2024.05~06

# SSAFY MALL


권서현 윤여준 이소연

# 목차순서

## 팀원 소개

### 권서현

- 사용자 간 유사도 분석 알고리즘 (Jaccard Similarity)
- JDA
- RDBMS
- 프론트 퍼블리싱

### 윤여준

- 프로젝트 기획 전반
- REST API
- GCP Sentiment 기반 상품 평점 관리
- Natural Language Entity 기반 상품 간 유사도 분석 알고리즘(Cosine Similarity)
- JPA

### 이소연

- 스프링 시큐리티(로그인 보안)
- NAVER ad api 연관 키워드 추출 및 검색 엔진 확장
- 데이터 시각화(Entity WordCloud)

SSAFY MALL << 1

검색하기

주제

- 기획
- 진행 계획
- 주요 기능과 알고리즘
- 기대 가능성

# 기획 배경

기존보다
더 편리한 쇼핑몰을
만들 순 없을까?

다른 사람들의 생각을
한 눈에
볼 수는 없을까?

# 시장 분석

## 고객 피드백 활용 부족

고객님의 리뷰를
단순히 텍스트로만 보여주고
이 데이터를 효과적으로
활용하지 못함

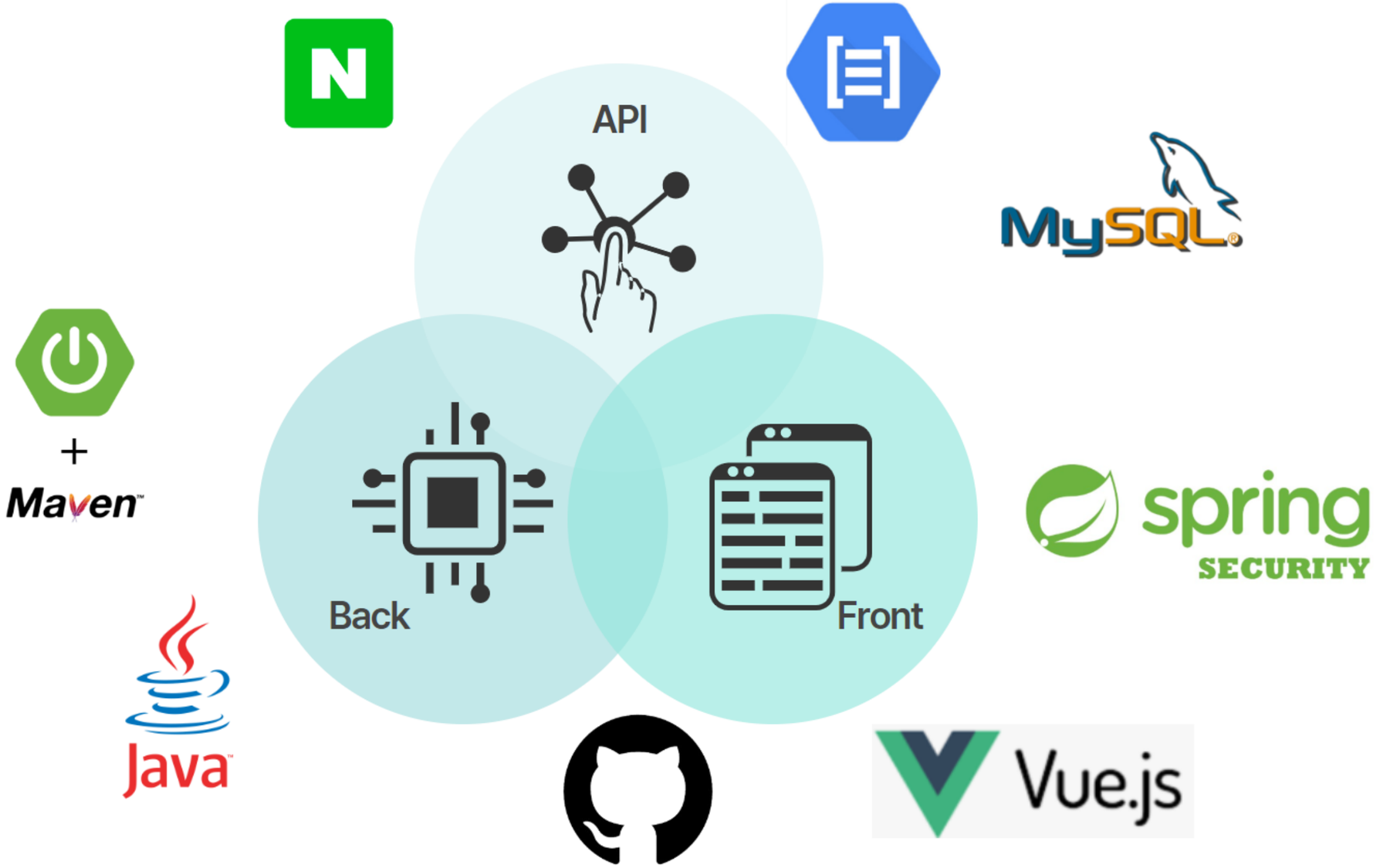
## 추천 상품의 한계

단순히
구매 이력이나
평점 기반으로
작동하여 제한적인 추천 제공

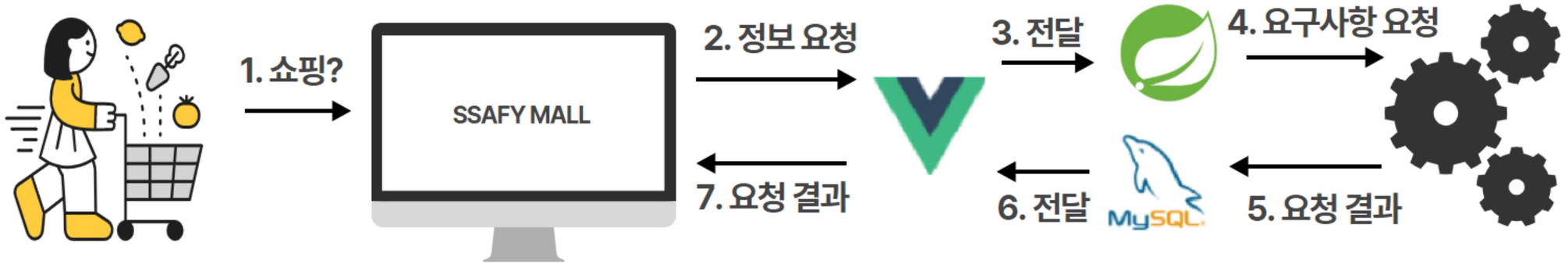
