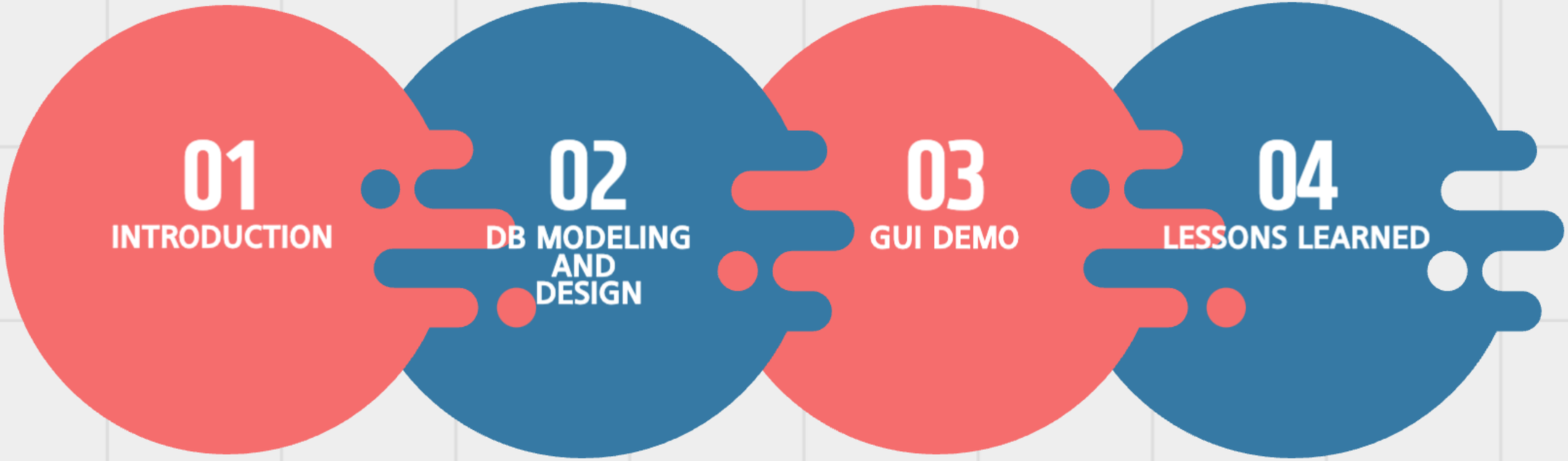
VENDORS FOR YOU

GROUP 5

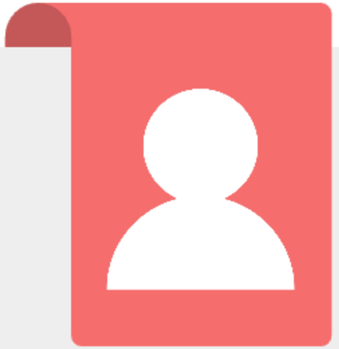
SERIN KIM
JIWON JUNG
SUNWOO HWANG

RELATIONAL DATABASES

OUTLINE


01
INTRODUCTION
02
DB MODELING AND DESIGN
03
GUI DEMO
04
LESSONS LEARNED

Introduction



Target user

- food vendors-seekers

among the people we have location information

Aim

- choose vendors by diverse criteria
- record vendors users are interested in

Introduction

Main function

- Login
- Searching vendors with categories
- Showing 'like' history
- Showing vendors by 4 criteria

Tools

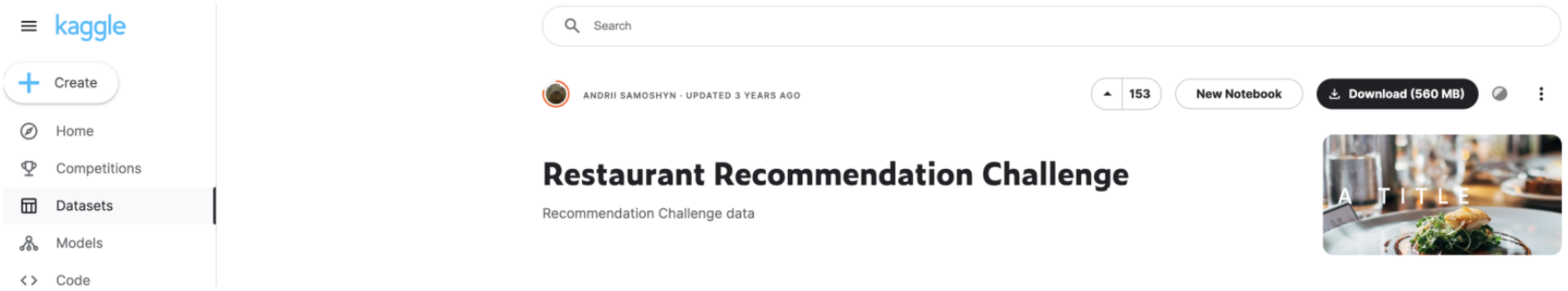
- Language: MySQL, Python
- Packages: Pandas, mysql-connector-python
- GUI Framework: Tkinter

DB MODELING AND DESIGN

DATA EXPLANATION

SOURCE: KAGGLE

https://www.kaggle.com/datasets/mrmorj/restaurant-recommendation-challenge/data



DB MODELING AND DESIGN

DATA EXPLANATION

1. TRAIN_LOCATIONS.CSV → CUSTOMERS.CSV

(59503, 5) TRAIN_LOCATIONS.CSV

customer_id	location_number	location_type	latitude	longitude
02SFNJH	0	NA	1.6823915384615757	-78.78973718
02SFNJH	1	NA	1.6791365450962032	0.766823305388639
02SFNJH	2	NA	-0.498648397	0.661241436545465
RU43CXC	0	Home	0.10085325202709	0.4381654655391503
BDFBPRD	0	NA	2.523125243914389	0.733463724228484
WMGKW6W	0	NA	12.66341321723571	-1.4299192
NDYLK9A	0	NA	-0.203279808	-78.56688511
NDYLK9A	1	NA	-0.204371143	0.4355927487568151
PB2B28D	0	Home	2.575605196878078	0.708270181327257

DB MODELING AND DESIGN

DATA EXPLANATION

1. TRAIN_LOCATIONS.CSV → CUSTOMERS.CSV

```
[4]   1 filtered_df = df[(df['latitude'] > -5) & (df['latitude'] < 5) & (df['longitude'] > -5) & (df['longitude'] < 5)]
      2 filtered_df['location_number'] = filtered_df.groupby('customer_id').cumcount()
```

1) Drop rows that are not in range of −5 < latitude < 5, −5 < longitude < 5
2) Rearrange location numbers from 0

