

# 종합설계 프로젝트 수행 보고서

프로젝트명	FaceVisitor
팀번호	S2-11
문서제목	수행계획서( ) 2차발표 중간보고서( ) 3차발표 중간보고서( ) 4차발표 중간보고서( ) 최종결과보고서( O )

2019.01.26

팀원 : 김동욱  
허주영

지도교수 : 배유석 (인)

## 문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2020.01.20	김동욱	1.0	수행계획서	최초작성
2020.03.02	김동욱	2.0	본론4 추가	2차 보고서
2020.04.06	김동욱	2.1	내용 보충	재심사대상
2020.05.02	김동욱	3.0	본론5 추가	3차 보고서
2020.06.03	허주영	3.1	상세설계 추가	
2020.06.26	김동욱	4.0	본론 6, 7 추가	4차 보고서
2020.12.04.	김동욱	5.0	결론 추가	최종 결과보고서

## 문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 (1~6) II. 본론 (1~3) 참고자료	I. 서론 (1~6) II. 본론 (1~4) 참고자료	I. 서론 (1~6) II. 본론 (1~5) 참고자료	I. 서론 (1~6) II. 본론 (1~7) 참고자료	I II III

이 문서는 한국산업기술대학교 컴퓨터공학부의  
 “종합설계”교과목에서 프로젝트“FaceVisitor”을 수행하는  
 (S2-11, 김동욱, 허주영)들이 작성한 것으로 사용하기 위해서는  
 팀원들의 허락이 필요합니다.

# 목 차

## I. 서론

1. 작품선정 배경 및 필요성 .....	4P
2. 기존 연구/기술동향 분석 .....	4P
3. 개발 목표 .....	5P
4. 팀 역할 분담 .....	5P
5. 개발 일정 .....	6P
6. 개발 환경 .....	6P

## II. 본론

1. 개발 내용 .....	7P
2. 문제 및 해결방안 .....	7P
3. 시험시나리오 .....	8P
4. 상세 설계 .....	9P
5. Prototype 구현 .....	23P
6. 시험/ 테스트 결과 .....	28P
7. Coding & DEMO .....	30P

## III. 결론

1. 연구 결과 .....	33P
----------------	-----

참고자료 .....	34P
------------	-----

# 서론

## 1. 작품선정 배경 및 필요성

항목	내 용
작품선정 배경	<ul style="list-style-type: none"> <li>• <b>CCTV</b>: 현재 거의 모든 장소에 CCTV 설치는 필수적인만큼 CCTV인프라와 인식이 크게 좋아짐</li> <li>• <b>추천 시스템</b>: 넷플릭스, 유튜브처럼 추천시스템은 이제 E-커머스 분야에서 뿐만이 아니라 대부분의 비즈니스에 필요한 시스템</li> <li>• <b>인건비</b>: 최저임금의 급격한 인상으로 매장운영에 인건비 문제가 크게 대두됨</li> </ul>
필요성	<ul style="list-style-type: none"> <li>• 실시간으로 고객의 <b>얼굴을 인식</b>하고 <b>분석</b>하여 매장 운영에 도움을 줌</li> <li>• 현재 오프라인 매장에 상품 추천 시스템이 없음. <b>개인화된 상품 추천 시스템</b>을 개발함으로써 오프라인 매장의 제한적인 마케팅 및 수입 채널을 한 단계 추가</li> <li>• 얼굴인식을 통한 로그인, 직원이 필요없는 <b>셀프 결제 서비스</b>를 개발하여 고객에게는 <b>편리한 결제</b>, 매장에게는 <b>인건비 감소효과</b>를 제공</li> <li>• 관리자 웹을 통해 고객을 <b>분석</b>하고 이를 <b>활용</b>할 수 있음</li> <li>• 고객용 웹을 통해 빠른 <b>로그인, 셀프결제, 상품 추천</b> 서비스를 고객에게 제공할 수 있음</li> </ul>

## 2. 기존 연구/기술동향 분석

항목	내 용
기존 연구	<ul style="list-style-type: none"> <li>■ <b>NHN TOAST(토스트)</b> <ul style="list-style-type: none"> <li>• 토스트는 자사 카메라를 이용해 매장안의 고객 수와 테이블 점유율을 분석해줌</li> </ul> </li> <li>■ <b>AMAZON</b> <ul style="list-style-type: none"> <li>• 아마존은 고객이 상품을 살펴본 시간을 분석해 개인화 상품 추천 시스템을 제공</li> </ul> </li> <li>■ <b>LOPLAT</b> <ul style="list-style-type: none"> <li>• 로플랫은 실내 위치를 측정해 얻은 고객의 동선 및 매장 방문 이력 데이터를 마케팅에 활용한다.</li> <li>• 고객의 로프라인 행동 분석을 기반으로 정교한 타겟 마케팅 실현 및 위치기반 서비스 구현</li> <li>• Wi-Fi 신호 기반으로 매장또는 층 단위로 장소를 인식하여 정교한 위치기반 서비스 구현</li> </ul> </li> </ul>
기술동향 분석	<ul style="list-style-type: none"> <li>• 딥러닝을 활용한 추천 알고리즘이 대세를 이루고, 얼굴인식 또한 딥러닝의 발전으로 보다 높은 인식률을 얻을 수 있음</li> <li>• 롯데마트, 인터파크, 에이블리 등등 AWS PERSONALIZE 서비스를 이용해 추천 서비스 제공</li> <li>• 유튜브는 시청했던 영상/좋아요/구독(이벤트) 등의 데이터들을 기반으로 시청자에게 도움이 될 만한 동영상을 제공</li> </ul>

### 3. 개발목표

항목	내 용
서비스 개발	<ul style="list-style-type: none"> <li>• 실시간 고객 얼굴인식</li> <li>• 개인화된 상품 추천 시스템 개발</li> <li>• 셀프 결제 서비스 개발</li> </ul>
웹 개발	<ul style="list-style-type: none"> <li>• 매장용 관리자 웹 개발</li> <li>• 매장용 고객용 웹 개발</li> <li>• 최고 관리자 웹 개발</li> </ul>
서버 개발	<ul style="list-style-type: none"> <li>• 매장용 관리자 서버 개발</li> <li>• 매장용 고객용 서버 개발</li> <li>• 최고 관리자 서버 개발</li> </ul>
개발 방향	<ul style="list-style-type: none"> <li>• 빠른 제품 출시가 필요하고 AI 전담 팀을 꾸리기 힘든 현업 상황을 프로젝트 연구 상황으로 가정</li> <li>• 개발 시 많은 노력과 비용이 필요한 머신러닝 파트를 AWS의 SaaS 또는 오픈소스 라이브러리를 이용</li> <li>• 직접 처음부터 구현하는 것 보다 빠르고 성능 있게 구현하여 핵심 비즈니스 로직에 더 집중한다.</li> </ul>

### 목표

- 고객 얼굴을 기반으로 하는 개인화된 고객분석 및 추천, 셀프 결제 기능 서비스를 제공하는 플랫폼

### 4. 팀 역할 분담

	김동욱	허주영
설계 및 구현	<ul style="list-style-type: none"> <li>• 최고 관리자 웹 사이트</li> <li>• 셀프 결제</li> <li>• 웹 크롤링</li> </ul>	<ul style="list-style-type: none"> <li>• 얼굴인식</li> <li>• 상품 추천</li> <li>• 고객,매장,관리자 API 서버</li> <li>• 클라우드 인프라</li> <li>• 고객, 매장용 웹 사이트</li> </ul>
QA 및 유지보수	<ul style="list-style-type: none"> <li>• QA 전담</li> </ul>	<ul style="list-style-type: none"> <li>• 유지보수 전담</li> </ul>

## 5. 개발 일정

추진사항	12월	1월	2월	3월	4월	5월	6월	7월	8월	9월
요구사항 정의 및 분석										
시스템 설계										
구현										
통합 및 테스트										
유지보수										
문서화 및 졸업 작품 중간 보고 서 작성										
졸업작품 최종 보고서 작성										

## 6. 개발 환경

API Server	운영체제	Amazon Linux2
	SSD	8GB
	프레임워크	Spring boot 2.2, Hibemate 5.3
고객, 관리자 웹	언어	Javascript es6
	프레임워크	Vue.js 2
웹캠	모델명	로지텍 c270
	해상도	1200x960
도메인	<a href="https://www.facevisitor.co.kr">https://www.facevisitor.co.kr</a>	