## 키워드 분석 부족

고객님의 리뷰에서
중요한 키워드를 추출, 분석
기능이 부족하여
고객님의 실제 요구 사항을
파악하는 데 어려움

# 개발 환경



# 개발 결과_ 시스템 구조도

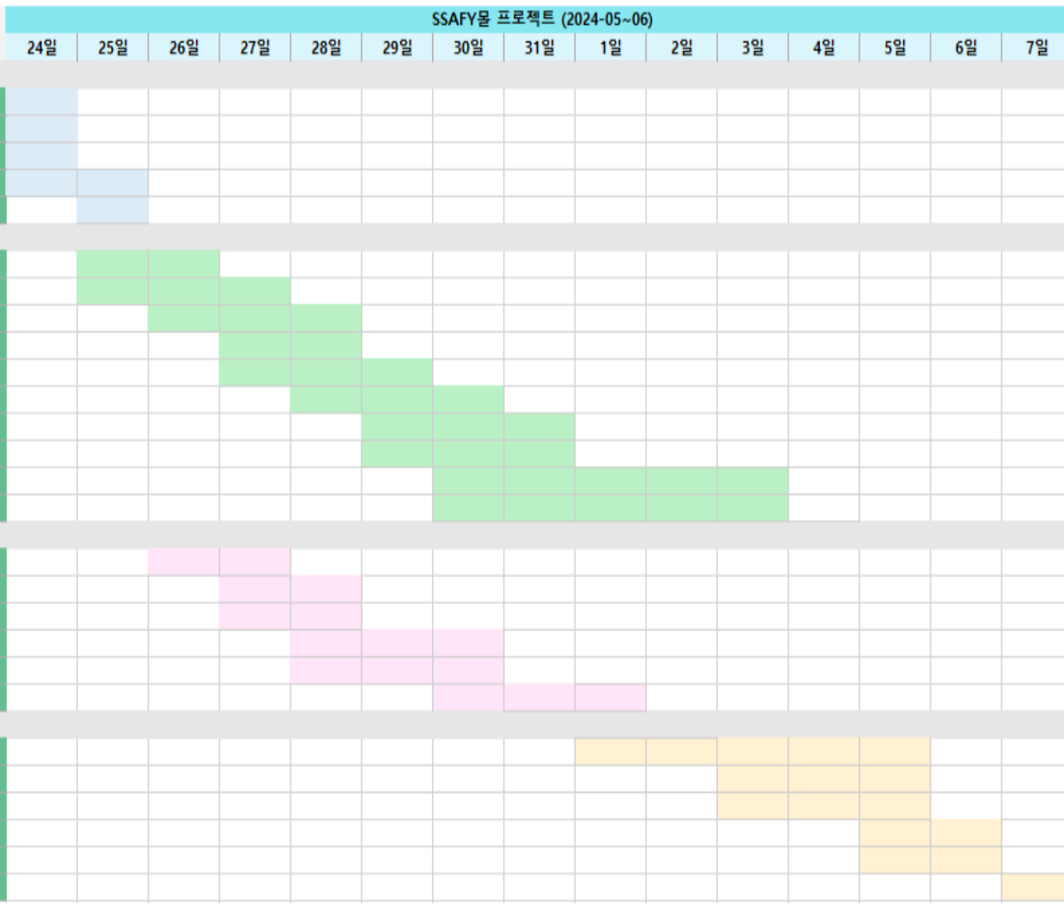


# 프로젝트 진행 계획_ Gantt Chart

## SSAFY Mall Gantt Chart

| | | | |
|---|---|---|---|
| 프로젝트 이름 | SSAFY MALL Project | 프로젝트명 | SSAFY MALL |
| 프로젝트 팀 구성원 | 권서현 윤여준 이소연 | 날짜 | 2024.05.24 ~ 2024.06.07 |

