



주류 추천 어플리케이션

주류 추천 어플리케이션  
**DRUNKEN**

**WHALE**

술 고 래



TAVE CONFERENCE

알만추

민지웅 박명진 안유진 홍서영 김동현



## CONTENTS

### 01 프로젝트 개요

- 주제 선정 배경
- 구현 기능 설명

### 02 추천시스템이란?

- 추천시스템 개요
- 추천시스템의 예시

### 03 구현한 추천 프로그램

- 데이터 설명
- 유사도 측정법
- 각자 구현한 코드

### 04 앱 구성 및 시연



주류 추천 어플리케이션 술고래

01 프로젝트 개요

## 주제 선정 배경

### 1) 왜 우리는 항상 같은 술만 마셔야 하는가!



**소맥** 한국의 국민 폭탄주!

소주와 맥주의 알콜자를 모아 '소맥'  
가장 쉽게 구할 수 있고, 가장 쉽게 먹을 수 있어  
한국인이 가장 즐겨 먹는 폭탄주! 국민 폭탄주!

**RECIPE**

1. 소주 한 잔과 맥주 2/3잔을 준비한다.
2. 맥주잔에 소주잔을 뿌려놓거나 부어마신다.

재료

소주 60ml + 맥주 140ml

18.4 Proof

**TIP!** 소주 농도가 너무 세다면, 잔을 두개 겹쳐서 계량하여 섞어도 좋다!

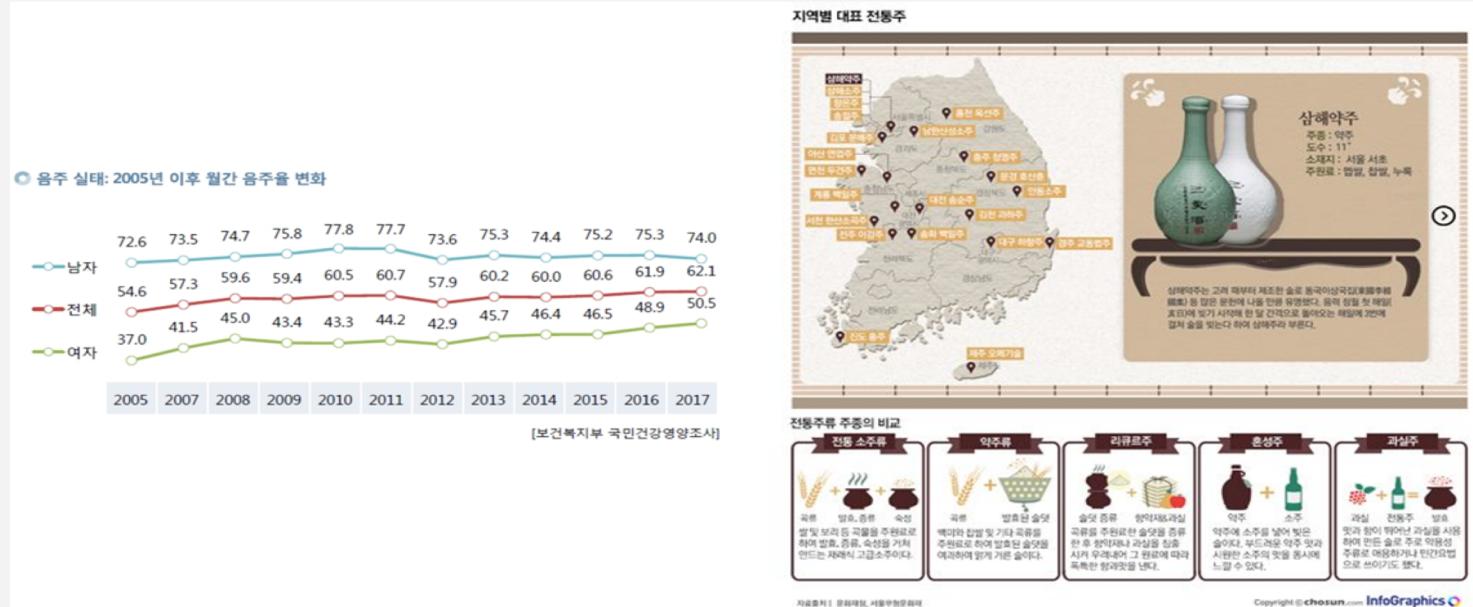


## 주류 추천 어플리케이션 술고래

## 01 프로젝트 개요

# 주제 선정 배경

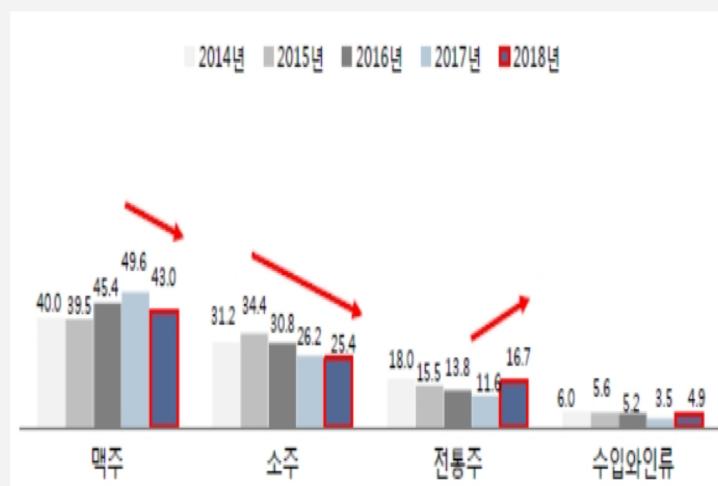
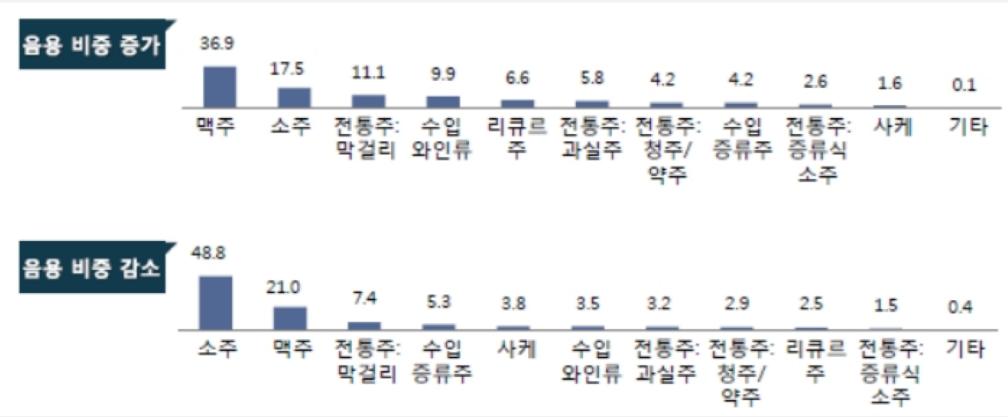
## 2) 변화하고 있는 음주 트렌드