## 본 론

### 1. 개발 내용

항목	내 용
서버와 데이터베이스	<ul style="list-style-type: none"> <li>Spring boot, jpa, tomcat을 이용한 서버 구축 및 RDS Mysql &amp; DynamoDB로 DB서버 구축</li> <li>고객 웹 서버: Python Flask</li> <li>배포: AWS EC2, LoadBalancer, ACM을 이용</li> </ul>
관리자 웹	<ul style="list-style-type: none"> <li>Vue.js를 이용해 SPA 방식으로 개발</li> <li>S3 web hosting을 이용해 배포</li> </ul>
실시간 얼굴 인식 및 분석	<ul style="list-style-type: none"> <li><b>인식:</b> Dlib library(알고리즘 :<b>Histograms of Oriented Gradients</b> , face landmark estimation)를 사용한 OpenFace 오픈소스 프로젝트가 훈련한 딥러닝 신경망 기술을 라이브러리형식으로 쉽게 사용하여 얼굴을 인식 (정확도: 99.38%)</li> <li><b>분석:</b> <b>AWS Rekognition</b>는 AWS 비전 과학자들이 구현한 딥러닝 알고리즘으로 방대한 양의 이미지를 미리 학습한 딥 러닝 기술을 API,SDK 인터페이스로 제공. 이 중 <b>Underlying detection</b> 알고리즘으로 얼굴의 각 특징을 추출 ex) 성별, 나이, 악세서리</li> </ul>
추천 시스템	<ul style="list-style-type: none"> <li>데이터 셋 준비: <b>beautifulsoup4</b> 웹 크롤링으로 올리브영,쿠팡 등에서 상품 데이터 준비</li> <li>이벤트 정의: 고객의 개인화된 행동을 기반으로 <b>이벤트</b> 정의 (특정상품 섹션에 머문 시간, 상품주문, 상품클릭, 장바구니 추가 등등)</li> <li>추천 시스템 : Surprise (scikit learn 추천시스템 특화 라이브러리)를 이용해 SVD알고리즘을 통한 추천시스템을 제공</li> <li>데이터셋을 준비하고 유저 이벤트만 설정한 딥러닝 추천 알고리즘으로 계산해주고 이 결과를 손쉽게 얻을 수 있음.</li> </ul>

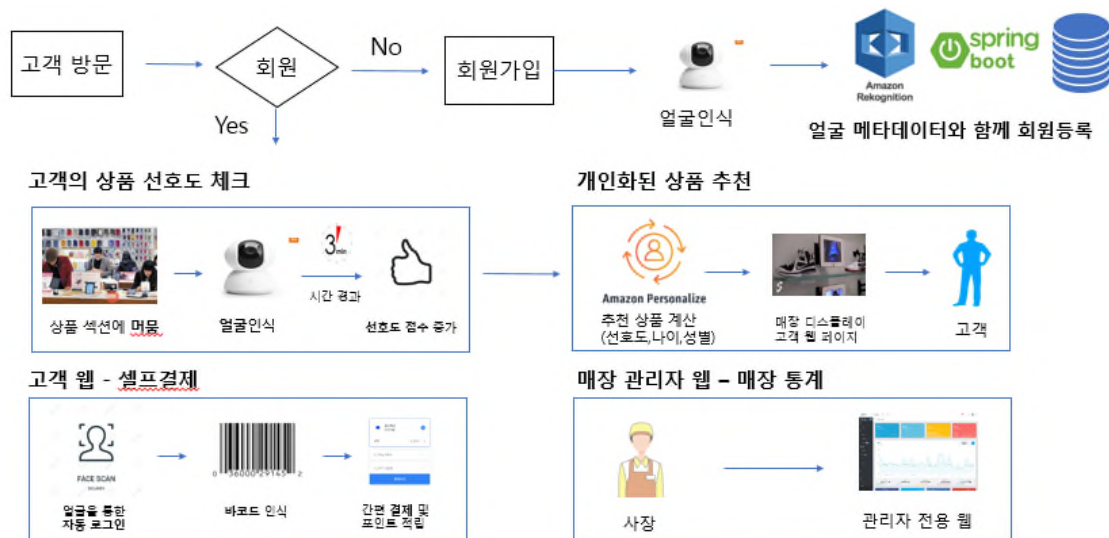
### 2. 문제 및 해결방안

항목	문제	해결방안
인건비	<ul style="list-style-type: none"> <li>최저임금의 급격한 인상으로 매장 운영에 인건비 문제가 크게 대두됨</li> </ul>	<ul style="list-style-type: none"> <li>셀프 결제 서비스를 이용해 점원에게 사용되는 인건비를 줄일 수 있음</li> </ul>
상품 추천	<ul style="list-style-type: none"> <li>오프라인 상품 추천 시스템이 없음</li> </ul>	<ul style="list-style-type: none"> <li>개인화된 상품 추천 시스템을 개발함으로써 오프라인 매장의 제한적인 마케팅 및 수입 채널을 한 단계 추가</li> </ul>

### 3. 시험 시나리오

항목	내 용
얼굴인식, 회원구분	• 웹캠을 통해 정상적으로 얼굴 데이터가 입력되고 회원 여부를 구분하는지 확인
상품 선호도 체크	• 미리 정의한 이벤트에 따라 선호도가 증가하는지 확인
개인화된 상품 추천	• 정상적으로 추천 상품을 계산하고 추천하는지 확인
셀프 결제	• 바코드를 인식하고 정상적으로 서버와 통신하는지 확인
매장 통계	• 저장된 데이터를 기반으로 정상적으로 매장 통계를 생성하는지 확인

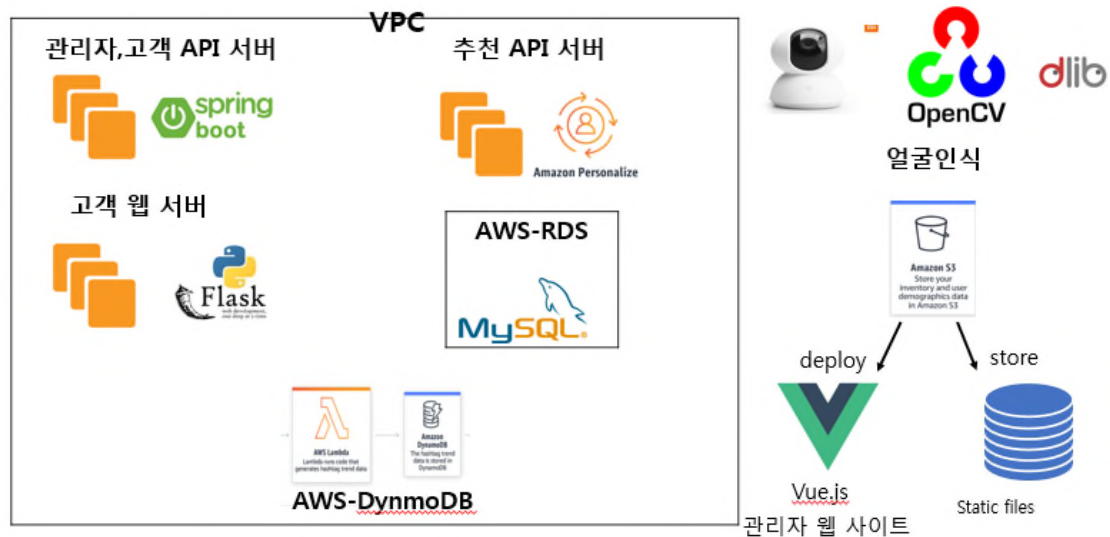
#### 시험 시나리오(그림)





## 4. 상세 설계

시스템 구성도 (사진 1)



- 상세 설계부터는 Git을 이용하여 Pull Request 생성 후 Merge하였음.