| 번호 | 작업 제목 | 시작일 | 종료일 | 기간 | 작업 완료 비율 |
|---|---|---|---|---|---|
| 1 | 프로젝트 기획 | | | | |
| 1.1 | 요구사항 정의서 작성 | 24년 5월 24일 | 24년 5월 24일 | 1 | 100% |
| 1.2 | 테이블 구조도(ERD) | 24년 5월 24일 | 24년 5월 24일 | 1 | 100% |
| 1.3 | UseCase 다이어그램 | 24년 5월 24일 | 24년 5월 24일 | 1 | 100% |
| 1.4 | 클래스 다이어그램 | 24년 5월 24일 | 24년 5월 25일 | 2 | 100% |
| 1.5 | 요구사항 명세서 작성 | 24년 5월 25일 | 24년 5월 25일 | 1 | 100% |
| 2 | BackEnd | | | | |
| 2.1 | 로그인 REST API | 24년 5월 25일 | 24년 5월 26일 | 2 | 100% |
| 2.1 | 로그인 보안 시큐리티 | 24년 5월 25일 | 24년 5월 27일 | 3 | 100% |
| 2.2 | 쇼핑몰 CRUD 기능 | 24년 5월 26일 | 24년 5월 28일 | 3 | 100% |
| 2.3 | 리뷰 기능 | 24년 5월 27일 | 24년 5월 28일 | 2 | 100% |
| 2.3 | 리뷰 Entity 수집 및 gnl api 분석 | 24년 5월 27일 | 24년 5월 29일 | 3 | 100% |
| 2.3 | 리뷰 센티멘트 분석 및 평점 산정 | 24년 5월 28일 | 24년 5월 30일 | 3 | 100% |
| 2.4 | 상품 찜 기능 | 24년 5월 29일 | 24년 5월 30일 | 3 | 100% |
| 2.5 | 검색 기능_NAVER ad api 확장 | 24년 5월 29일 | 24년 5월 30일 | 3 | 100% |
| 2.6 | 상품 추천 기능_자카드 유사도 분석 알고리즘 | 24년 5월 30일 | 24년 6월 03일 | 5 | 100% |
| 2.6 | 상품 추천 기능_코사인 상품 유사도 분석 알고리즘 | 24년 5월 30일 | 24년 6월 03일 | 5 | 100% |
| 3 | FrontEnd | | | | |
| 3.1 | 로그인 페이지 | 24년 5월 26일 | 24년 5월 27일 | 2 | 100% |
| 3.2 | 마이 페이지 | 24년 5월 27일 | 24년 5월 28일 | 2 | 100% |
| 3.3 | 메인 페이지 | 24년 5월 27일 | 24년 5월 28일 | 2 | 100% |
| 3.4 | 리뷰 페이지 | 24년 5월 28일 | 24년 5월 30일 | 3 | 100% |
| 3.4 | 리뷰 wordCloud 시각화 | 24년 5월 28일 | 24년 5월 30일 | 3 | 100% |
| 3.5 | 장바구니 기능 구현 | 24년 5월 30일 | 24년 6월 01일 | 3 | 100% |
| 4 | 결과 준비 | | | | |
| 4.1 | 코드 리뷰 | 24년 6월 2일 | 24년 6월 5일 | 4 | 100% |
| 4.2 | 기타 rest api 점검 | 24년 6월 3일 | 24년 6월 5일 | 3 | 100% |
| 4.3 | 단위테스트 | 24년 6월 3일 | 24년 6월 5일 | 3 | 100% |
| 4.4 | 통합테스트 | 24년 6월 5일 | 24년 6월 6일 | 2 | 100% |
| 4.5 | 시스템테스트 | 24년 6월 5일 | 24년 6월 6일 | 2 | 100% |
| 4.6 | 최종 점검 | 24년 6월 6일 | 24년 6월 6일 | 1 | 100% |