## 주류 추천 어플리케이션 술고래

### 2) 변화하고 있는 음주 트렌드



### 01 프로젝트 개요

## 주제 선정 배경



## 주류 추천 어플리케이션 술고래

### 2) 변화하고 있는 음주 트렌드

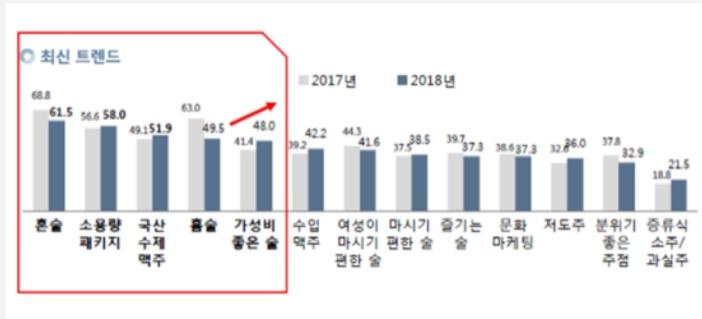
**아이뉴스24**

### "소맥 대신 쏘토닉이 대세"...'토닉워터' 판매량 급증

기사입력 2019.04.04. 오전 10:24 기사등록 스크랩 본문듣기 - 설정

▶ 공감 ▶ 댓글

▶ 글씨변경 ▶ 가 ▶ 목록 ▶



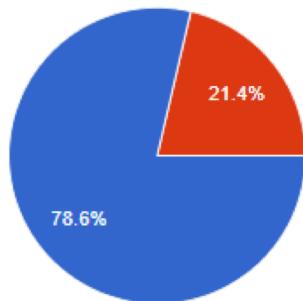


어플에 대한 필요성을 알아보기 위해 사전 설문 진행

총 103명 응답

1. 평소 술을 마실 때 새로운 술을 마시고 싶다는 생각이 든다.

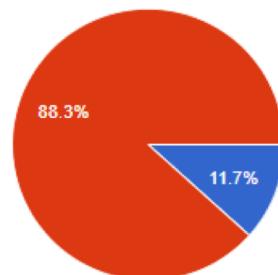
응답 103개



- 그렇다
- 아니다

2. 술에 관련된 정보를 얻는 곳이 있나요?

응답 103개



- 꾸준하게 정보를 얻는 곳이 있다.
- 따로 정보를 얻지는 않는다.



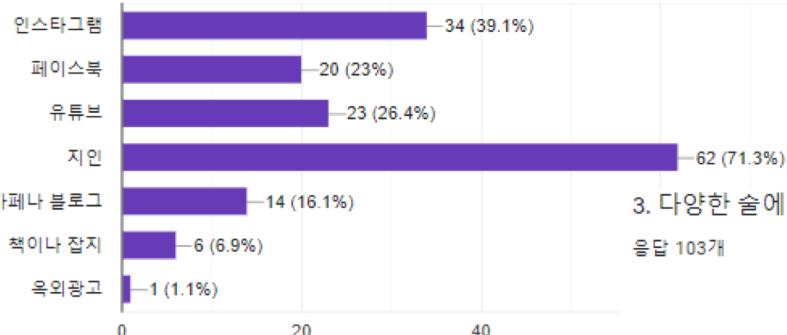
## 주류 추천 어플리케이션 술고래

### 01 프로젝트 개요

## 주제 선정 배경

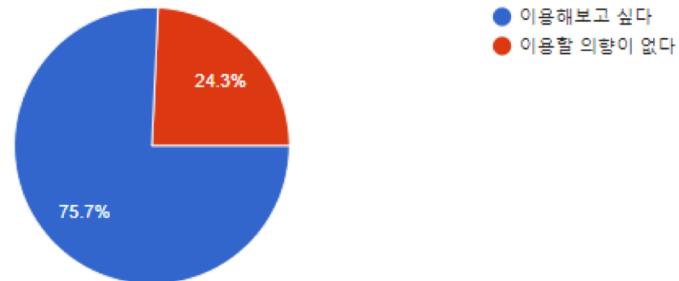
2-1. 술에 관련된 정보를 얻는 매체는?

응답 87개



3. 다양한 술에 관련된 정보를 제공하는 어플이 출시된다면 이용할 의향이 있나요?

응답 103개





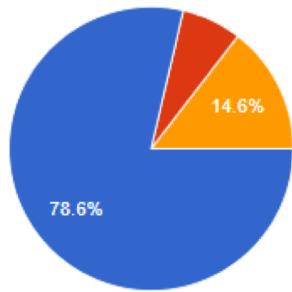
## 주류 추천 어플리케이션 술고래

## 01 프로젝트 개요

### 주제 선정 배경

4. 취향에 맞는 술을 추천해주는 어플이 출시된다면 이용할 의향이 있나요?

응답 103개

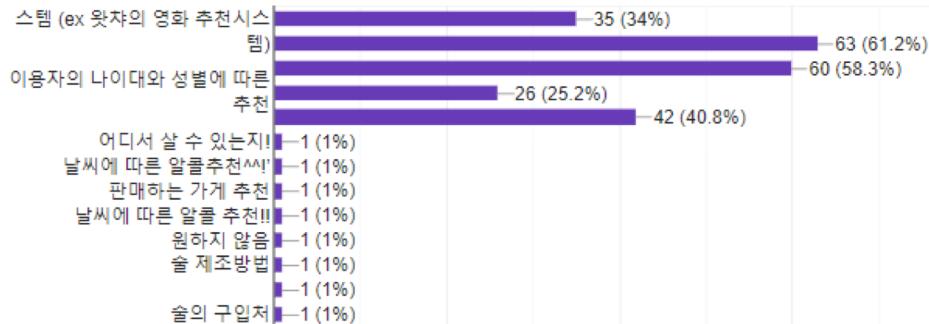


- 이용해보고 싶다
- 이용할 의향이 없다
- 잘 모르겠다

5. 술을 추천해주는 어플에 기대하는 기능이 있다면?