항목 (표 1)	내 용
Fork	<ul style="list-style-type: none"> <li>타겟 프로젝트의 저장소를 자신의 저장소로 Fork</li> </ul>
Clone 설정	<ul style="list-style-type: none"> <li>Fork로 생성한 저장소를 로컬에 Clone</li> </ul>
Remote 설정	<ul style="list-style-type: none"> <li>로컬에 원격 저장소 추가</li> </ul>
Branch 생성	<ul style="list-style-type: none"> <li>로컬 컴퓨터에서 코드를 추가하는 작업은 Branch를 만들어서 진행</li> </ul>
Add, Commit, Push	<ul style="list-style-type: none"> <li>코드 작성, 수정 작업 후 Add, Commit, Push</li> </ul>
Pull Request	<ul style="list-style-type: none"> <li>Pull request 생성</li> </ul>
Merge	<ul style="list-style-type: none"> <li>코드 리뷰 후 Merge 여부 결정</li> </ul>

항목(표 2)	내 용
데이터 셋 준비	<ul style="list-style-type: none"> <li>BeautifulSoup4 라이브러리를 이용하여 웹 크롤링, 상품 데이터 준비</li> </ul>
이벤트 정의	<ul style="list-style-type: none"> <li>고객의 특정한 행동을 이벤트로 정의하고 점수를 매김(장바구니 추가 : 7점 , 주문 10점 , 얼굴 인식 된 유저 10점)</li> </ul>
딥러닝 추천 계산	<ul style="list-style-type: none"> <li>SVD 알고리즘으로 각 고객에게 맞춤 추천 상품을 계산</li> </ul>
결과	<ul style="list-style-type: none"> <li>유저의 이벤트가 기록되어있는 CSV파일을 읽고 SVD알고리즘으로 훈련</li> <li>협업필터링을 통한 추천시스템을 통해 가장 높은 점수를 줄 것으로 예상되는 상품들을 정렬하여 상위 20개를 뽑아 추천한다.</li> </ul>

## ✓ 데이터 셋 준비

```

Elements Console Sources Network Performance Memory Application Security Audits
▶<div class="cate_align_box">...</div>
▼<ul class="cate_prd_list">
  ▼<li class="flag" data-index="0">
    ▼<div class="prd_info">
      ▼<a href="javascript:;" class="prd_thumb_goodslist" data-ref=goodsno="A000000129553" data-ref-dispcatno="1000001000100060009" data-ref-itemno="001">
        <span class="thumb_flag best">베스트</span>
        
      </a>
      ▼<div class="prd_name">
        ▼<a href="javascript:;" class="goodslist" data-ref=goodsno="A000000129553" data-ref-dispcatno="1000001000100060009" data-ref-itemno="001">
          <span class="tx_brand">비올레인</span>
          <p class="tx_name">비올레인 녹두 약산성 클렌징폼 80ml</p> == $0
        </a>
      </div>
    </div>
  </li>
</ul>

```

사진2(올리브영 홈페이지 HTML 구조)

- 올리브영 홈페이지(<https://www.oliveyoung.co.kr>)의 클렌징, 헤어 소품, 건강 카테고리의 상품들을 웹 크롤링하여 JSON파일로 변환 후 사용자 웹 사이트의 상품 목록 구축

### 웹 크롤링 소스코드 (사진 3)

```
import requests
import urllib.request
from bs4 import BeautifulSoup

def get_product_info(prd_info):
    name_div = prd_info.find("div", {"class": "prd_name"})

    brand_spans = name_div.find("span", {"class": "tx_brand"})

    name_ptags = name_div.find("p", {"class": "tx_name"})

    price_ptag = prd_info.find("p", {"class": "prd_price"})
    price_spans = price_ptag.find("span", {"class": "tx_cur"})
    price_spans2 = price_spans.find("span")

    brand = brand_spans.text    ① 브랜드 명
    name = name_ptags.text      ② 상품 명
    price = price_spans2.text    ③ 가격

    imgURL = prd_info.find("img")["src"]    ④ 이미지 URL

    return {'brand': brand, 'name': name, 'price': price, 'img': imgURL}

def get_page_products(url):
    result = requests.get(url)
    bs_obj = BeautifulSoup(result.content, "html.parser")
    ul = bs_obj.findAll("ul", {"class": "cate_prd_list"})

    for list_num in range(0, len(ul)):
        boxes = ul[list_num].findAll("div", {"class": "prd_info"})
        for prd_info in boxes:
            product_info_list.append(get_product_info(prd_info))
```

- 파이썬 언어, BeautifulSoup 라이브러리를 이용하여 웹 크롤링 소스코드 작성
- 카테고리는 다르지만 HTML 구조는 같으므로 URL만 바꾸어서 웹 크롤링 실행
- ① 브랜드 명, ② 상품 명, ③ 가격, ④ 이미지 URL을 사전형으로 반환

## 샘플 상품과 실행 결과 (사진 4)



- 샘플 상품은 클렌징 카테고리의 가장 첫 번째 항목, 실행 결과를 보면 첫 번째 항목의 브랜드 명, 상품명, 가격과 일치
- 아래의 실행 결과와 같이 한 행에 하나의 카테고리의 상품들을 출력
- 실행 결과를 복사해서 JSON 파일 생성
- JSON 파일로 고객 웹 상품목록 구성

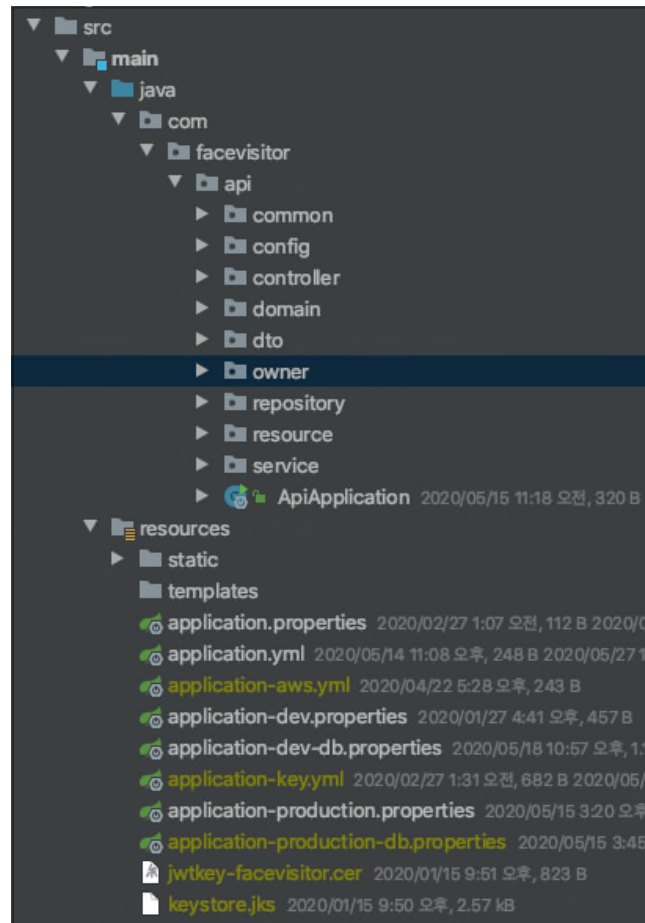
```
C:\git\python\crawler\venv\Scripts\python.exe C:\git\python\crawler\coupan\union.py
[{"brand": "비플레인", "name": "비플레인 녹두 약산성 클렌징폼 80ml", "price": "9,720", "img": "https://image.oliveyoung.co.kr/uploads/images/goods/400/10/0000/0012/400000012955102ko.jpg?l=ko"},
{"brand": "몰리몰리", "name": "몰리몰리 알미리 헤어젤", "price": "3,000", "img": "https://image.oliveyoung.co.kr/uploads/images/goods/400/10/0000/0011/400000011936901ko.jpg?l=ko"}, {"brand": "그린몬스터", "name": "[온라인만독] 탈리우드 48시간 기록세트 (탈리우드48시간+보틀)", "price": "24,500", "img": "https://image.oliveyoung.co.kr/uploads/images/goods/400/10/0000/0012/400000012955102ko.jpg?l=ko"}]
Process finished with exit code 0
```

### 카메라, 얼굴 인식 모듈 - 표 3

- 웹캠을 통해서 고객의 얼굴을 입력받고, 라이브러리를 이용하여 인식하고 분석함.
- 회원가입, 로그인, 정의한 이벤트에 사용됨.
- Opencv 프레임을 가져온 뒤 속도를 위해 크기 색상 조정
- Face recognition 라이브러리를 통해 얼굴 인식 및 기존 얼굴 이미지와 비교 수행
- 기존 가입된 얼굴일 시 설정된 상품 섹션 선호도 추가 및 프라임 라벨에 이메일 추가
- 프레임을 다시 원래 크기, 색상으로 조정
- 프레임을 jpg 포맷으로 변환하여 웹에서 스트리밍
- 실시간 얼굴 인식
- 기존 고객 이미지와 실시간으로 입력받은 얼굴 이미지 비교 및 유사값 계산
- 머문 시간 계산 등 선호도 계산

항목(표 3)	내 용
웹캠(로지텍 C270)	<ul style="list-style-type: none"> <li>• 웹캠을 사용하여 얼굴을 입력하고 회원가입, 로그인, 정의한 이벤트에 사용</li> </ul>
얼굴 인식	<ul style="list-style-type: none"> <li>• dlib의 얼굴 인식 기능인 FaceRecognition 패키지를 얼굴 인식, 저장된 얼굴과 실시간 얼굴과의 유사성 계산, 상품앞에 서 있는 시간 계산, 즉 정의한 이벤트(제품에 대한 선호도 등)에 사용</li> </ul>
얼굴 분석	<ul style="list-style-type: none"> <li>• AWS Rekognition을 이용하여 Underlying detection 알고리즘으로 얼굴의 각 특징을 추출하여 얼굴의 메타 데이터를 생성하고 이미 생성된 얼굴 데이터인지 체크</li> </ul>