# 프로젝트 진행 계획_ 요구사항명세서

| 번호 | 대상 | 분류 | 기능 | 설명 | 상세 기능 |
|---|---|---|---|---|---|
| S01 | user | 로그인/회원관리 | 회원가입 | id, name, email, password,address 입력 시 회원 등록 | 중복 ID, email 체크<br>비밀번호 & 비밀번호 확인 비교하여 일치 여부 확인<br>role은 admin?....<br>가입 완료 클릭시 로그인 화면으로 이동 |
| S01 | user | 로그인/회원관리 | 로그인 | id, password 입력 | 아이디가 없거나 아이디&비밀번호 불일치시 오류메시지 출력 |
| S01 | user | 로그인/회원관리 | 로그아웃 | | 로그아웃 시 메인 페이지로 이동<br>세션에서 로그인 정보 삭제 |
| S01 | user | 로그인/회원관리 | 로그인 admin | 관리자 로그인 | 관리자 로그인 시 ADMIN 페이지로 이동 |
| | | | | | |
| S02-1 | user | 마이페이지 | 회원 정보 확인 / 수정 | password 입력 후 정보 수정 가능 | 주소, 이름 변경 가능? |
| S02 | user | 마이페이지 | 비밀번호 수정 | 현재 password 입력 후 수정 가능 | password 변경 후, 로그인 페이지로 이동<br>세션에서 로그인 정보 삭제 후 로그인 페이지로 이동 |
| S02 | user | 마이페이지 | 회원 정보 삭제 | password 입력 후 정보 수정 페이지에서 가능 | 삭제하시겠습니까? 메시지 출력<br>DB에서 삭제 |
| S02 | user | 마이페이지 | 주문 내역 & 리뷰 확인 | 사용자의 주문 내역, 리뷰 확인 마이페이지에서 가능 | 나의 쇼핑 내역, 나의 리뷰 내역 확인 |
| S02-2 | cart | 장바구니 | 장바구니 담기 | | 장바구니 기능은 로그인 시 가능 |
| S02 | cart | 장바구니 | 장바구니 조회 | | 장바구니 기능은 로그인 시 가능 |
| S03 | cart | 장바구니 | 장바구니 삭제 | | 장바구니 기능은 로그인 시 가능 |
| S04 | cart | 장바구니 | 장바구니 상품 주문 | | |
| | | | | | |
| S03 | | 메인페이지 | 전체 상품 조회 | 전체 상품 | 전체 상품 내역<br>사진과 금액을 함께 리스트에 나타내고, 한 페이지에 6개?의 데이터 출력<br>상품 클릭하면 상품 상세 페이지로 이동 |
| S03 | | 메인페이지 | 추천 상품 조회 | 전체 상품 하단에 추천 상품 리스트 조회 | 자연어 알고리즘을 통해 사용자의 관심도에 맞는 상품 추천 |
| S03 | | 상세페이지 | 상세 상품 조회 | 해당 상품 조회 | 해당 상품 내역 |
| S03 | | 상세페이지 | 추천 상품 조회 | 상품에 해당하는 추천 상품 리스트 조회 | |
| S03 | | 상세페이지 | 상품 검색 | NAVER ad api 사용 | NAVER ad api 확장 |
| S03 | | | | | |
| | | | | | |
| S04 | | 제품 리뷰 페이지 | 제품 리뷰 조회 | 제품 클릭 시 하단에 리뷰 확인 가능 | 제품 총 별점 조회<br>각 리뷰와 별점 조회<br>리뷰 작성자 클릭 시 구매 내역 or 선호 키워드? 확인 |
| S04 | | 제품 리뷰 페이지 | 제품 리뷰 작성 | 구매 내역 중 리뷰 작성 | 마이페이지 구매 내역에서 작성 버튼 클릭 시 작성 가능 (회원만 가능)<br>별점, 내용 작성<br>이름, 선호 키워드 불러오기 |

# 주요 기능 소개

유사도 분석 알고리즘을 통한 상품 추천
- Cosine 유사도 분석 - 관련 상품 추천
- Jaccard 유사도 분석 - 관심사 분석 상품 추천

NLP(자연어처리) 활용 상품 분석
- 상품의 Entity 추출 후 벡터화하여 Cosine 유사도 분석에 활용
- 리뷰의 Sentiment를 분석하여 평점화
- Sentiment 기반 키워드 추출 후 WordCloud 시각화