응답 103개

- 이용자의 평점에 따른 추천 시스템 (ex 왓챠의 영화 추천시스템)
- 본인이 좋아하는 술의 특성에 따른 추천
- 선호하는 주종과 도수, 가격대에 따른 추천
- 이용자의 나이대와 성별에 따른 추천
- 선택한 술에 따른 안주 추천





## 개발 언어

- JAVA, Python, R

## 개발 환경

- Android Studio, Jupyter Notebook, R studio

## 데이터베이스

- Firebase

## 라이브러리

- Python: NumPy, Pandas, SciPy, Scikit-Learn
- R: readr, proxy, dplyr



## 기본 컨셉

- '술'에 대한 정보 제공 및 추천

## 1. 술 검색 및 정보 기능

- 술 검색
- 원하는 술에 대한 상세 정보 제공
- 좋아하는 술 즐겨찾기

## 2. 술 추천 기능

- 사용자가 선택한 술과 유사한 술을 추천
  - 전체 주종 내에서 추천 → 와인이면 비슷한 풍미를 가진 막걸리, 맥주 추천
  - 선택한 술과 같은 주종 내에서만 추천 → 와인이면 비슷한 와인 추천

## 3. 향후 서비스 방향성

- 사용자 정보가 누적되면, 이를 분석해 개인의 특성에 맞는 술을 추천
- 평점 분석 기능 추가



크게 두가지 접근방법이 존재

### 1. *Collaborative Filtering*

- 사용자의 평가 내역을 분석하여 추천
- 좋은 성능, 사용자의 행동 패턴에 따라 적절한 추천
- 그러나 수집된 정보의 양이 많아야 좋은 결과가 나온다.

### 2. *Content-based*

- 아이템의 내용을 분석하여 추천
- 적은 정보만으로도 좋은 추천 가능
- 그러나 모델링 방식에 따라 정확도가 많이 달라지고, 비슷한 아이템 끼리만 추천이 가능하여 추천 범위가 제한된다.



## 1. *Collaborative Filtering*

- User-based

유저의 행위를 측정 및 분석하고 이를 기반으로 유저간 유사도 측정하여 유사한 유저끼리 해당 유저의 선호 아이템 기반으로 추천

- Item-based

아이템간의 유사도를 측정하며, 유저가 어떤 아이템을 선호하면 유사한 다른 아이템을 추천

## 2. *Content-based*

- 아이템이나 유저를 분석하여 비슷한 아이템을 추천



### **Content-based (콘텐츠 기반) 추천 시스템 사용 이유**

- 유저 데이터가 없음  
: User-based를 쓰기 위해서는 주류 구매자에 대한 정보 및 선호하는 주류에 대한 정보가 있어야 함
- 아이템 선호도에 대한 정보가 없음  
: Item-based를 쓰기 위해서는 그 아이템 선호도에 대한 정보가 있어야 함  
  
→ Collaborative Filtering 사용 불가능
- Content-based는 아이템과 유저간의 액션을 분석하는 것이 아니라 **컨텐츠 자체를 분석!**  
→ 사용 가능



## Netflix의 추천 콘텐츠 시스템 작동 방법

### 아이템 정보 Data

- Netflix 서비스와의 상호작용(시청 기록, 다른 콘텐츠 평가 결과 등),
- 유사한 취향을 가진 회원 및 Netflix 서비스에서의 선호 대상
- 장르, 카테고리, 배우, 출시연도 등 콘텐츠 관련 정보

### 사용자 정보 Data

- 하루 중 시청 시간대
- Netflix를 시청하는 디바이스
- 시청 시간



## 주류 추천 어플리케이션 술고래

### 03 구현한 추천 프로그램

#### 데이터 설명

<u>id</u>	<u>class</u>	<u>category</u>	<u>name</u>	<u>percent</u>	<u>origin</u>	<u>producer</u>	<u>wine_grape</u>	<u>whisky_category</u>	<u>sweet</u>	<u>light</u>	<u>soft</u>	<u>bitter</u>	<u>clean</u>	<u>smell</u>
10	wine	와인-레드	볼베르	15.5	스페인	보데가스 볼베르	템프라니오		1	1	1	0	0	0
85	soju	소주-	처음처럼	17.5		롯데칠성음료			0	1	0	1	0	1
237	위스키	위스키-American	잭다니엘 허니 700ml	35	미국	잭다니엘		Tennessee	0	0	0	0	0	0
597	beer	맥주-밀맥주	1664 블랑	5	폴란드	CARLSBERG SUPPLY COMPANY			1	0	0	0	1	0

총 656개의 술 데이터



id	class	category	name	percent	origin	producer	wine_grape	whisky_category	sweet	light	soft	bitter	clean	smell
			도수	원산지	제조사	포도 품종 (와인)						- 풍미 6가지 요소 -		

### class 대분류

- wine / makgeoli / beer / vodka / soju / whisky / korean (전통주)
- 총 7개의 대분류

### category 중분류

- 와인-레드 / 와인-화이트 / 위스키-American / 위스키-Irish / 맥주-IPA / 맥주-라거 등
- 총 24개의 중분류



## 1) Cosine Similarity

### 코사인 유사도

내적공간의 두 벡터간 각도의 코사인값을 이용하여 측정된 벡터간의 유사한 정도

#### 특징

1. -1은 서로 완전히 반대되는 경우를 의미
2. 0은 서로 독립적인 경우를 의미
3. 1은 서로 완전히 같은 경우를 의미

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

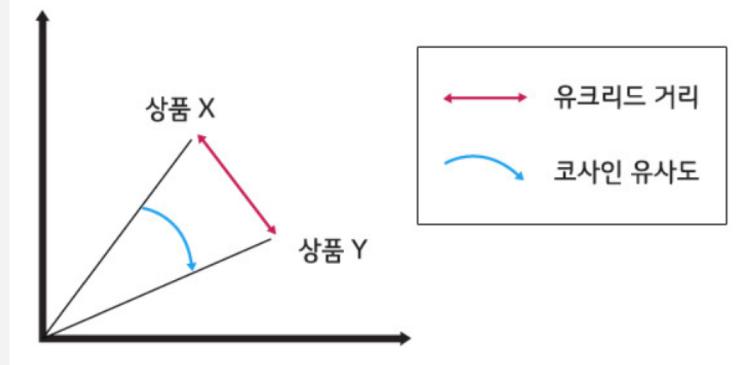


## 2) Euclidean Distance

### 유클리드 거리

n차원의 공간에서 두 점간의 거리를 알아내는 공식

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$





## 1) 서영 - Euclidean Distance + Cosine Similarity

도수 유사도는 euclidean distance로 구했고, 풍미 및 기타 정보의 유사도는 cosine similarity로 구했다.