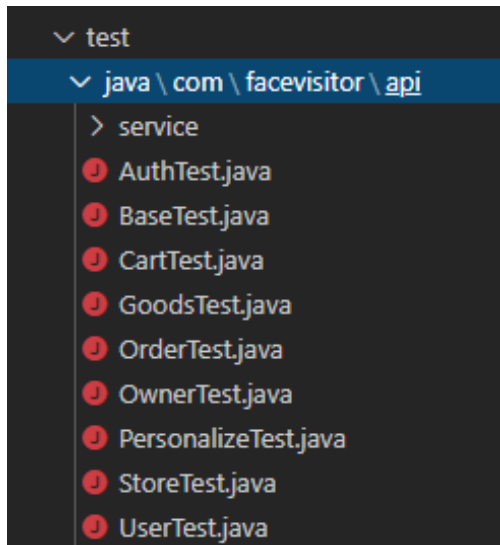
## 프로젝트 구조



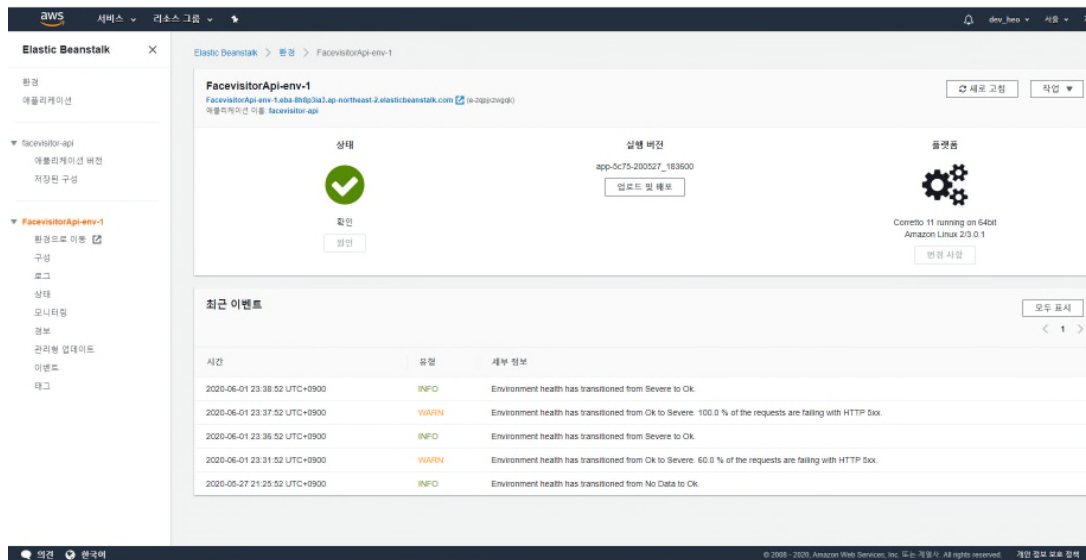
- common패키지에는 유틸이나 에러처리에 필요한 공통 유틸성 클래스들을 배치
- config 패키지에는 Cors, Security등의 설정에 필요한 클래스들을 배치
- Controller 패키지에는 REST Controller들을 배치
- domain 패키지에는 JPA Entity들을 배치
- dto 패키지에는 DTO 클래스들을 배치
- owner패키지에는 사장용 controller, service, dto패키지들을 따로 빼내어 배치
- repository패키지에는 각 entity에 해당하는 JPA Repository를 상속한 Repository들을 배치
- resource 패키지에는 HATEOS를 구현하기 위한 Resource를 배치
- service 패키지에는 각 entity에 해당하는 서비스들을 배치
- resources 패키지에는 스프링 설정에 해당하는 properties와 yml, jwt token 암호화에 쓰이는 인증서를 배치

항목(표 4)	내 용
도메인	• 유저, 얼굴, 상품, 주문, 장바구니, 포인트, 매장, 사장 등으로 나누어 DB 설계 및 서비스를 구현
테스트 - 사진 6	• 각 도메인 엔티티마다 해당하는 Controller 혹은 서비스를 테스트하는 테스트를 만들
DB	• DB는 Mysql 5.7 버전을 이용했고 JPA를 이용해 모델링 하였음. JPA의 OnetoOne, OneToMany, ManyToMany등의 다양한 연관관계 설정을 통해 관계 모델링을 수행했음

## 테스트 (사진 6)



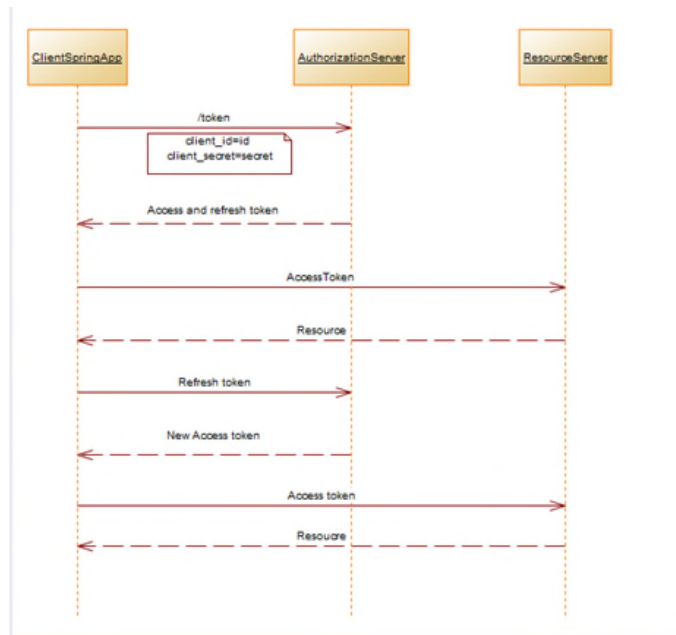
## rest api server 배포 & rds (사진 7)



- AWS ElasticBeanstalk cli를 이용해 EC2에 배포
- 구매해둔 facevisitor.co.kr 도메인을 Route53과 연결
- develop, production 버전으로 나누어 production버전으로 EC2에서 실행
- production일 때 db url은 AWS의 rds의 url로 설정
- Route53과 ec2와 연결
- api.facevisitor.com

### 인증 모듈(사진 8) - 표 5

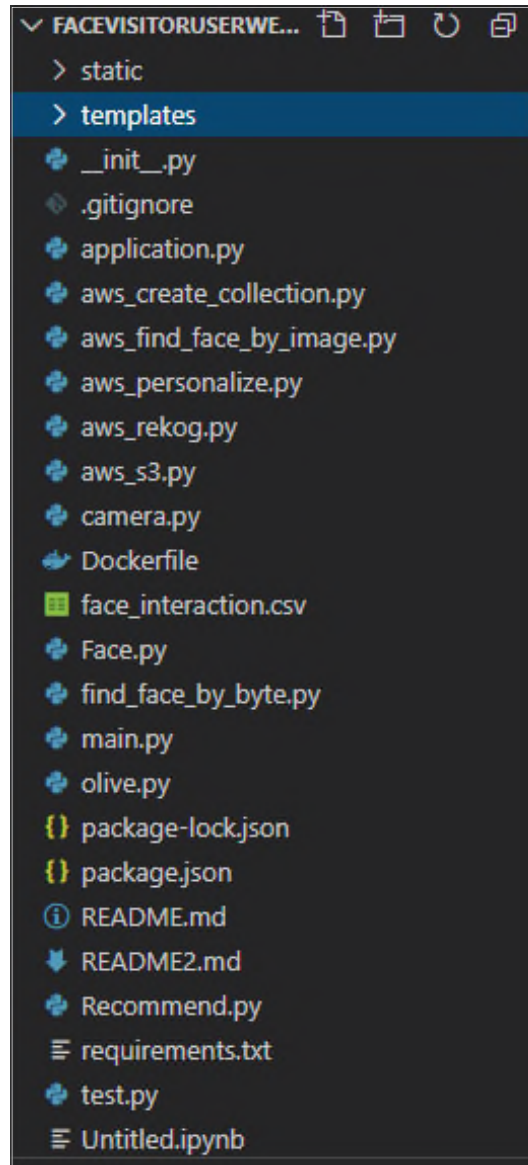
- OAUTH2, 인증 서버와 리소스 서버를 통합하여 API서버에 구축
- Resouce Owner Password Credentials Grant 승인 방식: 클라이언트가 암호를 사용하여 Access Token에 대한 사용자의 자격 증명을 교환
- Spring boot2 Oauth2 인증서버 설정 어댑터를 상속받아 구현
- 인증서비스: 고객 웹과 사장 웹의 회원가입, 로그인