API 확장 검색 기능
- 네이버 광고 API 활용 – 연관 키워드 추출
- 검색 기능 사용 시 연관 키워드 동시 검색

# 핵심 알고리즘

## Cosine 유사도 분석 – 관련 상품 추천 알고리즘

- NLP를 사용하여 리뷰 등록 시, 자연어 처리 과정을 거친 후 Entity를 수집
- 리뷰의 Sentiment를 분석하여수집한 Entity를 Positive와 Negative로 구분, 저장
- Positive Entity를 가중치에 따라 벡터화
- 다른 상품의 Positive Entity와 Cosine 유사도 비교
- 유사도가 높은 상품을 리스트화하여 추천

```
//코사인 유사도 값으로 정렬할 리스트
List<double[]> cosineList = new ArrayList<double[]>();
for (List<EntitiesDto> entityList : list) {
        //엔티티 리스트
        String[] entities = entityList(entityList, Item);
        Map<String, Integer> index = new HashMap<>();

        for (int i=0; i<entities.length;
                index.put(entities[i], i
        }
        //A와 B벡터화
        double [] ItemVector = vectorizat
        double [] otherVector = vectoriz

        //A와 B 코사인 유사도 계산
        double cosineScore = calculateCos
        double [] temp = {itemIndex++, c
        cosineList.add(temp);
}
//코사인 유사도 순으로 정렬(내림차)
Collections.sort(cosineList, (o1,o2)->Do

for(int i = 0; i < cosineList.size() && i
        String name = list.get((int)cosi
        result.add(itemRepositiry.findBy
}

return result;
```

```
//엔티티 리스트화
public static String[] entityList(List<EntitiesDto> A, List<EntitiesDto> B) {
        Set<String> entitySet = new HashSet<>();
        // 상품 A 엔티티 추가
        for (EntitiesDto entity : A) {
                if(entity.getWeight()<0) continue;
            entitySet.add(entity.getEntity());
        }
        // 상품 B 엔티티 추가
        for (EntitiesDto entity : B) {
                if(entity.getWeight()<0) continue;
            entitySet.add(entity.getEntity());
        }

        // 엔티티 목록을 배열로 변환
        return entitySet.toArray(new String[0]);
}
//벡터화
public static double [] vectorization(Map<String, Integer> map, int size, List<EntitiesDto> item
        double [] ItemVector = new double[size];

        for(EntitiesDto entity : item) {
                //각 가중치를 벡터값에 대입
                if(map.containsKey(entity.getEntity())){
                        int idx = map.get(entity.getEntity());
                        ItemVector[idx] = entity.getWeight();
                }
        }
        return ItemVector;
}

// 두 벡터의 코사인 유사도 계산
   public static double calculateCosineSimilarity(double[] vector1, double[] vector2) {
        double dotProduct = 0.0;
        double norm1 = 0.0;
        double norm2 = 0.0;

        // 내적 계산
        for (int i = 0; i < vector1.length; i++) {
            dotProduct += vector1[i] * vector2[i];
```

# 핵심 알고리즘

## Jaccard유사도 분석 – 관심사 기반 상품 추천

- 사용자의 찜 목록을 벡터화
- 타 사용자들의 벡터 데이터와 Jaccard 유사도 분석
- 유사도가 높은 사용자 리스트를 추출
- 추출한 사용자들의 찜 목록 Counting Sort

⇒ 유사도가 높은 사용자 리스트의 찜 목록에 속한 상품을 카운팅하고 내림차순으로 정렬

- 사용자가 관심을 가질만한 상품을 리스트화하여 추천

```
private List<Item> CalJaccardSimilar(List<CartDetailDto> User, List<List<CartDetailDto>> list) {

        List<double[]> filteredList = new ArrayList<>();
        Set<String> user = new HashSet<>();
        for(int j = 0; j < User.size();j++) {
                user.add(User.get(j).getItemNm());
        }
        for (int i = 0; i < list.size(); i++) {
                List<CartDetailDto> tmp = list.get(i);
                Set<String> other = new HashSet<>();
                for(int j = 0; j < User.size();j++) {
                        other.add(tmp.get(j).getItemNm());
                }

                // 교집합 부분
                Set<String> intersection = new HashSet<>(user);
                intersection.retainAll(other);

                // 합집합 부분
                Set<CartDetailDto> union = new Hash
                union.addAll(tmp);

                double jaccardScore = (double) inte
                double [] temp = { i, jaccardScore
                filteredList.add(temp);
        }
        Collections.sort(filteredList, (o1, o2)->Do
        // 상위 10%의 데이터 수집
        int filteredSize = filteredList.size()/10;
```

```
// 인덱스 저장용
int indexVal = 0;
Map<String, Integer> map = new HashMap<>();

// 상위 10%유저의 찜 리스트 카운팅 정렬
List<String[]> countingList = new ArrayList<String[]>();
for(int i = 0; i< filteredSize; i++) {
        int idx = (int) filteredList.get(i)[0];
        List<CartDetailDto> cdd = list.get(idx);
        for(int j = 0; j<cdd.size();j++) {
                String name = cdd.get(j).getItemNm();
                if(user.contains(name))continue;
                if(map.containsKey(name)) {
                        countingList.get(map.get(name))[1] = ""+(Integer.parseInt(countingList.get(map.get(name))[1])+1)  ;
                }else {
                        map.put(name, indexVal++);
                        String []temp = { name, "1"};
                        countingList.add(temp);
                }
        }
}
Collections.sort(countingList,(o1,o2)->Integer.parseInt(o2[1])-Integer.parseInt(o1[1]));

// 결과( 10개이내 상품 추출 )
List<Item> result = new ArrayList<>();
for(int i = 0; i<10 && i<countingList.size(); i++) {
        Item item= itemRepository.findByItemNm(countingList.get(i)[0]);
        result.add(item);
}
return result;
```