술의 고유 id를 받아 비슷한 술의 고유 id를 return한다.

```
def general_recommendation():
```

도수 유사도 50% 풍미(6가지) 유사도 50% 고려해 주종 상관 없이 비슷한 술을 찾아줌

```
def wine_recommendation():
```

도수 유사도 50% 기타(category, origin, producer, wine\_grape, flavour) 50% 고려해 같은 대분류 내에서 비슷한 와인을 찾아줌



## 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # 도수 유사도 구하기
percent = data['percent']
dist_pair = []

# y축에 임의로 0을 부여한 거리 순서쌍 생성
for i in range(0, len(percent)):
    temp = []
    temp.append(data.loc[i]['percent'])
    temp.append(0)
    dist_pair.append(temp)

# get a distance matrix
df = pd.DataFrame(dist_pair, columns=['x', 'y'])
dist_matrix = distance_matrix(df.values, df.values)

# 정규화
min_max_scaler = MinMaxScaler()
regularised = min_max_scaler.fit_transform(dist_matrix)

# 1에서 빼줘서 더 가까운 것이 우선순위를 갖도록 변경하기
one_matrix = np.ones((654, 654))

final_dist = one_matrix - regularised

# 확인
print(final_dist)
```

```
[[1.          0.97560976 0.98795181 ... 0.82352941 0.82352941 0.82352941]
 [0.97619048 1.          0.98795181 ... 0.85154062 0.85154062 0.85154062]
 [0.98809524 0.98780488 1.          ... 0.83753501 0.83753501 0.83753501]
 ...
 [0.85      0.87073171 0.86024096 ... 1.          1.          1.          ]
 [0.85      0.87073171 0.86024096 ... 1.          1.          1.          ]
 [0.85      0.87073171 0.86024096 ... 1.          1.          1.          ]]
```

```
[ ] # flavour word list 만들기
flavour_list = [] # empty list

for i in range(0, len(data)):
    temp = ""
    if data.loc[i]['sweet'] == 1:
        temp = temp + "sweet "
    if data.loc[i]['light'] == 1:
        temp = temp + "light "
    if data.loc[i]['soft'] == 1:
        temp = temp + "soft "
    if data.loc[i]['bitter'] == 1:
        temp = temp + "bitter "
    if data.loc[i]['clean'] == 1:
        temp = temp + "clean "
    if data.loc[i]['smell'] == 1:
        temp = temp + "smell "
    flavour_list.append(temp)

# 해당 리스트 데이터 프레임에 추가
data["flavour"] = pd.DataFrame({"flavour":flavour_list})
flavour = data['flavour']

# flavour 코사인 유사도 구하기
# instantiating and generating the count matrix
count = CountVectorizer()
count_matrix = count.fit_transform(flavour)

# generating the cosine similarity matrix
cosine_sim = cosine_similarity(count_matrix, count_matrix)

# 확인
print(cosine_sim)
```

```
[[1.          0.5          0.35355339 ... 0.57735027 0.5          0.70710678]
 [0.5          1.          0.70710678 ... 0.57735027 1.          0.          ]
 [0.35355339 0.70710678 1.          ... 0.81649658 0.70710678 0.          ]
 ...
 [0.57735027 0.57735027 0.81649658 ... 1.          0.57735027 0.          ]
 [0.5          1.          0.70710678 ... 0.57735027 1.          0.          ]
 [0.70710678 0.          0.          ... 0.          0.          1.          ]]
```

텍스트의 코사인 유사도를 구하기 위해  
변수를 텍스트화



## 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # 도수, flavour를 모두 고려한 similarity 구하기 (weight는 각각 0.5)
new_sim = 0.5 * cosine_sim + 0.5 * final_dist

print(new_sim)

[{"user": [[1.0, 0.73780488, 0.6707526, ..., 0.70043984, 0.66176471, 0.7653181], [0.73809524, 1.0, 0.84752929, ..., 0.71444544, 0.92577031, 0.42577031], [0.67082431, 0.84745583, 1.0, ..., 0.8270158, 0.7723209, 0.41876751], ...], [0.71367513, 0.72404099, 0.83836877, ..., 1.0, 0.78867513, 0.5, ...], [0.675, 0.93536585, 0.78367387, ..., 0.78867513, 1.0, 0.5, ...], [0.77855339, 0.43536585, 0.43012048, ..., 0.5, 0.5, 1.0, ...]]]
```



## 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # 여러 주중 내 추천
# 주중에 상관 없이 도수와 풍미만 고려해 비슷한 술을 추천해줌
# 항목: 도수, 풍미 (약 6개 항목)

# 고유 id를 넣으면 해당 술과 비슷한 Top 10의 id를 return
def general_recommendation(input_id, new_sim = new_sim):

    # 고유 id로 index 찾기
    idx = data.index[data['id'] == input_id].tolist() # Int64Index 형식이라 list로 바꾸어줌

    # 해당 index의 유사도 리스트 sort in descending order
    score_series = pd.Series(new_sim[idx[0]]).sort_values(ascending = False)

    # 유사도 Top 10의 index 추출
    top_10_indexes = list(score_series.iloc[1:11].index)

    # 유사도 1인 항목이 하나 더 있어서 자기 자신이 포함되는 경우에는 자신을 뺀 Top 10의 index 재추출
    if top_10_indexes[0] == idx[0]:
        top_10_indexes = list(score_series.iloc[1:12].index)
        top_10_indexes.remove(idx[0])