DB MODELING AND DESIGN

DATA EXPLANATION

1. TRAIN_LOCATIONS.CSV → CUSTOMERS.CSV

(37515, 5) CUSTOMERS.CSV

customer_id	location_number	location_type	latitude	longitude
02SFNJH	0		1.679136545	0.766823305
02SFNJH	1		-0.498648397	0.661241437
RU43CXC	0	Home	0.100853252	0.438165466
BDFBPRD	0		2.523125244	0.733463724
NDYLK9A	0		-0.204371143	0.435592749
PB2B28D	0	Home	2.575605197	0.708270181
U9YKW1T	0	Work	0.100016988	0.004357422
GC6WIE3	0		0.875032932	0.536540053
K9BE00O	0		-0.820482585	-0.121503081

DB MODELING AND DESIGN

DATA EXPLANATION

2. VENDORS.CSV → VENDOR_CATEGORY.CSV

id, authentication_id, latitude, longitude, vendor_category_en, vendor_category_id, delivery_charge, serving_distance, is_open, OpeningTime, OpeningTime2, prepration_time, discount_percentage, status, verified, rank, language, vendor_rating, sunday_from_time1, sunday_to_time1, sunday_from_time2, sunday_to_time2, monday_from_time1, monday_to_time1, ……, open_close_flags, vendor_tag, country_id, city_id, updated_at

(100, 59) VENDORS.CSV

is_open	OpeningT	OpeningT	prepratior	commissic	is_akeed_c	discount_p	status	verified	rank	language	vendor_ra	sunday_fr	sunday_to	sunday_fr	sunday_to	monday_fi
1	11:00AM-	-	15	0	Yes	0	1	1	11	EN	4.4	0:00:00	0:30:00	8:00:00	23:59:00	0:00:00
1	08:30AM-	-	14	0	Yes	0	1	1	11	EN	4.7	0:00:00	1:30:00	8:00:00	23:59:00	0:00:00
1	08:00AM-	-	19	0	Yes	0	1	1	1	EN	4.5	8:00:00	22:45:00			8:00:00
1	10:59AM-	-	16	0	Yes	0	1	1	11	EN	4.5	9:00:00	23:30:00			9:00:00
1	11:00AM-	-	10	0	Yes	0	1	1	11	EN	4.4	0:01:00	0:30:00	11:00:00	23:59:00	0:01:00
1	11:00AM-	-	17	0	Yes	0	1	1	11	EN	4.6	8:00:00	23:59:00	0:00:00	1:00:00	8:00:00
1	11:00AM-	-	15	0	Yes	0	1	1	11	EN	4.3	0:01:00	0:30:00	11:00:00	23:59:00	0:01:00
1	11:00AM-	-	14	0	Yes	0	1	1	11	EN	4.3	0:01:00	0:30:00	11:00:00	23:59:00	0:01:00
1	09:00AM-	-	19	0	Yes	0	1	1	11	EN	4.5	9:00:00	23:00:00			9:00:00

DB MODELING AND DESIGN

DATA EXPLANATION

2. VENDORS.CSV → VENDOR_CATEGORY.CSV

VERDORS.CSV

non-atomic

id	latitude	longitude	vendor_tag	vendor_tag_name
4	-0.58859615	0.7544340	2,4,5,8,91,22,12,24,16,23	Arabic,Breakfast,Burgers,Desserts,Free Delivery,Grills,Lebanese,Salads,Sandwiches,Shawarma
13	-0.47165385	0.7444704	4,41,51,34,27,15,24,16,28	Breakfast,Cakes,Crepes,Italian,Pasta,Pizzas,Salads,Sandwiches,Soups
20	-0.40752692	0.6436806	4,8,91,10	Breakfast,Desserts,Free Delivery,Indian
23	-0.58538462	0.75381	5,8,30,24	Burgers,Desserts,Fries,Salads
28	0.4806019230	0.5528504	5	Burgers
33	-0.49465385	0.7433181	8,42	Desserts,Mexican
43	-0.11500962	0.5459734	1,5,30,16	American,Burgers,Fries,Sandwiches
44	-0.93655577	0.08193	1,5,30,16	American,Burgers,Fries,Sandwiches
55	-1.17015385	0.1034772	4,8,22,32,24,16,28	Breakfast,Desserts,Grills,Milkshakes,Salads,Sandwiches,Soups
66	0.4874884615	0.5650499	4,5,8,10,24	Breakfast,Burgers,Desserts,Indian,Salads
67	-0.18103846	0.4905181	4,8,22,32,24,16,28	Breakfast,Desserts,Grills,Milkshakes,Salads,Sandwiches,Soups
75	-0.60042308	0.7559090	3,8,9,11,24,54	Asian,Desserts,Healthy Food,Japanese,Salads,Sushi
76	0.2563346153	0.5766695	16,4,5,52	Sandwiches,Breakfast,Burgers,Mojitos
78	-0.55540385	0.1963363	15,34,4,28,27,24,8	Pizzas,Italian,Breakfast,Soups,Pasta,Salads,Desserts
79	0.6346538461	0.5272545	5,8,91,27,16	Burgers,Desserts,Free Delivery,Pasta,Sandwiches

DB MODELING AND DESIGN

DATA EXPLANATION

2. VENDORS.CSV → VENDOR_CATEGORY.CSV

```
[5]  tag_dict = {}

for idx, row in vendor.iterrows():
    if pd.notnull(row['vendor_tag']) and pd.notnull(row['vendor_tag_name']):
        tags = row['vendor_tag'].split(',')
        tag_names = row['vendor_tag_name'].split(',')

        for tag, tag_name in zip(tags, tag_names):
            if tag not in tag_dict:
                tag_dict[tag] = tag_name

vendor_tag = pd.DataFrame(list(tag_dict.items()), columns=['vendor_tag', 'vendor_tag_name'])
vendor_tag['vendor_tag'] = vendor_tag['vendor_tag'].astype(int)
vendor_tag.sort_values(by=['vendor_tag'], inplace = True)
```

```
category_mapping = {
    'Cuisine': ['American', 'Arabic', 'Asian', 'Chinese', 'Indian', 'Japanese', 'Lebanese', 'Omani', 'Thai', 'Italian', 'Mexican'],
    'Fast Food': ['Burgers', 'Sandwiches', 'Pizzas', 'Hot Dogs', 'Fries'],
    'Healthy Options': ['Healthy Food', 'Salads', 'Vegetarian', 'Organic'],
    'Desserts & Sweets': ['Desserts', 'Cakes', 'Ice creams', 'Sweets', 'Donuts', 'Churros', 'Frozen yoghurt'],
    'Beverages': ['Milkshakes', 'Smoothies', 'Fresh Juices', 'Coffee', 'Hot Chocolate', 'Mojitos', 'Karak', 'Spanish Latte'],
    'Snacks & Sides': ['Waffles', 'Bagels', 'Rolls', 'Fatayers', 'Pancakes', 'Pastry', 'Dimsum'],
    'Grilled & BBQ': ['Grills', 'Shawarma', 'Kebabs', 'Steaks'],
    'Seafood & Meat': ['Seafood', 'Biryani', 'Sushi'],
    'Bakery & Dough': ['Cafe', 'Pasta', 'Soups', 'Crepes', 'Manakeesh', 'Pizza', 'Pastas']
}

flattened_categories = [(major, sub) for major, subs in category_mapping.items() for sub in subs]

vendor_category = pd.DataFrame(columns=['Major Category', 'Sub Category', 'vendor_tag_num'])

for major, sub in flattened_categories:
    tag_nums = vendor_tag.loc[vendor_tag['vendor_tag_name'].str.lower() == sub.lower(), 'vendor_tag']
    if not tag_nums.empty:
        vendor_category = vendor_category.append({
            'Major Category': major,
            'Sub Category': sub,
            'vendor_tag_num': tag_nums.iloc[0]
        }, ignore_index=True)

vendor_category
```