# 핵심 알고리즘

## NLP 기반 WordCloud 시각화

- Google Cloud Platform의 Natural Language Processing API를 통해 리뷰의 Sentiment를 분석하고 Entity를 추출
- Sentiment를 바탕으로 Entity를 Positive와 Negative로 구분
- 리뷰의 Sentiment를 Average하여 수치화
- 수집한 Entity를 장단점으로 나누어 WordCloud 시각화

```
//긍정리뷰 엔티티 반환(워드 클라우드 생성 및 코사인유사도 활용)
public List<EntitiesDto> getPositiveEntities(String itemName){
        List<Review> reviewList = reviewRepository.findByItemNm(itemName);
        String text = "";
        for(Review review : reviewList) {
                double Score = 0;
                List<EntitiesDto> tmp = analyzeText(review.getContent(),  itemName);
                Score = tmp.get(0).getWeight();
                System.out.println("Score점수: "+ Score);
                if(Score < 0 ) continue;
                text+=review.getContent()+" ";
        }

        List<EntitiesDto>result = analyzeText(text,  itemName);
        result.remove(0);
        return result;
}
//부정리뷰 반환(워드 클라우드 생성용)
public List<EntitiesDto> getNegativeEntities(String itemName){
        List<Review> reviewList = reviewRepository.findByItemNm(itemName);
        String text = "";
        for(Review review : reviewList) {
                double Score = 0;
                List<EntitiesDto> tmp = analyzeText(review.getContent(),  itemName);
                Score = tmp.get(0).getWeight();
                System.out.println("Score점수: "+ Score);
                if(Score >= 0 ) continue;
                text+=review.getContent()+" ";
        }

        List<EntitiesDto>result = analyzeText(text,  itemName);
        result.remove(0);
        return result;
}
//평점 반환
public double getTotalScore(String itemName) {
        List<Review> reviewList = reviewRepository.findByItemNm(itemName);
        String text = "";
        double Score = 0;
        for(Review review : reviewList) {
                text+=review.getContent()+" ";
        }
```

# 프로젝트 진행 후기

## 권서현

추천 알고리즘을 적용하여 서비스를 구현할 수 있어 뿌듯한 프로젝트였습니다. 프로젝트 진행 중 다양한 아이디어를 떠올릴 수 있었습니다. SSAFY 2학기 프로젝트를 통해 이를 구체화하는 경험을 하고 싶습니다.

## 윤여준

지난 관통 프로젝트에서 시간이 부족해 구상만 하고 구현하지 못했던 알고리즘을 구현해볼 수 있는 기회가 되었습니다. 이번 프로젝트를 진행하면서 보완하고 싶은 부분이 많이 생겨 앞으로 더 공부하고 직접 구현해보고 싶습니다.

## 이소연

삼성 청년 sw 아카데미 상반기 마지막 프로젝트에서 배운 점과 아쉬웠던 점을 이번 프로젝트에서 구현하려고 노력해 공백기를 메울 수 있었습니다.
하반기에는 프론트, 백엔드 에서 더 다양한 경험을 쌓고 싶습니다.