    # 고유 id를 담기 위한 empty list 생성
    top_10_id = []

    # id list
    for i in top_10_indexes:
        id = data.loc[i]['id']
        top_10_id.append(id)

    return top_10_id

# test
print(general_recommendation(10))
```



[323, 90, 383, 92, 192, 557, 136, 349, 397, 360]



## 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # wine word list 만들기
wine_list = [] # empty list

# wine index list 추출
wine_idx = data.index[data['class'] == "wine"].tolist()

# wine: category, percent, origin, producer, wine_grape, flavour
for i in wine_idx:
    temp = ""
    temp = temp + data.loc[i]['category'] + " " + data.loc[i]['origin'] + " " + data.loc[i]['producer'].replace(" ", "") + " " + data.loc[i]['wine_grape'].replace(" ", "") + " "
    for flavour in data.loc[i]['flavour']:
        temp = temp + flavour
    wine_list.append(temp)

# wine 코사인 유사도 구하기
# instantiating and generating the count matrix
count = CountVectorizer()
wine_matrix = count.fit_transform(wine_list)

# generating the cosine similarity matrix
wine_sim = cosine_similarity(wine_matrix, wine_matrix)

# wine 도수 유사도 구하기
wine_pair = []
```

텍스트의 코사인 유사도를 구하기 위해  
변수를 텍스트화



## 1) 서영 - Euclidean Distance + Cosine Similarity

```
# y축을 임의로 0을 부여한 거리 순서쌍 생성
for i in wine_idx:
    temp = []
    temp.append(data.loc[i]['percent'])
    temp.append(0)
    wine_pair.append(temp)

# get a distance matrix
wine_df = pd.DataFrame(wine_pair, columns=['x', 'y'])
wine_matrix = distance_matrix(wine_df.values, wine_df.values)

# 정규화
min_max_scaler = MinMaxScaler()
wine_regularised = min_max_scaler.fit_transform(wine_matrix)

# 1에서 빼줘서 더 가까운 것이 우선순위를 갖도록 변경하기
wine_one_matrix = np.ones((len(wine_idx), len(wine_idx)))

wine_final_dist = wine_one_matrix - wine_regularised

# 도수 유사도, 타 정보 유사도 각각 0.5씩 weight 부여 후 새로운 matrix 생성
wine_new_sim = 0.5 * wine_sim + 0.5 * wine_final_dist

# 확인
print(wine_new_sim)
```

[[1. 0.71660997 0.65957047 ... 0.65957047 0.55027799 0.63608276]
 [0.70966553 1. 0.70204326 ... 0.62489159 0.43589744 0.60416667]
 [0.65773224 0.70367725 1. ... 0.64285714 0.38484398 0.62305335]
 ...
 [0.65773224 0.62652557 0.64285714 ... 1. 0.38484398 0.54590167]
 [0.57912415 0.5 0.43009285 ... 0.43009285 1. 0.45833333]
 [0.63608276 0.61111111 0.62489159 ... 0.54773991 0.42948718 1.]]

```
[ ] # 해당 id의 술이 wine인지 체크
def is_wine(input_id):
    temp_idx = data.index[data['id'] == input_id].tolist() # Int64Index 형식이라 list로 바꾸어줌
    result = data.loc[temp_idx[0]]['class'] == "wine"
    return result

# 와인의 고유 id를 넣으면 해당 와인과 비슷한 Top 10 와인의 id를 return
def wine_recommendation(input_id, wine_new_sim = wine_new_sim):

    # 고유 id로 index 찾기
    idx = data.index[data['id'] == input_id].tolist() # Int64Index 형식이라 list로 바꾸어줌

    # wine_idx list 내에서 몇번째 와인인지 구하기
    w_idx = wine_idx.index(idx[0])

    # 해당 index의 유사도 리스트 sort in descending order
    score_series = pd.Series(wine_new_sim[w_idx]).sort_values(ascending = False)

    # 유사도 Top 10의 index 추출
    wine_top_10_indexes = list(score_series.iloc[1:11].index)

    # 유사도 1인 항목이 하나 더 있어서 자기 자신이 포함되는 경우에는 자신을 뺀 Top 10의 index 재추출
    if wine_top_10_indexes[0] == w_idx:
        wine_top_10_indexes = list(score_series.iloc[1:12].index)
        wine_top_10_indexes.remove(w_idx)

    # 고유 id를 담기 위한 empty list 생성
    wine_top_10_id = []

    # id list
    for i in wine_top_10_indexes:
        index = wine_idx[i] # wine list에서 몇번째인지가 아니라 전체 술 list에서 몇번째인지 구함
        id = data.loc[index]['id']
        wine_top_10_id.append(id)

    return wine_top_10_id
```