1) Splitted tags and tag names from rows. These tags and tag names are then stored in tag_dict, which maps each tag ID to its corresponding name.

2) Category_mapping is defined, which maps major categories to their subcategories.

3) Created a Vendor Category DataFrame by finding the corresponding tag number for each subcategory

DB MODELING AND DESIGN

DATA EXPLANATION

2. VENDORS.CSV → VENDOR_CATEGORY.CSV

Major_category	Sub_category	vendor_tag_num
Cuisine	American	1
Cuisine	Arabic	2
Cuisine	Asian	3
Cuisine	Chinese	7
Cuisine	Indian	10
Cuisine	Japanese	11
Cuisine	Lebanese	12
Cuisine	Omani	13
Cuisine	Thai	19
Cuisine	Italian	34
Cuisine	Mexican	42
Fast Food	Burgers	5
Fast Food	Sandwiches	16
Fast Food	Pizzas	15
Fast Food	Hot Dogs	29

DB MODELING AND DESIGN

DATA EXPLANATION

3.VENDORS.CSV → VENDOR_TAG.CSV

VENDOR_TAG.CSV

id	vendor_tag
4	2
4	4
4	5
4	8
4	91
4	22
4	12
4	24
4	16
4	23
13	4
13	41
13	51
13	34

DB MODELING AND DESIGN

DATA EXPLANATION

4. VENDORS.CSV → VENDORS.CSV

VENDORS.CSV(preprocessed): Select features that can be sorted, and divide business hours into opening and closing time

vendor_id	latitude	longitude	delivery_charge	serving_distance	open_time	end_time	is_open	preparation_time	discount_percentage	vendor_rating
4	-0.5885961538	0.7544340909	0	6	11:00	23:30	1	15	0	4.4
13	-0.4716538462	0.7444704545	0.7	5	8:30	22:30	1	14	0	4.7
20	-0.4075269231	0.6436806818	0	8	8:00	22:45	1	19	0	4.5
23	-0.5853846154	0.7538113636	0	5	10:59	22:30	1	16	0	4.5
28	0.4806019231	0.5528504545	0.7	15	11:00	23:45	1	10	0	4.4
33	-0.4946538462	0.7433181818	0.7	6	11:00	22:30	1	17	0	4.6
43	-0.1150096154	0.5459734091	0.7	15	11:00	23:45	1	15	0	4.3
44	-0.9365557692	0.08193295455	0.7	15	11:00	23:45	1	14	0	4.3
55	-1.170153846	0.1034772727	0.7	10	9:00	23:30	1	19	0	4.5
66	0.4874884615	0.56505	0	15	17:00	23:00	1	20	10	4
67	-0.1810384615	0.4905181818	0	15	8:00	23:30	1	15	0	4.3
75	-0.6004230769	0.7559090909	0.7	5	11:00	22:30	1	15	0	4.6
76	0.2563346154	0.5766696455	0.7	15	8:30	21:30	1	15	0	4.6
78	-0.5554038462	0.1963363636	0.7	15	11:00	23:00	0	17	0	4.4
79	0.6346538462	0.5272545455	0	15	11:59	2:15	1	15	0	4.7
81	-0.7119615385	0.06336818182	0	15	8:00	12:30	1	15	0	3.8
82	-0.4171923077	0.64585	0.7	8	8:00	23:59	1	11	0	4.4
83	-0.9279442308	0.1458995455	0.7	15	11:00	23:00	1	15	0	4.2

DB MODELING AND DESIGN

DATA EXPLANATION

5. ORDERS.CSV → LIKES.CSV

ORDERS.CSV

customer_	item_coun	grand_tot	payment_	promo_co	vendor_di	promo_co	is_favorite	is_rated	vendor_rating
92PEE24	1	7.6	2		0			No	
QS68UD8	1	8.7	1		0			No	
MB7VY5F	2	14.4	1		0			No	
KDJ951Y	1	7.1	1		0			No	
BAL0RVT	4	27.2	1		0			No	
U263OCD	3	18.2	2		0			No	
I9DNSMJ	2	14.4	1		0			No	
I9DNSMJ	2	14.4	1		0			No	
QYXXJCF	2	9.2	1		0			No	
52ZNRWH	5	17.1	2		0			No	
QQEWRHI	1	8.7	1		0			No	
DRR8CO9	2	11.4	1		0			No	
VJP7GIV	1	5.7	1		0			No	