항목(표 5)	내 용
로그인	<ul style="list-style-type: none"> <li>• 로그인 시 요청된 아이디와 비밀번호 파라미터를 이용하여 자격 인증</li> </ul>
토큰 요청	<ul style="list-style-type: none"> <li>• 아이디, 비밀번호를 기반으로 권한 서버에 Access Token 정보 요청</li> </ul>
토큰 발급	<ul style="list-style-type: none"> <li>• 권한 서버에서 Access Token 정보와 Refresh Token 정보 응답</li> </ul>
리소스 서버와 통신	<ul style="list-style-type: none"> <li>• Access Token을 기반으로 리소스 서버와 통신, 서버측에서 API 요청 시 토큰의 위변조, 유효기간 등 체크</li> </ul>

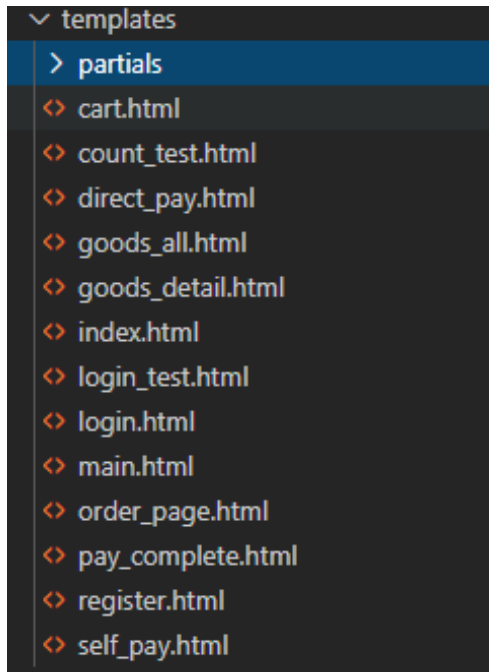


- REST API통신, View(html vue), 얼굴인식 기능을 제공하는 파이썬 Flask 프레임워크를 활용한 Controller + View 서버

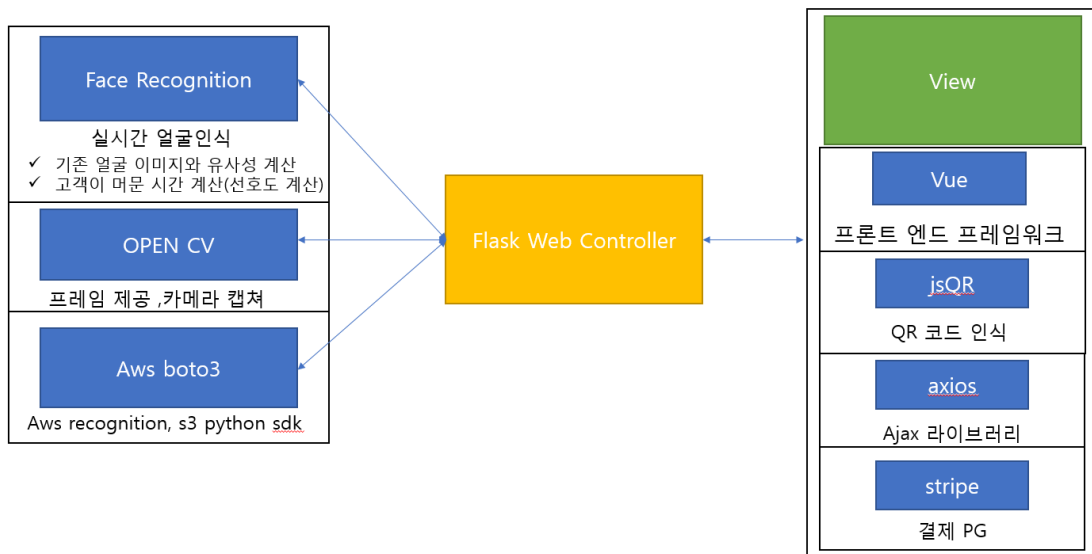


항목(표 5)	내 용
모듈화	<ul style="list-style-type: none"> <li>얼굴인식(Face.py), 서버(application.py), 추천시스템(Recommend.py)등으로 나누어 모듈화</li> </ul>
Vue - 사진 10	<ul style="list-style-type: none"> <li>templates 폴더 안에 html을 배치하고 해당하는 url에 따라 html을 제공했는데, 이 때 Vue.js를 cdn을 이용해 webpack 등의 번들화 필요없이 html만을 이용해 vue를 사용함</li> </ul>

Vue (사진 10)



유저 웹 서버의 상세 모듈 아키텍처 (사진 11)



## 추천 시스템

- **Surprise:** 파이썬 skit-learn의 추천 기능에 특화된 라이브러리
- **알고리즘:** Surprise는 SVD, KNN, BaseLine, CoCluster등의 다양한 알고리즘을 제공하는데 이 중 넷플릭스 추천 알고리즘으로 유명한 SVD 알고리즘을 선택
- **데이터셋:** SVD로 훈련하기 위해 csv파일의 데이터가 필요한데 user\_id, item\_id, rating 칼럼 형식으로 정함. 천개정도의 랜덤 데이터와 실제 데이터를 합쳐 데이터를 만들어 훈련에 용이하도록 함.
- **이벤트 정의:** 유저가 어떤 이벤트를 행하면 csv파일에 user\_id, item\_id, rating,timestamp 형식으로 행이 추가 됨.

```
user_id,item_id,rating,timestamp
5      ,2353    ,2      ,2133785114
5      ,2366    ,2      ,2133785114
7      ,2151    ,2      ,2133785114
7      ,2280    ,2      ,2133785114
7      ,2459    ,2      ,2133785114
68     ,2282    ,2      ,1765593085
51     ,2412    ,2      ,1633053639
37     ,2121    ,2      ,2098628494
46     ,2431    ,2      ,1924541546
77     ,2159    ,2      ,2395933752
```

- rating은 각 유저의 이벤트마다 다르게 부여됨.

```
if type == "view":
    value = 2
elif type == "like":
    value = 4
elif type == "cart":
    value = 7
elif type == "order":
    value = 10
elif type == "face":
    value = 10
```

## 추천 시스템

- **훈련:** dataset을 읽어들이고 SVD 알고리즘으로 훈련한다. 훈련된 결과를 dump 파일로 저장하여 사용한다.
- **추천 상품 제공:** user\_id에게 추천 상품을 기본적으로 10개 가져옴. n파라미터에 숫자를 입력하여 가져올 추천상품 개수를 설정할 수 있음

```
def get_recommend_by_user_id(self, user_id, n=10):
    print("User ID :", user_id)
    ests = [(iid, est) for uid, iid, true_r, est, _ in
self.predictions if uid == user_id]
    if ests is not None:
        ests.sort(key=lambda x: x[1], reverse=True)
        results = ests[:n]

        for (iid, est) in results:
            print("Item id : ", iid, "Estimate Value : ", est)

        item_id = [result[0] for result in results]
        return item_id
    else: return []
```

- **인기상품제공:** 인기상품은 유저 이벤트(face\_interaction.csv)의 행이 가장 많은 아이템을 가져와 보여준다.

```
def get_popularity_item_id(self, n=10):
    item_list = []
    popularDf = self.df.groupby('item_id').count()
    for i, row in popularDf.iterrows():
        item_list.append((row.name, row['user_id']))
    item_list.sort(key=lambda x: x[1], reverse=True)
    item_id = [result[0] for result in item_list[:n]]
    print("pop" + str(item_id))
    return item_id
```

### 얼굴인식을 통한 상품 선호도 체크

- 스케줄러: 3시간 마다 dataset을 재훈련하여 추천시스템을 업데이트. 매 초마다 얼굴인식을 통한 상품 선호도 체크 함수를 수행

```
# every 3 hour training recommend dataset
scheduler.add_job(func=recommend.train, trigger="interval", seconds=60 * 60 * 3)

# every 1 seconds tracking user face
scheduler.add_job(func=faceObject.facePreference, trigger="interval", seconds=1)
```

### • 얼굴 인식을 통한 상품 선호도 체크 - 사진 12

1. 매 1초마다 카메라에 얼굴이 인식되면 기존에 회원가입 된 유저의 얼굴과 비교하여 유사성을 구해 기존 회원여부를 계산
2. 회원일 시 pandas dataframe에 회원 이메일로 된 dataframe이 존재하는지 찾음
3. 만약 dataframe이 이미 존재할 시 count를 +1
4. 만약 dataframe데이터가 존재하지 않는다면 email, count =1 , timestamp = now의 형식으로 초기화
5. 데이터프레임에서 count가 180(3분 이상 고객이 해당 섹션에서 머물렀을 경우)이 넘는 유저를 찾아 id를 구하고 face\_interaction.csv에 user\_id,item\_id,rating=10을 기입하고 count를 0으로 초기화