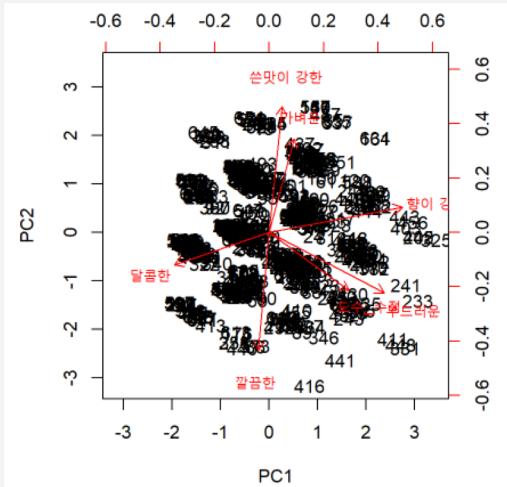


## 2) 지옹 - Cosine Similarity

## 1. 1차 추천

1) PCA - 도수와 특징을 이용. 차원 축소 통해서 2차원으로 나타낸 후 그 데이터를 추천에 이용

```
```{r}
library(proxy)
pca <- prcomp(df_pca, scale. = TRUE)
head(pca$x[,1:2])
screeplot(pca, main = "", col = "green", type = "lines", pch = 1,
npcs = length(pca$sdev))
biplot(pca, scale = 0)
head(pca$x[,1:2])
```
(1) PCA 후 코사인 유사도
```{r}
pca_mat <- as.matrix(pca$x[,1:2])
cosine_pca <- as.matrix(dist(pca_mat, method = "cosine"))
cosine_pca <- as.data.frame(cosine_pca)
recommend_pca <- function(x){
  a <- as.data.frame(cosine_pca[x,])
  names(a) <- df[,4]
  a <- sort(a)
  print(a[,2:11])
}
recommend_pca(1)
```



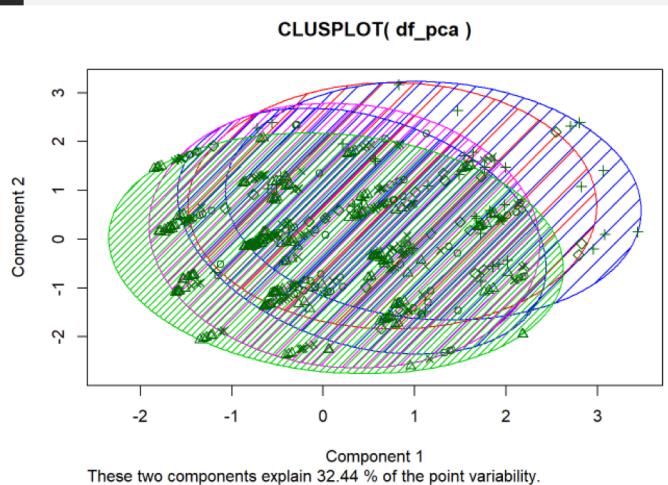
```
##  기린 미치방 1000억 유산균 막걸리 포엑스 골드   참살미L&F
## 1 5.912078e-07      1.037085e-06 6.779194e-06 1.281475e-05
##  미나리 생막걸리 고향생막걸리  부라더#소다 복순도가순막걸리
## 1  1.281475e-05 1.281475e-05 1.410997e-05 2.408544e-05
##  편백숲 산소막걸리 스파클링 헬쌀 누보 막걸리
## 1 3.272098e-05 3.929598e-05
```



## 2) 지옹 - Cosine Similarity

클러스터링을 시도 했지만 특성을 난수로 지정했기 때문에  
클러스터링을 이용하기가 어려움

```
```{r}
library(cluster)
kmeans <- kmeans(x=df_pca, centers =5)
clusplot(df_pca, kmeans$cluster, color = T, shade = T, labels =
13, lines = 0)
```





## 2) 지옹 - Cosine Similarity

```
2) 유사도
  (1) 코사인 유사도
  ``{r}
cosine_dist <- as.matrix(dist(df_pca, method = "cosine"))
cosine_dist <- as.data.frame(cosine_dist)
recommend_c <- function(x){ # 도수와 특징을 이용한 유사도
  a <- as.data.frame(cosine_dist[,x])
  names(a) <- df[,4]
  a <- sort(a)
  print(a[,2:11])
}
recommend_c(1)

  (2) 마할라노비스 거리
  ``{r}
mahalanobis_dist <- as.matrix(dist(df_pca, method =
"mahalanobis"))
mahalanobis_dist <- as.data.frame(mahalanobis_dist)
recommend_m <- function(x){ # 도수와 특징을 이용한 유사도
  a <- as.data.frame(mahalanobis_dist[,x])
  names(a) <- df[,4]
  a <- sort(a)
  print(a[,2:11])
}
recommend_m(1)

  (3) 마하라노비스 거리-1
  ``{r}
df_m <- df[,-c(1:9)] # 도수를 제외하고 특징만을 이용한 유사도
df_mat1 <- as.matrix(df_m)
mahalanobis_dist <- as.matrix(dist(df_mat1, method =
"mahalanobis"))
mahalanobis_dist <- as.data.frame(mahalanobis_dist)
recommend_m1 <- function(x){
  a <- as.data.frame(mahalanobis_dist[,x])
  names(a) <- df[,4]
  a <- sort(a)
  print(a[,2:11])
}
recommend_m1(1)
```

03 구현한 추천 프로그램  
각자 구현한 코드

```
명작복분자 백운복분자주 석로주 스타베리오디 그랑꼬또 M66
1 0.002894362 0.002894362 0.002894362 0.002953518 0.002953518
프리미엄 애플와인 한스오차드 마황주 청명주 부자 송산포도
1 0.003174908 0.003370952 0.003370952 0.004552062
장수홍삼주
1 0.004890276
```

```
석로주 프리미엄 애플와인 한스오차드 마황주 청명주 미화주
2.146295 2.153218 2.175987 2.175987 2.190528
대포막걸리 백운복분자주 스타베리오디 그랑꼬또 M66 명작복분자
2.20992 2.22265 2.282233 2.287653 2.288074
```

```
부라더#소다(바나나) 대포막걸리 프리미엄 애플와인 한스오차드 석로주
2.146291 2.146291 2.146291 2.146291 2.146291
마황주 청명주 미화주 나누우리 막걸리 용막걸리 백운복분자주
2.146291 2.146291 2.146291 2.221461 2.221461 2.221461
```



## 2) 지옹 - Cosine Similarity

```
2. 2차 추천
```{r}
alcl <- read_csv('C:/Users/R/Desktop/alcohol/complete_complete.csv')
df1 <- as.data.frame(alcl)
str(df1)