DB MODELING AND DESIGN

DATA EXPLANATION

5. ORDERS.CSV → LIKES.CSV

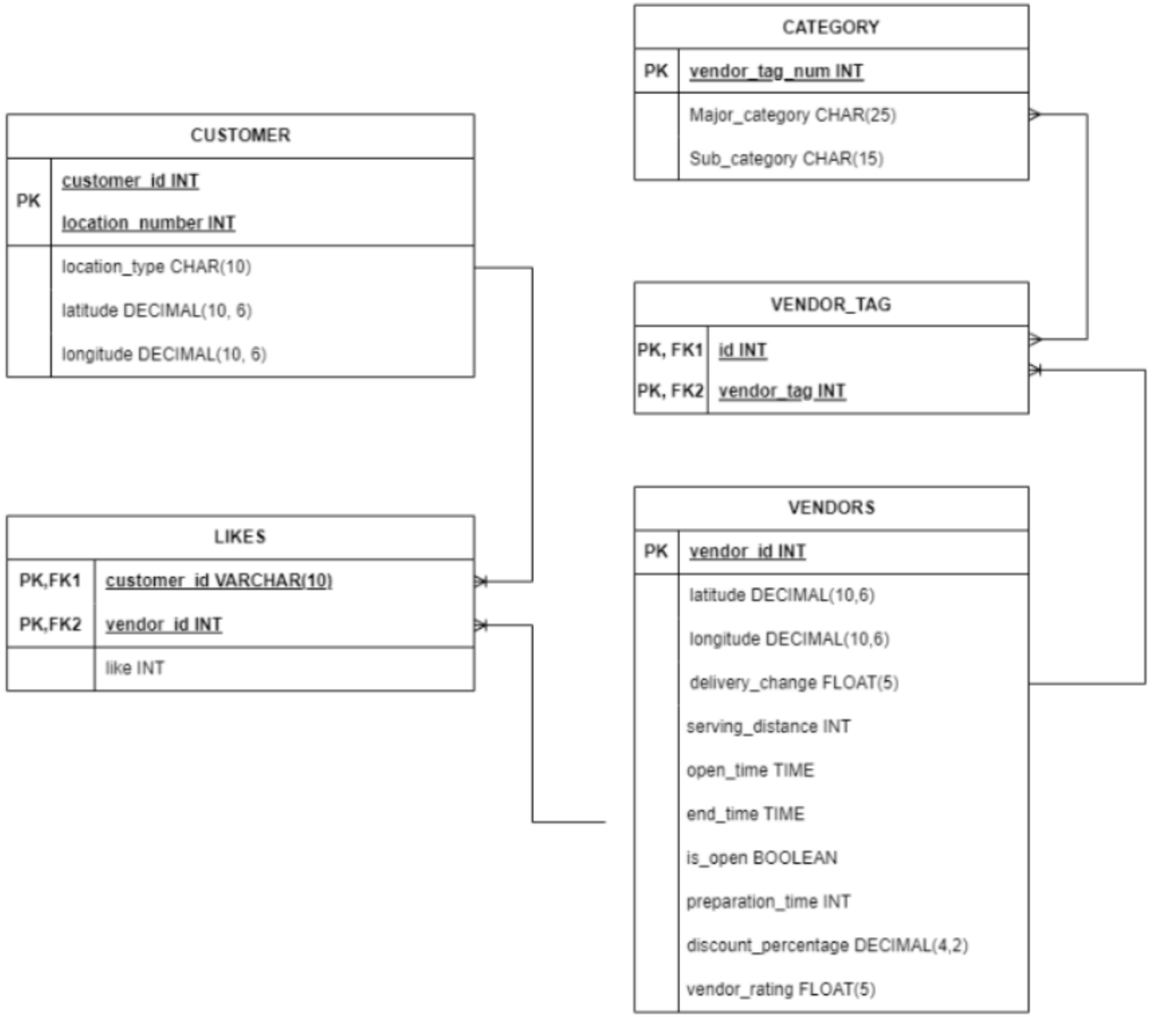
LIKES.CSV

customer_	vendor_id	like
OF8VT8O	105	1
30PZXDS	78	1
30PZXDS	115	1
30PZXDS	189	1
6S9427H	243	1
EBVHPAO	193	1
FKG1F3U	104	1
GE2IW1I	83	1
1P5RQ9H	115	1
2QBB6A9	106	1
FZXHMXG	193	1
R4WTEY6	221	1
R4WTEY6	195	1
ECT8K2K	85	1
UOURCZ1	78	1

If the user gives a rating of 5 points,
it will be interpreted as "like"

DB MODELING AND DESIGN

ER DIAGRAM



MAIN QUERY

1. FETCING VENDORS

```
# Create a list to put queries in the nearest order
vendors_info = []
for vendor_id, _ in vendors_with_distance:
    myCursor.execute("""
        SELECT vendor_id, vendor_rating, delivery_charge, preparation_time, open_time,end_time,is_open,sum(`like`)
        FROM vendors JOIN likes USING (vendor_id)
        GROUP BY vendor_id
        HAVING vendor_id = %s
    """, (vendor_id,))
    vendor_data = myCursor.fetchone()
    if vendor_data:
        vendors_info.append(vendor_data)
```

MAIN QUERY

2. UPDATING 'LIKE' STATUS

```
# function for updating 'like' status
def update_like_main(values,user,btn):
    vendor_id = values[0]
        # if a vendor's status is 'like', the status is switched to 'normal'
    if btn.cget('text') == '♥':
        update_query = f"UPDATE likes SET `like` = 0 WHERE vendor_id = {vendor_id} && customer_id = '{user}'"
        myCursor.execute(update_query)
        mydb.commit()
        btn['text'] = '♡'
        # if a vendor's status is 'normal', the status is switched to 'like'
    else:
        try: # Use "UPDATE" if the customer have ever pressed "like" before
            like_dic[vendor_id]
            update_query = f"UPDATE likes SET `like` = 1 WHERE vendor_id = {vendor_id} && customer_id = '{user}'"
        except: # Use "INSERT" if the customer have never pressed "like" before
            update_query = f"INSERT INTO likes VALUES ('{user}',{vendor_id},1)"
            like_dic[vendor_id] = 1
        myCursor.execute(update_query)
        mydb.commit()
        btn['text'] = '♥'
```

