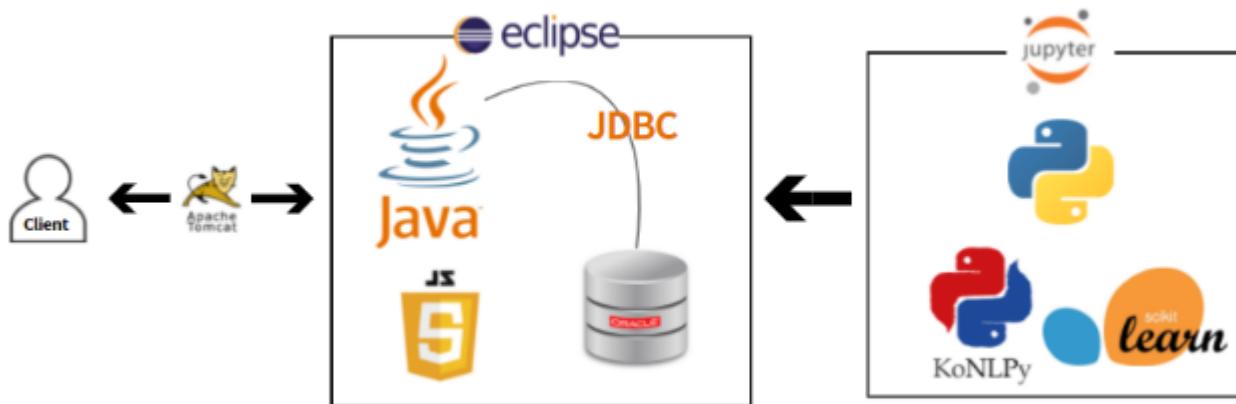
개발 일정

추진 일정	1월					2월							
	27	28	29	30	31	1	2	3	4	5	6	7	8
회의 및 주제 선정													
기획안작성													
주요기능 설계													
데이터 수집													
주요 기능 구현													
발표준비													

서비스 흐름도



프로그래밍 구성도



DB 모델링



2. 토픽모델링을 활용한 주차별 키워드 추천 기능

```
# 크롤링한 말뭉치 불러오기
data = pd.read_csv("C:\Users\SMHRD\2ndPJ\week_result\result.csv", keep_default_na = False)
data = data.get("제목") + " " + data.get("내용")
target = data.astype(str)

# TF-IDF Vectorizing 사용할 Tokenizer 정의
okt = Okt()
def myTokenizer(text):
    return okt.nouns(text)

# TF-IDF 알고리즘을 이용해 가중치가 부여된 단어사전 제작
tfidf = TfidfVectorizer(stop_words = stopwords,
                        tokenizer=myTokenizer,
                        max_features= 1000,
                        ngram_range=(1,3),
                        min_df=2,
                        max_df= 0.95,)

vecdata = tfidf.fit_transform(target)
# LDA(Latent Dirichlet allocation) 알고리즘에 기반한 토픽모델링 수행
lda = LatentDirichletAllocation(n_components=2, learning_method="batch",
                                max_iter = 100, random_state=0)

# 제작한 단어사전을 LDA 모델에 입력
document_topics = lda.fit(vecdata)
```

임산부 커뮤니티에서 크롤링한 게시글 데이터를 임신주차별로 분류한 뒤 TF-IDF 알고리즘을 적용해 단어 사전 제작. 제작한 단어사전을 이용해 LDA 알고리즘에 기반한 토픽모델링 진행



```
# 토픽모델링을 통해 추출한 단어 모음을 기반으로 키워드 도출
sorting = np.argsort(lda.components_, axis=1)[:, ::-1]
feature_names = np.array(tfidf.get_feature_names())
mglearn.tools.print_topics(topics=range(2), feature_names=feature_names,
                           sorting=sorting, topics_per_chunk=2, n_words=10)
```

topic 0	topic 1
진통	유도
이슬	예정일
새벽	제왕
양수	애기
계속	자연
태동	입원
통증	자궁
피	진통
생리통	출산
내진	자연 진통

32주차 토픽

1. 출산 정후

2. 분만 방식

인공지능이 도출한 단어 모음을 토대로 주차별 키워드 추출

3. 빈도수 기반의 워드클라우드 제작

```
# 크롤링한 말뭉치 불러오기
data = pd.read_csv("C://Users//SMHRD//2ndPJ//product_result//product_result.csv", keep_default_na = False)
data = data.get("제목") + " " + data.get("내용1") + " " + data.get("내용2")
data = data.astype(str)
data.isnull().values.any()
# dataframe 형식의 데이터를 배열에 저장
content_list = []
for j in range(len(data)):
    content_list.append(data.iLoc[j])
word_list = data
# 말뭉치를 형태소 분석기에 넣을 수 있도록 정규표현식을 이용해 전처리
content_result = []
for k in range(len(word_list)):
    target = re.sub(only_BMP_pattern, "", str(word_list[k]))
    target = re.sub(han, "", target)
    target = re.sub(mypattern, "", target)
    target.replace("♥", "")
    content_result.append(target)
```

수집한 데이터를 KoNLPy에서 제공하는 형태소 분석기에 넣을 수 있도록 정규표현식을 이용해 전처리 진행



```
# 형태소 분석기(꼬꼬마)를 이용해 일뭉치를 형태소별로 분류
sentences_tag = []
for i in content_result:
    try:
        morph = kkma.pos(i)
        sentences_tag.append(morph)
    except:
        pass
# 분류된 데이터셋 중 명사만 추출하고, 불용어는 제외
noun_list = []
for sentence in sentences_tag:
    for word, tag in sentence:
        if tag in ['NNG'] and word not in stopwords:
            noun_list.append(word)
# 빈도수가 높은 명사 50개를 추출해 워드클라우드로 표현
counts = Counter(noun_list)
tags = counts.most_common(50)
stylecloud.gen_stylecloud(dict(tags),
                           font_path = 'C:\Users\SMHRD\NanumFontSetup_TTF_ALL\NanumBarunGothic.ttf',
                           icon_name='fas fa-heart',
                           palette = "colorbrewer.sequential.Greys_4",
                           gradient = 'vertical',
                           output_name ="age heart{}.jpg".format(i))}
```

전처리가 끝난 데이터 중 명사만 추출한 뒤 빈도수 기반으로 워드 클라우드 제작