1) 와인
```{r}
df_wine <- df1 %>%
  filter(술_유형_대분류=="와인")
df_wine <- df_wine[,-c(1:5, 9, 11)]
head(df_wine)
df_wine[,4] <- as.numeric(df_wine[,4])
df_wine[sapply(df_wine, is.character)] <-
  lapply(df_wine[sapply(df_wine, is.character)], as.factor)
head(df_wine)
str(df_wine)
# 빽터가 포함되면 유사도 계산이 안 되기 때문에 일단 제외
df_wine1 <- df_wine[,-c(2,3,5)]
str(df_wine1)

(1) 마할라노비스 거리
```{r}
recommend_wm <- function(x){ # 도수와 가격 특징을 이용한 유사도
  mahal_dist <- as.matrix(dist(df_wine1, method = "mahalanobis"))
  mahal_dist <- as.data.frame(mahal_dist)
  a <- as.data.frame(mahal_dist[x,])
  names(a) <- (df %>% filter(술_유형_대분류=="와인"))[,4]
  a <- sort(a, decreasing = T)
  print(a[,1:10])
}
# recommend_wm(1)
```

보시오, 모스카토 다스티 발레벨보, 모스카토 다스티 품반 멀롯 듀블, 브뤼	6.583996	6.265423	6.148212	6.146096
바타시올로, 바톨로 로버트 바일, 라인가우 리슬링 카비넷	5.811912	5.811625		
위라 위라, 처치 블록 테레 디 산 레오나르도 큐피드 모스카토 다스티	5.761348	5.734048	5.732844	
알랩브리 모스카텔 드 세투발	5.730213			



## 2) 지옹 - Cosine Similarity

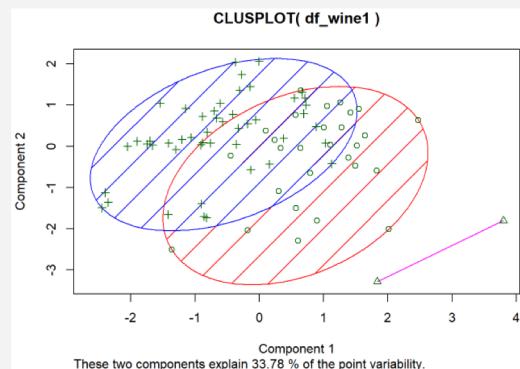
```
(2) 코사인 유사도
```{r}
recommend_wc <- function(x){ # 도수와 가격 특징을 이용한 유사도
  cos_dist <- as.matrix(dist(df_wine1, method = "cosine"))
  cos_dist <- as.data.frame(cos_dist)
  a <- as.data.frame(cos_dist[x,])
  names(a) <- (df %>% filter(술_유형_대분류=="와인"))[,4]
  a <- sort(a, decreasing = T)
  print(a[,1:10])
}
recommend_wc(1)
```

```

```
(3) 클러스터링
```{r}
library(cluster)
kmeans <- kmeans(x=df_wine1, centers = 3)
clusplot(df_wine1, kmeans$cluster, color = T, shade = T, labels = 13, lines = 0)
```

```

|                                         |              |              |              |
|-----------------------------------------|--------------|--------------|--------------|
| 보리 피노 누마 로제 아와테레 소비뇽 블랑 프로베토, 로사토 세미-세코 | 6.327194e-07 | 2.201224e-07 | 1.240406e-07 |
| 산테로, 테스코 모스카토 스푸만테 테레 디 산 레오나르도         | 1.191695e-07 | 9.631664e-08 | 9.546421e-08 |
| 롱반 멀롯                                   |              |              |              |
| 파미유 페랄, 라 비에미유 페름 루즈 타율레로, 몬테풀치아노 다브루즈  | 8.424998e-08 | 8.087654e-08 |              |
| 코노, 피노 누마 레알 쁨빠니아, 가르나차                 | 7.137971e-08 | 6.928774e-08 |              |





## 3) 명진 - Cosine Similarity

```
##사용자가 원하는 주류 입력##  
new_data<-c() ##사용자가 원하는 술 정보를 입력할 빈 데이터  
percent_new<- readline(prompt="원하는 도수 입력:") ##도수입력  
sweet_new<- readline(prompt = '달콤한 술을 원하면 1, 아니면 0 입력:') ##달콤유무 입력  
light_new<- readline(prompt = '가볍게 마실수 있는 술을 원하면 1, 아니면 0 입력:') ##가볍게  
soft_new<- readline(prompt = '목넘김이 부드러운 술을 원하면 1, 아니면 0 입력:') ##부드러움  
bitter_new<- readline(prompt = '쓴맛이 있는 술을 원하면 1, 아니면 0 입력:') ##쓴맛 유무  
clean_new<- readline(prompt = '뒷맛이 깔끔한 술을 원하면 1, 아니면 0 입력:') ##깔끔 유무  
smell_new<- readline(prompt = '향이 강한 술을 원하면 1, 아니면 0 입력:') ##향 유무 입력  
count_head<- readline(prompt="추천받고 싶은 개수:") ##몇개 볼껀지 입력
```

```
> recommend() ##실행  
원하는 도수 입력:|
```

```
> recommend() ##실행  
원하는 도수 입력:13  
달콤한 술을 원하면 1, 아니면 0 입력:1  
가볍게 마실수 있는 술을 원하면 1, 아니면 0 입력:1  
목넘김이 부드러운 술을 원하면 1, 아니면 0 입력:0  
쓴맛이 있는 술을 원하면 1, 아니면 0 입력:1  
뒷맛이 깔끔한 술을 원하면 1, 아니면 0 입력:1  
향이 강한 술을 원하면 1, 아니면 0 입력:0  
추천받고 싶은 개수:10
```



## 3) 명진 - Cosine Similarity

## 2. 사용자의 입력 데이터 정리

```
## 사용자의 입력 데이터 정리##
new_data<-append(new_data,9999) ##사용자 데이터 만들기
new_data<-append(new_data,NA) ##사용자 데이터 만들기
new_data<-append(new_data,NA) ##사용자 데이터 만들기
new_data<-append(new_data,'사용자') ##사용자 데이터 만들기
new_data<-append(new_data,percent_new) ##사용자 데이터 만들기
new_data<-append(new_data,NA) ##사용자 데이터 만들기
new_data<-append(new_data,NA) ##사용자 데이터 만들기
new_data<-append(new_data,NA) ##사용자 데이터 만들기
new_data<-append(new_data,NA) ##사용자 데이터 만들기
new_data<-append(new_data,sweet_new) ##사용자 데이터 정리
new_data<-append(new_data,light_new) ##사용자 데이터 정리
new_data<-append(new_data,soft_new) ##사용자 데이터 정리
new_data<-append(new_data,bitter_new) ##사용자 데이터 정리
new_data<-append(new_data,clean_new) ##사용자 데이터 정리
new_data<-append(new_data,smell_new) ##사용자 데이터 정리
new_mat<-matrix(new_data,ncol=15) ##매트릭스로 만들기
new_df<-as.data.frame(new_mat,stringsAsFactors = F)
new_df[,c(1,5,10,11,12,13,14,15)]<-as.integer(new_df[,c(1,5,10,11,12,13,14,15)])
colnames(new_df)<-colnames(data)
```

```
> new_df
   id class category    name percent origin producer wine_grape whisky_category sweet light soft bitter clean smell
1 9999 <NA>      <NA> 사용자     13    <NA>      <NA>      <NA>          <NA>         1     1     0      1     1     0
```



## 3) 명진 - Cosine Similarity

## 3. 추천시스템 만들기