### 얼굴인식을 통한 상품 선호도체크(사진 12)

```
def facePreference(self):
    self.face_names = []
    for face_encoding in self.face_encodings:
        distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        if len(distances) > 0:
            min_value = min(distances)
            if min_value < similarity:
                index = np.argmin(distances)
                name = self.known_face_names[index]

                if len(self.countDf.loc[self.countDf['user_name'] == name]["user_name"].values) > 0:
                    count = self.countDf.loc[self.countDf['user_name'] == name]["count"].values[0] + 1
```

```

        self.countDf.at[self.countDf['user_name'] ==
name, 'count'] = count
        self.countDf.at[self.countDf['user_name'] ==
name, 'timestamp'] = self.now()

        print("email : " + name)
        print("유사성 : " + str(min_value))
        print("현재 초 : " + str(time.localtime().tm_
sec))

        print("카운트 : " + str(count))
        # print(self.countDf)

    else:
        print("init new user")
        newCount = pd.DataFrame(
            data={"user_name": [name], "count": [1],
"timestamp": [datetime.datetime.now()]})
        self.countDf = self.countDf.append(newCount)
        self.countDf = self.countDf.reset_index(drop=
True)

        # customerFaceCount[name] = {'count': 0, 'lik
e': 0, 'timestamp': self.now()}

    self.cal_like()

```

## 5. Prototype 구현

홈페이지 (사진 1) - 표 1



이제 페이스 비지터로  
매장방문의 색다른 경험을 누리보세요



얼굴 로그인

얼굴인식만으로  
바로 로그인하실 수 있습니다.



셀프결제

상품을 인식하고 직접 빠르게 결제  
하세요!



개인화된 추천

실제 고객님의 관심있어하는 상품  
들을 추천드립니다.


순식간에 가입할 수 있어요!

지금 체험하기



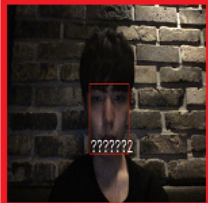
항목 (표 1)	내 용
회원가입	<ul style="list-style-type: none"> <li>얼굴 인식을 이용하여 회원가입을 통해 로그인, 셀프 결제, 상품 추천 서비스를 이용할 수 있음</li> </ul>
로그인	<ul style="list-style-type: none"> <li>얼굴 인식을 통해서 빠르고 간단하게 로그인 할 수 있음</li> </ul>
셀프 결제	<ul style="list-style-type: none"> <li>셀프 결제를 통해서 빠르고 간단하게 상품을 결제할 수 있음</li> </ul>
개인화된 추천	<ul style="list-style-type: none"> <li>개인화된 상품 추천 서비스를 통해서 자신에게 맞는 상품을 추천 받을 수 있음</li> </ul>

## 회원가입 (사진 2) - 표 2


셀프결재 장바구니 직원호출

### 고객 정보

로그인에 사용할 이메일과 비밀번호를 입력해주세요.

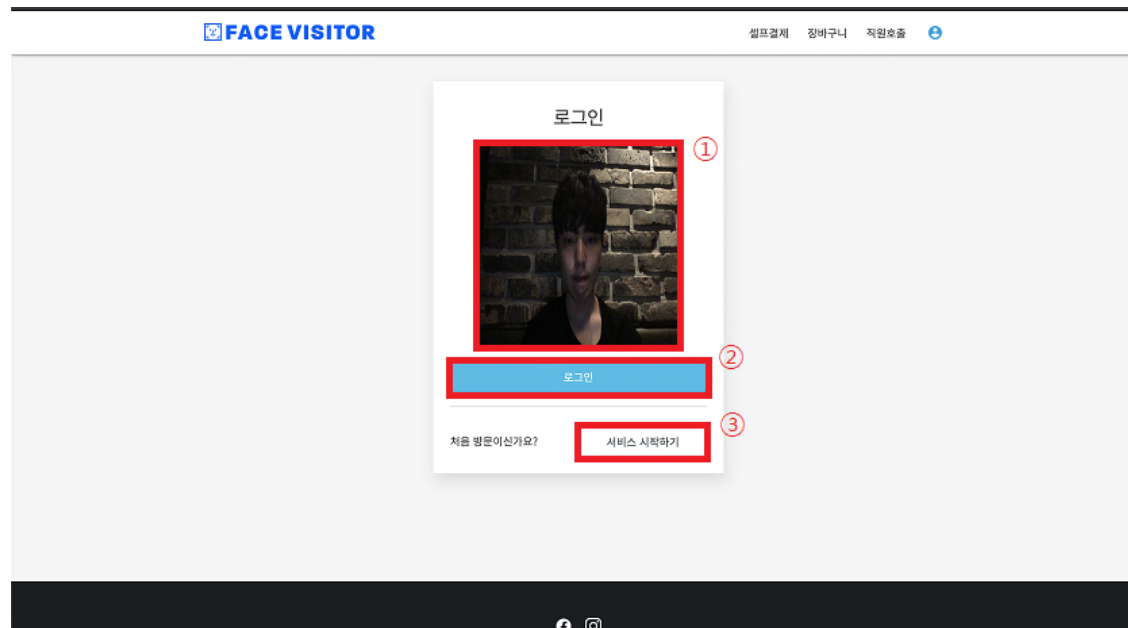

①

②

항목 (표 2)	내 용
얼굴인식	<ul style="list-style-type: none"> <li>얼굴을 인식하고 값 저장</li> </ul>
필수 입력값 확인	<ul style="list-style-type: none"> <li>이메일, 이름, 비밀번호, 비밀번호 확인, 전화번호가 모두 입력되었는지 확인</li> </ul>