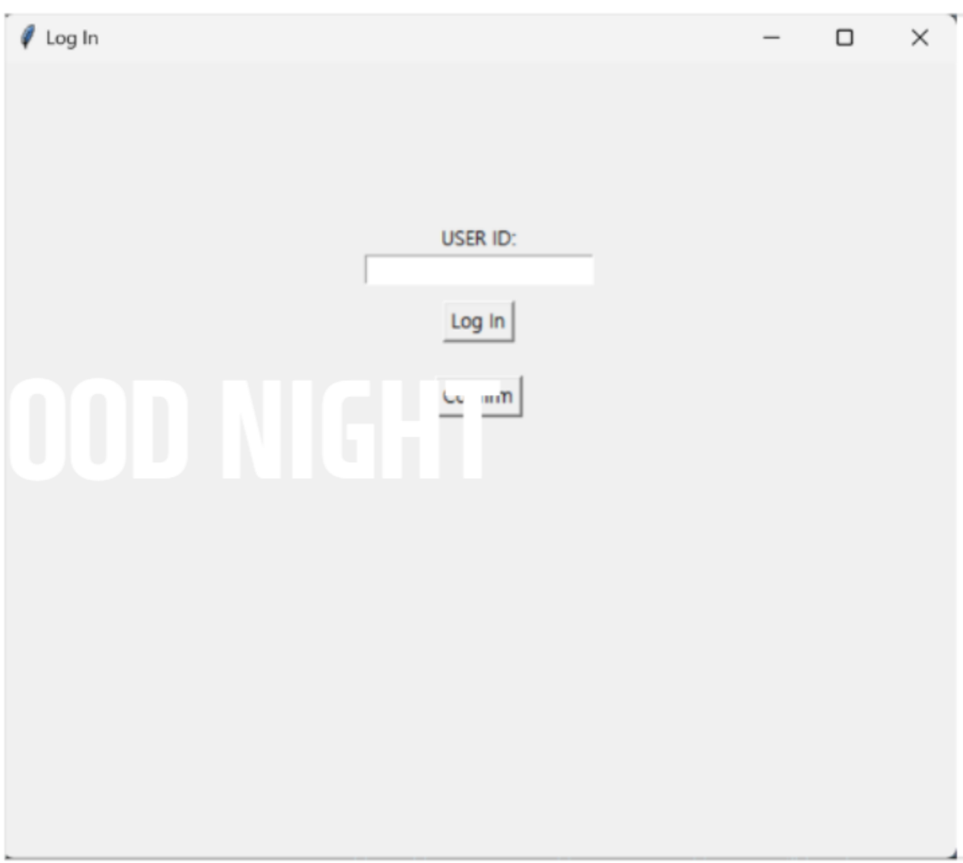
MAIN QUERY

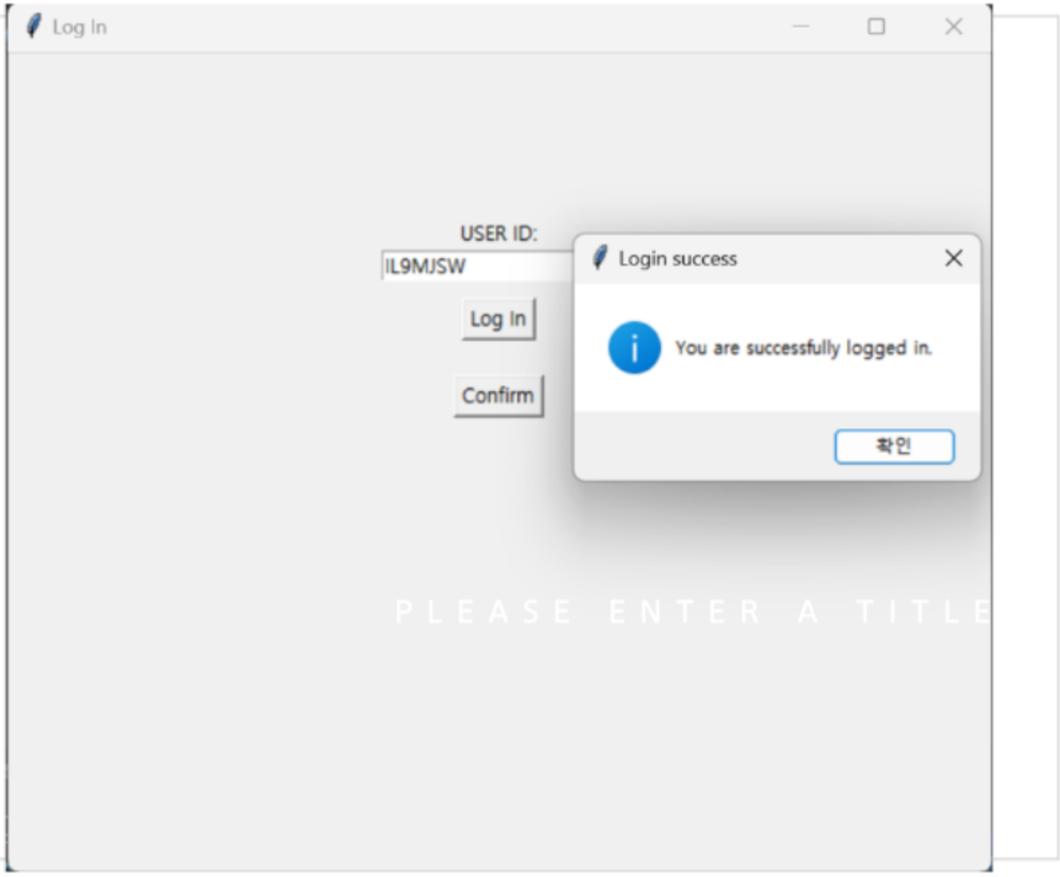
3. SORTING VENDORS

```
query = f"SELECT vendor_id, vendor_rating, delivery_charge, preparation_time, open_time,end_time,is_open,sum(`like`) FROM vendors JOIN likes USING (vendor_id) GROUP BY vendor_id HAVING vendor_id IN {vendor_ids_sql}"
################################################
if sort_by == 'delivery':  # Sort by delivery charge
    query += " ORDER BY delivery_charge;"
elif sort_by == 'preparation': # Sort by food preparation time
    query += " ORDER BY preparation_time;"
elif sort_by == 'rating':  # Sort by rating
    query += " ORDER BY vendor_rating DESC;"
elif sort_by == 'open':   # Restaurant open by user's connected time
    query += " AND HOUR(CURTIME()) BETWEEN HOUR(open_time) AND HOUR(end_time) ORDER BY vendor_id;"
```