```
##주류 데이터에 사용자가 원하는 주류 데이터 넣기##
new_data<-rbind(data,new_df)

##코사인 유사도를 이용하여 추천하는 주류 출력 ##
df<-new_data[,c(5,10,11,12,13,14,15)] ##군집화 할때 사용할 변수들(도수 및 특징들6개)
df_scale<-as.data.frame(scale(df)) ##전체 정규화
cosine_dist_Doc_mat <- as.matrix(dist(df_scale, method = "cosine")) ##코사인 유사도
new_data2<-data.frame(new_data$id,cosine_dist_Doc_mat) ##id 옆에 코사인 유사도 붙이기
colnames(new_data2)<-paste0('id',c(0,new_data$id))
new_data3<-new_data2[-nrow(new_data2),] ##new 데이터 제거
fin_data<-new_data3 %>% arrange(new_data3$id9999) ##사용자 데이터로 정렬
recommend_data<-fin_data$id0 ##정렬된거 아이템 이름만 따로 추출
print('추천하는 주류 명단입니다')
return(data %>% filter(id %in% head(c(recommend_data),count_head)))
}
```



### 3) 명진 - Cosine Similarity

#### 4. 함수 실행 및 출력

```

> recommend() ##실행
원하는 도수 입력:13
달콤한 술을 원하면 1, 아니면 0 입력:1
가볍게 마실수 있는 술을 원하면 1, 아니면 0 입력:1
목넘김이 부드러운 술을 원하면 1, 아니면 0 입력:0
쓴맛이 있는 술을 원하면 1, 아니면 0 입력:1
떫맛이 깔끔한 술을 원하면 1, 아니면 0 입력:1
향이 강한 술을 원하면 1, 아니면 0 입력:0
추천받고 싶은 개수:5
[1] "추천하는 주류 명단입니다"
   id class      category          name percent origin producer wine_grape whisky_category sweet light soft bitter clean smell
1    1   wine 와인-레드 도멘 드 벨렌, 부르고뉴 메종 디유        13 프랑스 도멘 드 벨렌 피노 누아           1   1   0   1   1   1   0
2  301 korean 전통주-과실주   프리미엄 애플와인 한스오차드        11                   0   1   0   1   1   1   0
3  352 korean 전통주-약.청주       석로주        13           0   1   0   1   1   1   0
4  359 korean 전통주-약.청주       아황주        17           0   1   0   1   1   1   0
5  396 korean 전통주-약.청주       청명주        17           0   1   0   1   1   1   0

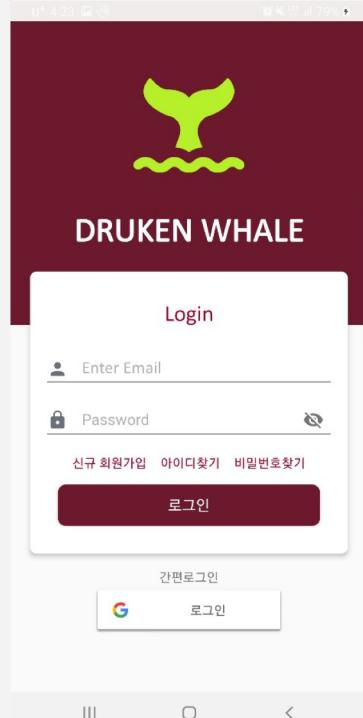
```



주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

## 앱 구성



1. 로그인
2. 회원가입
3. 아이디/비밀번호 찾기
4. 구글 간편 로그인



## 주류 추천 어플리케이션 술고래

### 04 앱 구성 및 시연 앱 구성



1. 술 이름 및 주종 대분류, 중분류를 통한 검색 기능
2. 내가 쓴 평점에 대한 기능 (개발예정)



## 주류 추천 어플리케이션 술고래

# 04 앱 구성 및 시연 앱 구성



1. 주종에 따른 커뮤니티 구성
2. 전체/기타 게시판
3. 글쓰기



## 주류 추천 어플리케이션 술고래

### 04 앱 구성 및 시연

#### 앱 구성



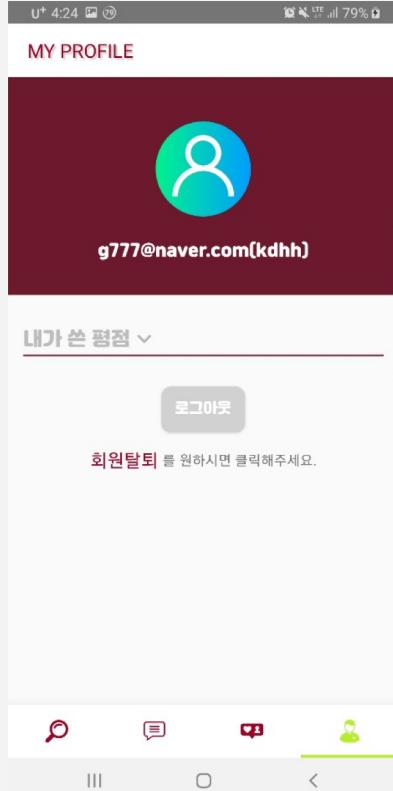
주종 및 취향에 따른 사용자의 즐겨찾기 기능



주류 추천 어플리케이션 술고래

## 04 앱 구성 및 시연

### 앱 구성



1. 프로필 이미지 설정

2. 로그아웃

3. 회원탈퇴



## 주류 추천 어플리케이션 술고래



## 04 앱 구성 및 시연 앱 구성

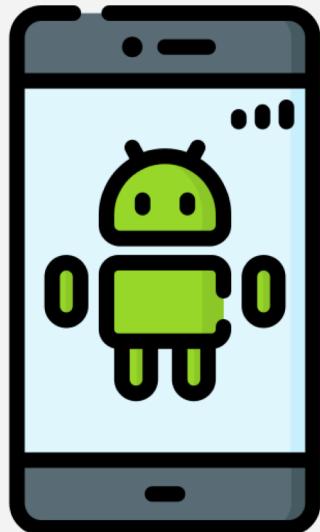
1. 술 주종 및 이름에 따른 도수, 가격 등 상세정보를 보여주는 기능
2. 해당 주종에 맞는 추천 주종을 보여주는 기능
3. 평점 기능 (개발예정)



주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

앱 시연





주류 추천 어플리케이션

THANK YOU  
FOR LISTENING!