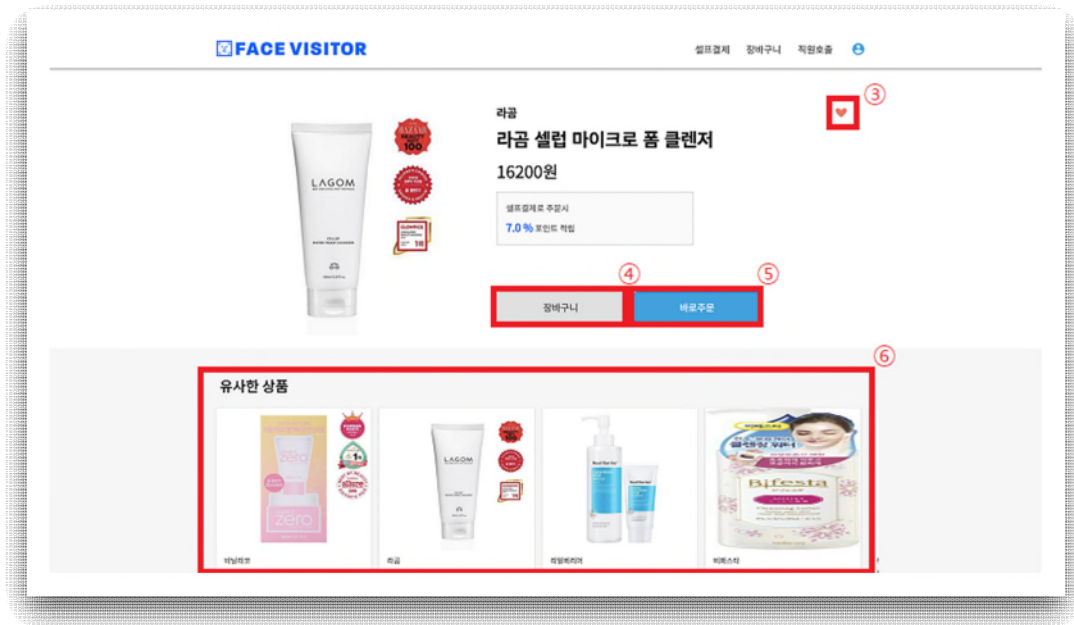


## 로그인 (사진 3) - 표 3



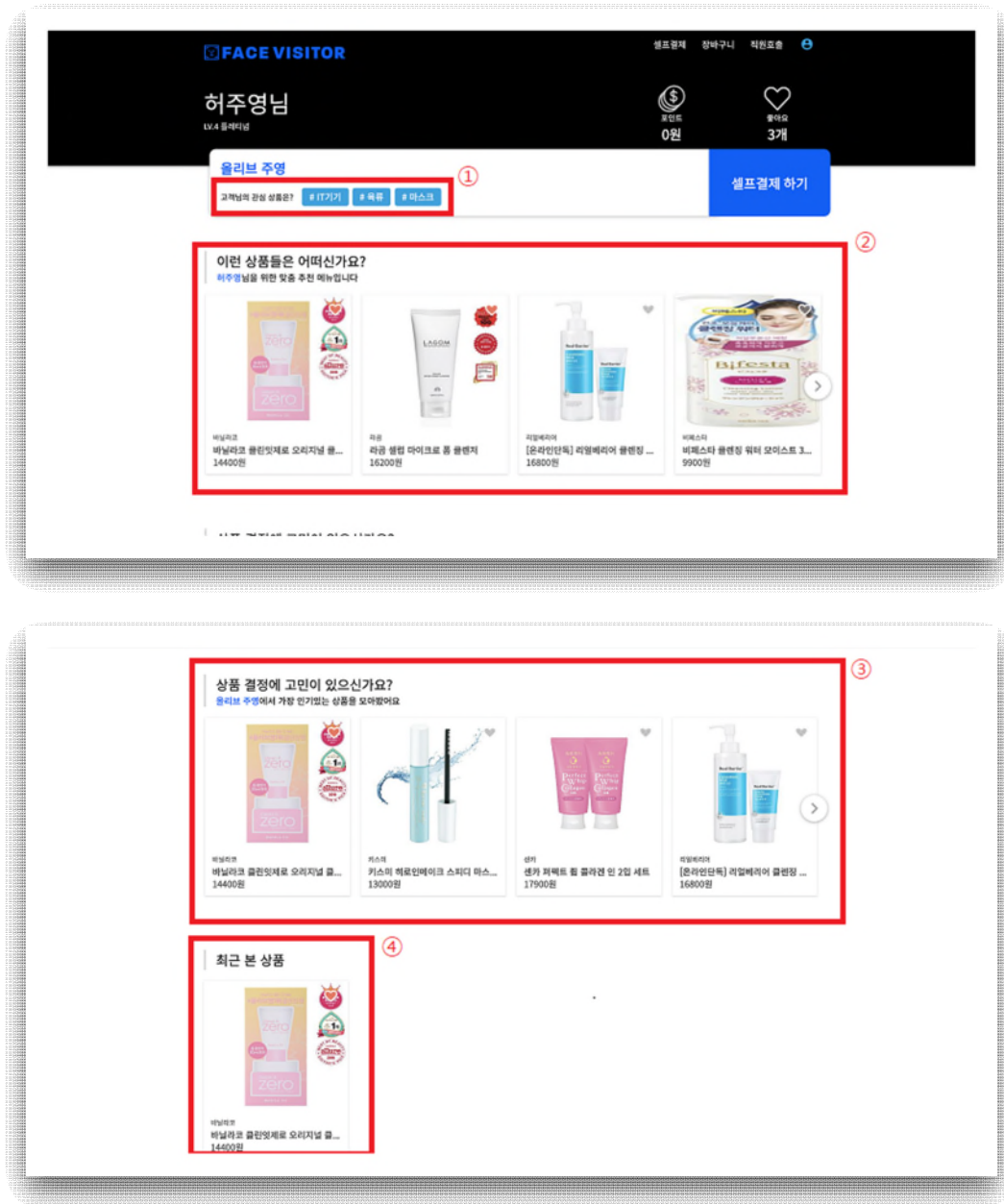
항목 (표 3)	내 용
얼굴인식	<ul style="list-style-type: none"> <li>얼굴을 인식하고 저장된 값과 비교/확인</li> </ul>
로그인	<ul style="list-style-type: none"> <li>얼굴을 인식하고 로그인</li> </ul>
회원가입	<ul style="list-style-type: none"> <li>회원이입을 하지 않았다면, 회원가입 진행</li> </ul>

상품 검색 (사진 4) - 표 4



항목 (표 4)	내 용
상품 검색	• 상품명이나 브랜드를 검색하여 상품 찾기
셀프 결제	• 상품을 장바구니에 담았다면, 상품 코드값을 이용하여 셀프 결제
좋아요	• 상품에 '좋아요'를 표시함으로써 나중에 다시 볼 수 있으며 선호도 값(상품 추천에 이용)을 증가시킴
장바구니	• 상품을 장바구니 담아서 한 번에 결제할 수 있음
바로 주문	• 선택한 상품 하나를 주문
유사한 상품	• 보고있는 상품과 유사한 상품을 추천

상품 추천 (사진 5) - 표 5



항목 (표 5)	내 용
관심 상품	• 관심있는 상품 카테고리를 나타냄
개인화된 상품 추천	• 개인의 선호도 값을 종합하여 상품들을 추천
인기 상품 추천	• 매장에서 가장 인기있는 상품들을 추천
최근 본 상품	• 고객이 최근에 본 상품을 나타냄

## 6. 시험/테스트 결과

상품 추천 시스템 알고리즘 (사진 1) - 표 1

유저2는 여자, 20대

유저 ID	상품 ID	성별	나이
2	2138	여	30
2	2586	여	30
3	2138	여	30
3	2586	여	30
3	2575	남	10
4	2586	여	30
4	2410	남	20
5	2138	여	30
5	2450	여	20

- 오프라인 매장에서의 상품 추천 시스템에서 같은 나이대와 성별, 구매 이력을 가진 고객에게 협업 필터링이 어떻게 적용되는지 테스트
- 사진 1에서 유저 2와 유저 3은 모두 상품 ID 2138, 2586을 선호하므로 성향이 가장 비슷한 유저임을 알 수 있음. 따라서 유저 3이 선호하는 2575를 유저 2에게 추천해줌(사진 2).

## 비슷한 성향의 유저에게 상품을 추천(사진 2)

```
유저의 성별 : Female
유저의 나이대 : 20
기본 추천 상품 점수

goods id : 2575 , score : 9
goods id : 2450 , score : 8
goods id : 2410 , score : 7
goods id : 2357 , score : 6
goods id : 2297 , score : 5
goods id : 2174 , score : 4
goods id : 2428 , score : 3
goods id : 2349 , score : 2
goods id : 2186 , score : 1
```

## 사진2의 내용을 기반으로 비슷한 나이와 성별을 고려한 상품을 추천(사진 3)

```
----최종 추천 상품 점수 ----
goods id : 2450 , score : 9.2
goods id : 2575 , score : 9
goods id : 2114 , score : 7.6
goods id : 2119 , score : 6.6
goods id : 2167 , score : 5.6
goods id : 2413 , score : 4.6
goods id : 2107 , score : 3.6
goods id : 2166 , score : 2.6
goods id : 2110 , score : 1.6
goods id : 2370 , score : 0.6
```

- 유저의 성별과 나이대가 상품의 선호 성별 나이대와 같을 시 각각 0.6의 점수를 더함.
- 두 번째 추천상품이었던 상품 2450은 성별과 나이대가 모두 같으므로 1.2점수가 더해져 가장 최우선 상품이 됨.

## 7. Coding & Demo

- 얼굴 인식을 이용한 상품 추천 시스템 Code: 보고서 21P

```
# every 3 hour training recommend dataset
scheduler.add_job(func=recommend.train, trigger="interval", seconds=60 * 60 * 3)
# every 1 seconds tracking user face
scheduler.add_job(func=faceObject.facePreference, trigger="interval", seconds=1)
```

스케줄러: 3시간 마다 dataset을 재훈련하여 추천시스템을 업데이트. 매 초마다 얼굴인식을 통한 상품 선호도 체크 함수를 수행

```
def facePreference(self):
    self.face_names = []
    for face_encoding in self.face_encodings:
        distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        if len(distances) > 0:
            min_value = min(distances)
            if min_value < similarity:
                index = np.argmin(distances)
                name = self.known_face_names[index]

                if len(self.countDf.loc[self.countDf['user_name'] == name]['user_name'].values) > 0:
                    count = self.countDf.loc[self.countDf['user_name'] == name]['count'].values[0] + 1
                    self.countDf.at[self.countDf['user_name'] == name, 'count'] = count
                    self.countDf.at[self.countDf['user_name'] == name, 'timestamp'] = self.now()

                    print("email : " + name)
                    print("유사성 : " + str(min_value))
                    print("현재 초 : " + str(time.localtime().tm_sec))

                    print("카운트 : " + str(count))
                    # print(self.countDf)

                else:
                    print("init new user")
                    newCount = pd.DataFrame(
                        data={"user_name": [name], "count": [1], "timestamp": [datetime.datetime.now()]})
                    self.countDf = self.countDf.append(newCount)
                    self.countDf = self.countDf.reset_index(drop=True)

                    # customerFaceCount[name] = {'count': 0, 'like': 0, 'timestamp': self.now()}

    self.cal_like()
```

- 추천 시스템에서의 추천 상품 제공 Code: 보고서 20P

```
def get_recommend_by_user_id(self, user_id, n=10):
    print("User ID :", user_id)
    ests = [(iid, est) for uid, iid, true_r, est, _ in
self.predictions if uid == user_id]
    if ests is not None:
        ests.sort(key=lambda x: x[1], reverse=True)
        results = ests[:n]

        for (iid, est) in results:
            print("Item id : ", iid, "Estimate Value : ", est)

        item_id = [result[0] for result in results]
        return item_id
    else: return []
```

user\_id에게 추천 상품을 기본적으로 10개 가져옴. n파라미터에 숫자를 입력하여 가져올 추천상품 개수를 설정할 수 있음,

- 추천 시스템에서의 인기 상품 제공 Code: 보고서 20P

```
def get_popularity_item_id(self, n=10):
    item_list = []
    popularDf = self.df.groupby('item_id').count()
    for i, row in popularDf.iterrows():
        item_list.append((row.name, row['user_id']))
    item_list.sort(key=lambda x: x[1], reverse=True)
    item_id = [result[0] for result in item_list[:n]]
    print("pop" + str(item_id))
    return item_id
```

인기상품은 유저 이벤트(face\_interaction.csv)의 행이 가장 많은 아이টে를 가져와 보여줌.