GUI DEMO

LOGIN SCREEN



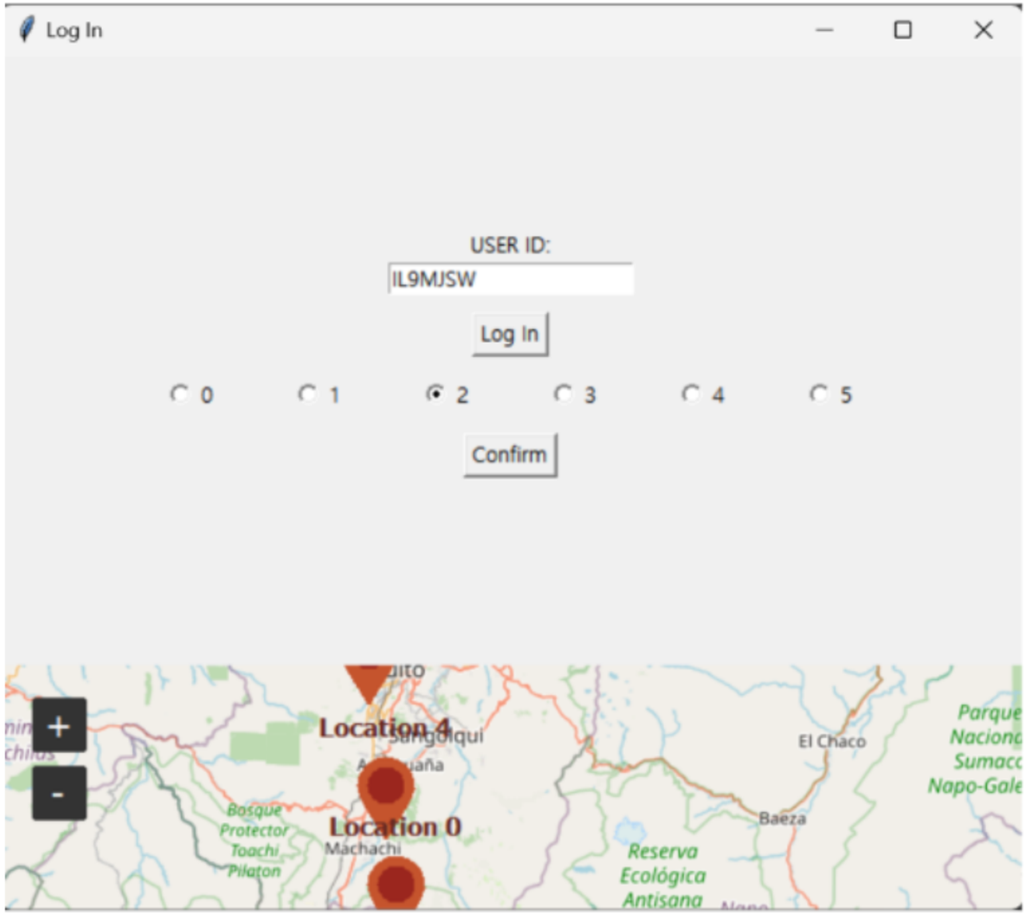


GUI DEMO

LOGIN SCREEN

When the login is complete, the user's location is selected.

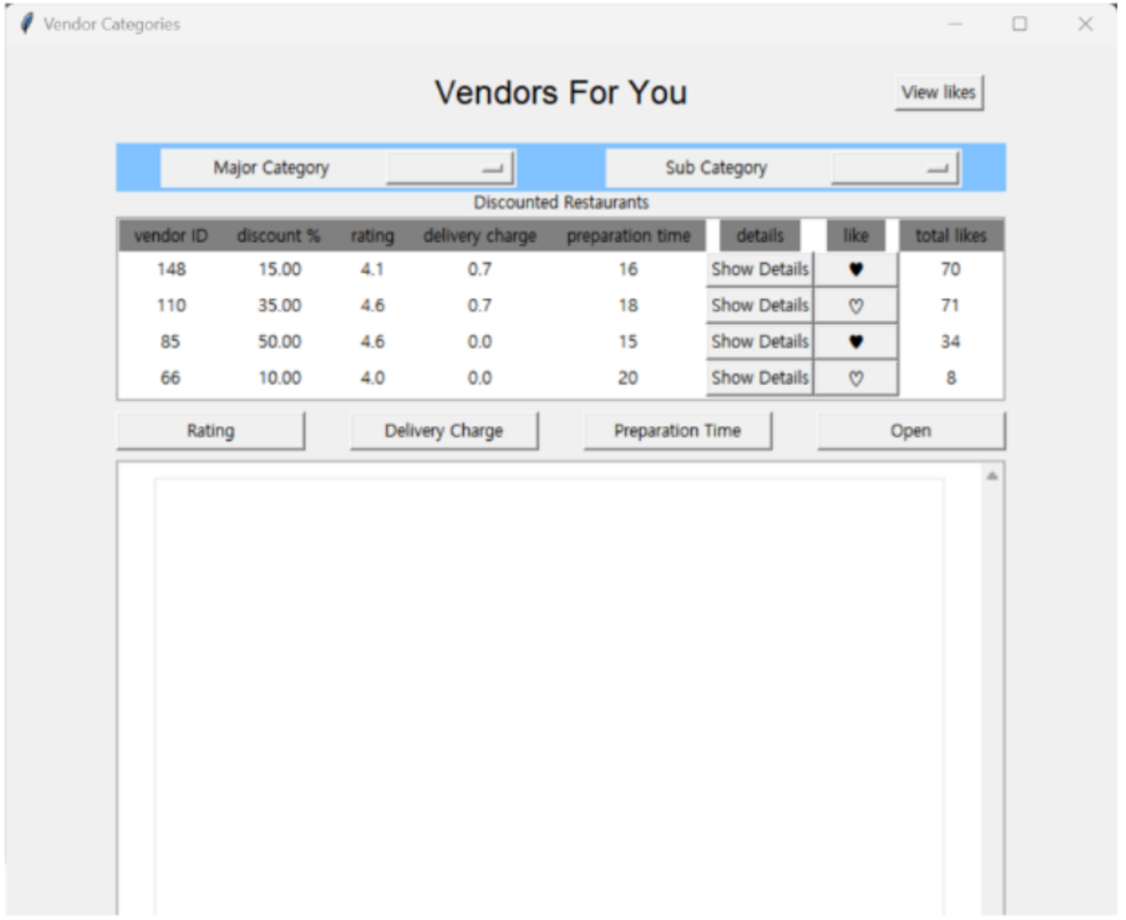
- specify the location number
- select the number
- press confirm