## 데모 배경



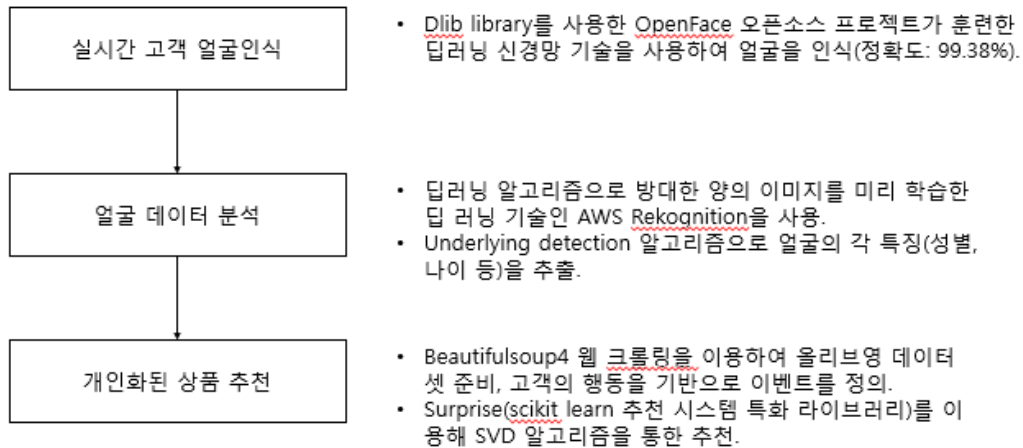
- 데모 배경: 사진 2에서의 교보문고의 도서검색대처럼 공동사용 PC환경을 가정.
- 매장 내에서 캠의 얼굴인식을 이용한 간편한 회원가입, 로그인, 셀프 결제, 포인트 적립, 개인화된 추천 서비스 제공
- 프로토타입(16P~27P)과 상품 추천 시스템을 실시간으로 적용하는 과정을 데모



# 결론

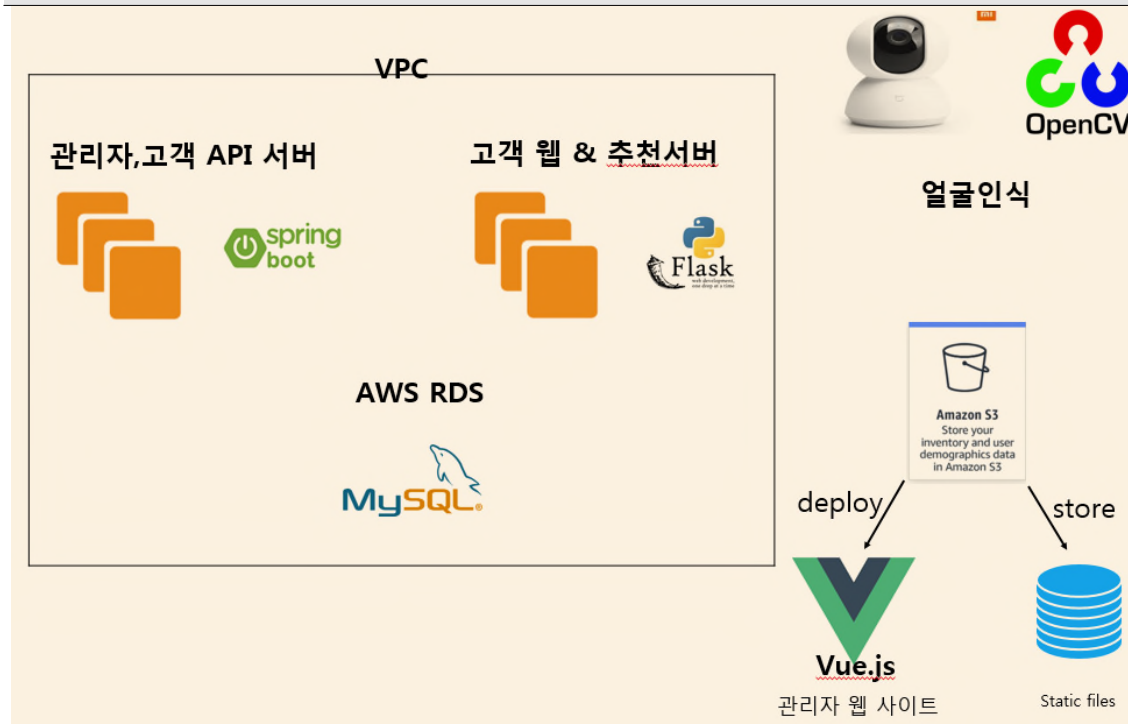
## 1. 연구결과

### 얼굴 인식을 이용한 고객관리시스템 흐름도



항목	최종 구현현황
얼굴 인식	<ul style="list-style-type: none"> <li>인식률: 99.38%</li> <li>회원가입, 로그인, 얼굴 분석(성별, 나이 구분)을 위한 얼굴 인식</li> <li>고객의 개인화된 행동을 기반으로 이벤트 정의(특정 세션에 머물기)</li> </ul>
상품 추천	<ul style="list-style-type: none"> <li>상품 수: 1000개</li> <li>고객의 행동을 기반으로 이벤트 정의(상품주문, 상품클릭, 장바구니 추가 등)</li> <li>Surprise(scikit learn 추천 시스템 특화 라이브러리)를 이용하여 SVD 알고리즘을 통한 추천</li> <li>협업필터링을 통한 추천 시스템을 이용하여 고객이 관심을 가질만한 상품을 예상하여 추천</li> </ul>
고객 웹	<ul style="list-style-type: none"> <li>고객이 회원가입, 로그인할 수 있도록 프레임을 제공</li> <li>상품 주문, 상품 추천, 셀프 결제와 장바구니 추가 등의 이벤트 활용 가능</li> </ul>
관리자 웹	<ul style="list-style-type: none"> <li>매장의 회원과 상품을 추가, 수정, 삭제할 수 있음</li> <li>매장의 통계를 볼 수 있음</li> </ul>
셀프 결제	<ul style="list-style-type: none"> <li>상품의 바코드를 통해 점원 없이 고객이 직접 결제할 수 있도록 함</li> </ul>

## 최종 SW 구성도



- 원래는 추천 API 서버가 있고 실시간으로 고객에게 개별화된 추천을 손쉽게 생성할 수 있는 AWS personalize 서비스를 사용하려고 했지만, 유지 비용이 많이 나가서 사용하지 않고 자체적으로 구현했음.
- AWS DynamoDB는 굳이 nosql을 사용할 필요가 없어서 RDS만 사용했음.

## 향후 연구과제

- 본 프로젝트에서는 얼굴인식과 협업 필터링을 이용한 상품 추천 방법을 제시하였다. 협업 필터링에서는 유사한 성향을 가진 사람이 누군지를 파악하는 것이 중요하며 유사도가 상품 추천의 척도이다. 따라서 사람이나 상품의 유사도에 따라 추천을 하는데 본 프로젝트에서 다수의 회원 수를 확보하지 못해 사용자들 간의 상품 추천을 다양한 방법으로 시험해보지 못했다. 따라서 다수의 회원 수에 대한 추가적인 연구들이 요구된다.

## 참고자료

### 도서:

- 추천 엔진을 구축하기 위한 기본서
- 토비의 스프링 3.1
- 자바 ORM 표준 JPA 프로그래밍
- 뷰 철저입문
- 스프링 부트와 혼자 구현하는 AWS 웹 서비스
- 혼자 공부하는 파이썬
- 한입에 웹 크롤링