GUI DEMO

MAIN SCREEN

screen for selecting categories

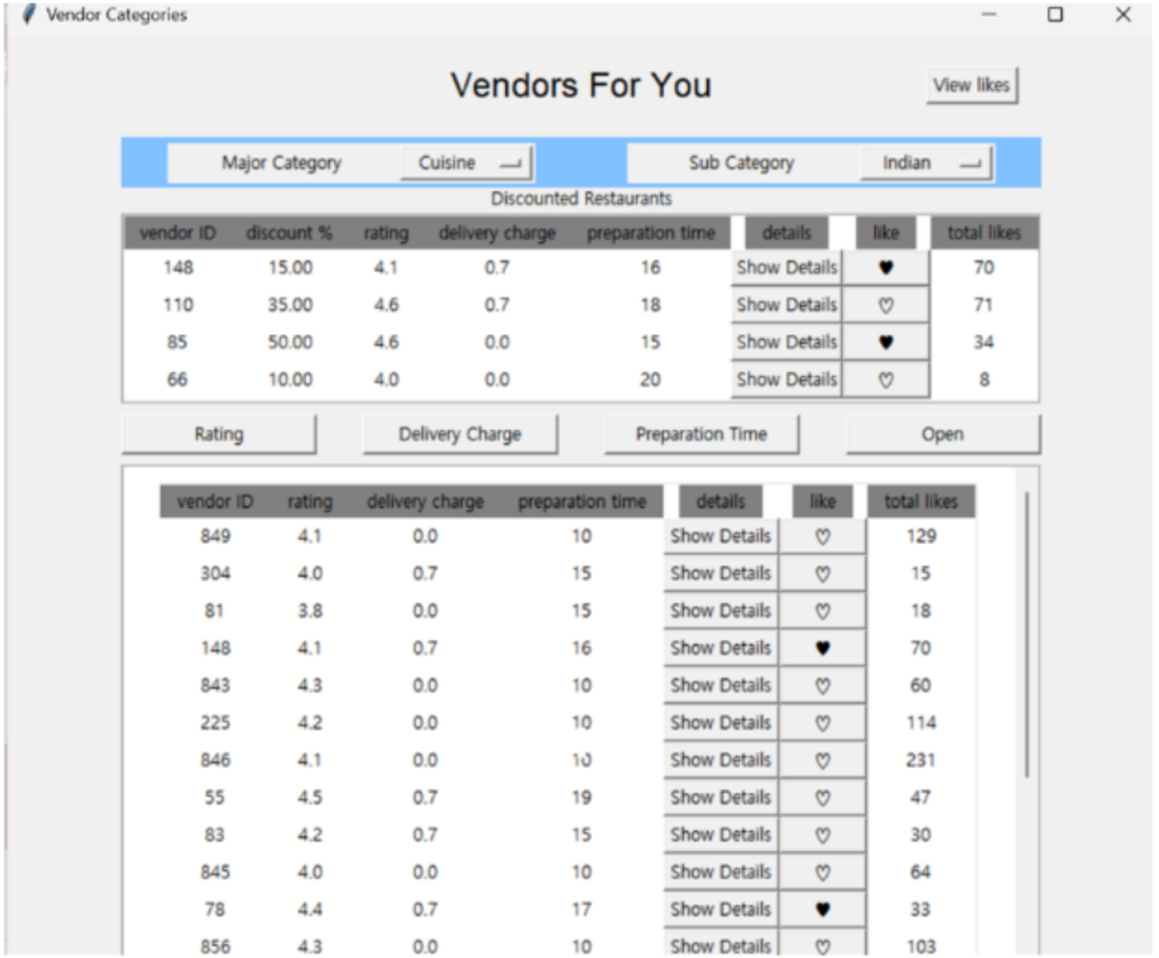


GUI DEMO

MAIN SCREEN

When a major category and sub category are selected,

- vendors are sorted in distance order
  from the user's location in the box below

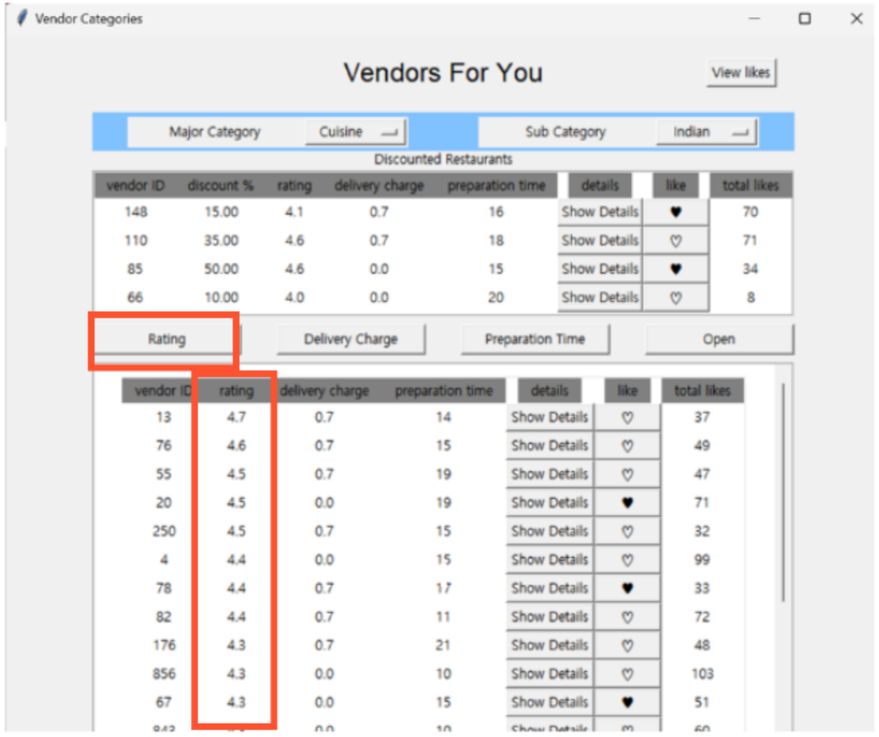


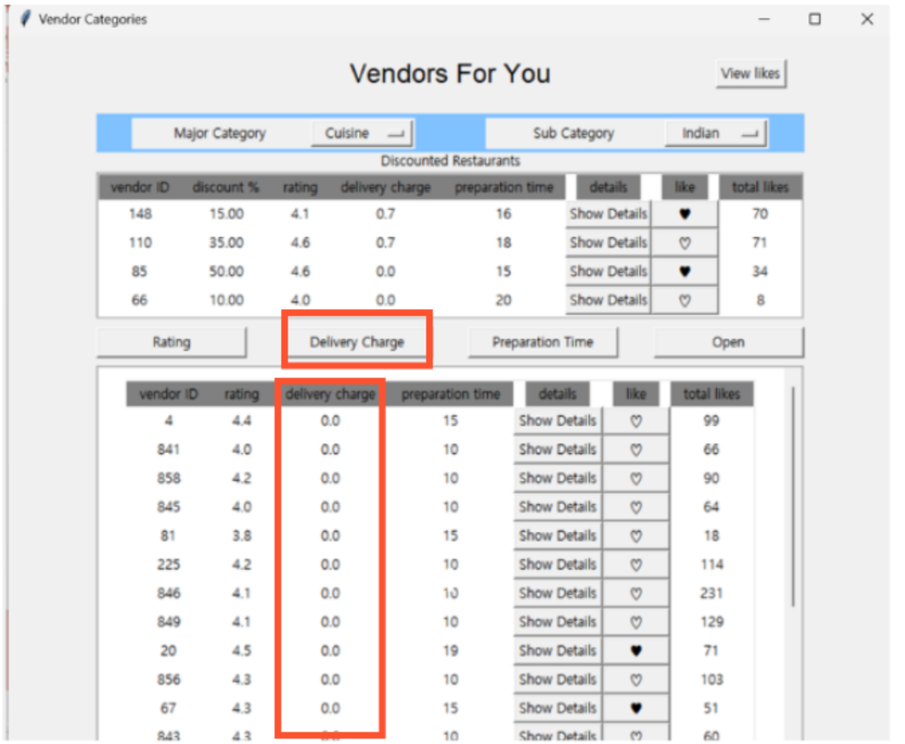
GUI DEMO

MAIN SCREEN

SORTED BY

- rating
- delivery charge



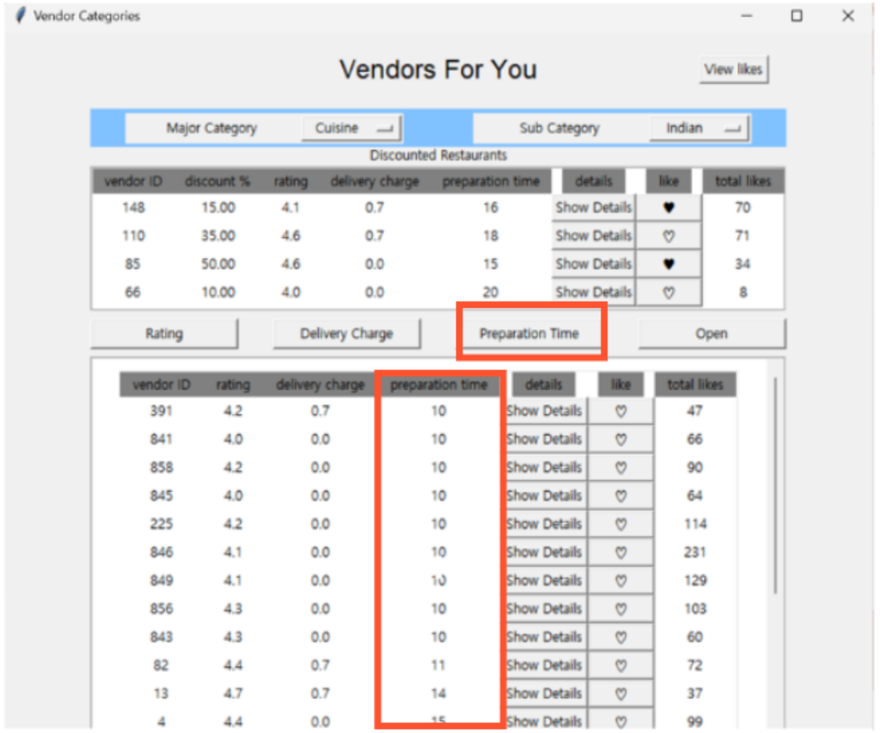


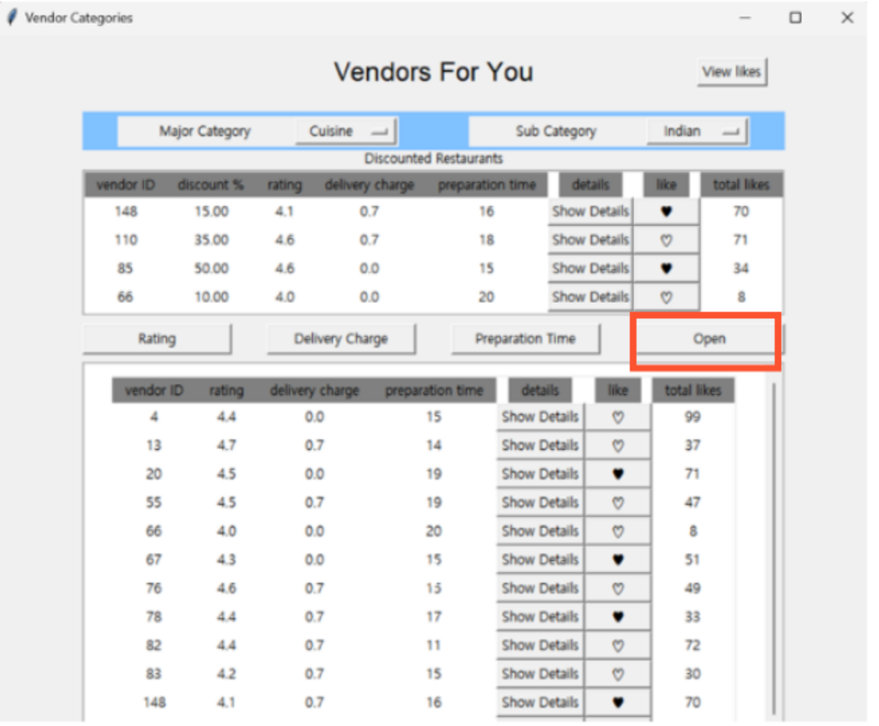
GUI DEMO

MAIN SCREEN

SORTED BY

- preparation time
- open(filtering)

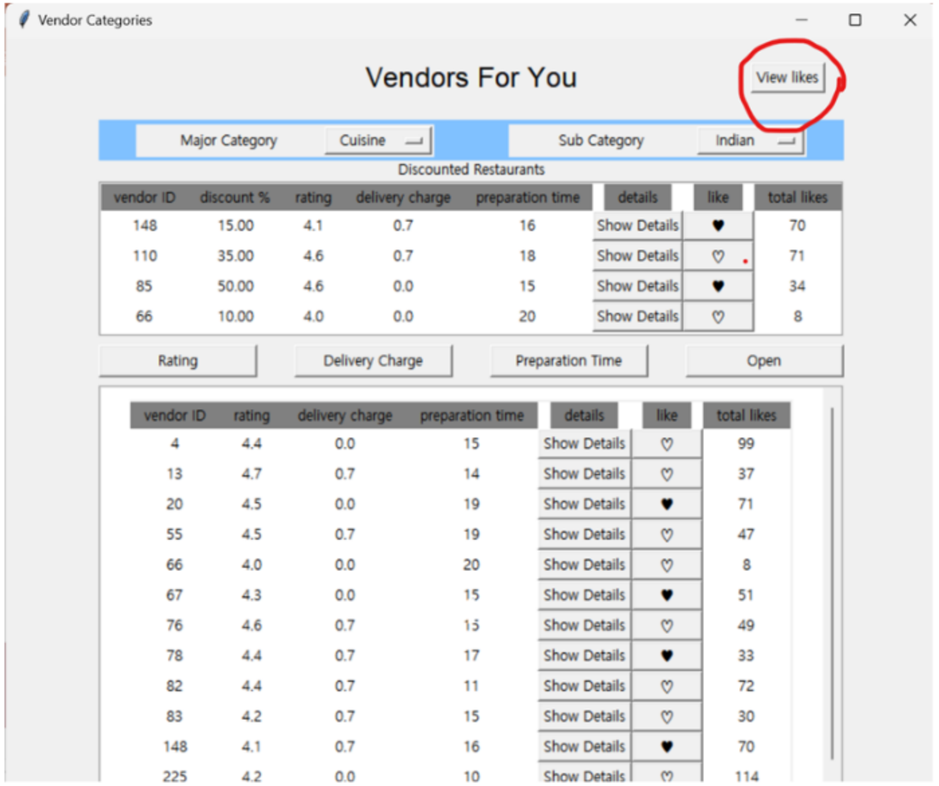


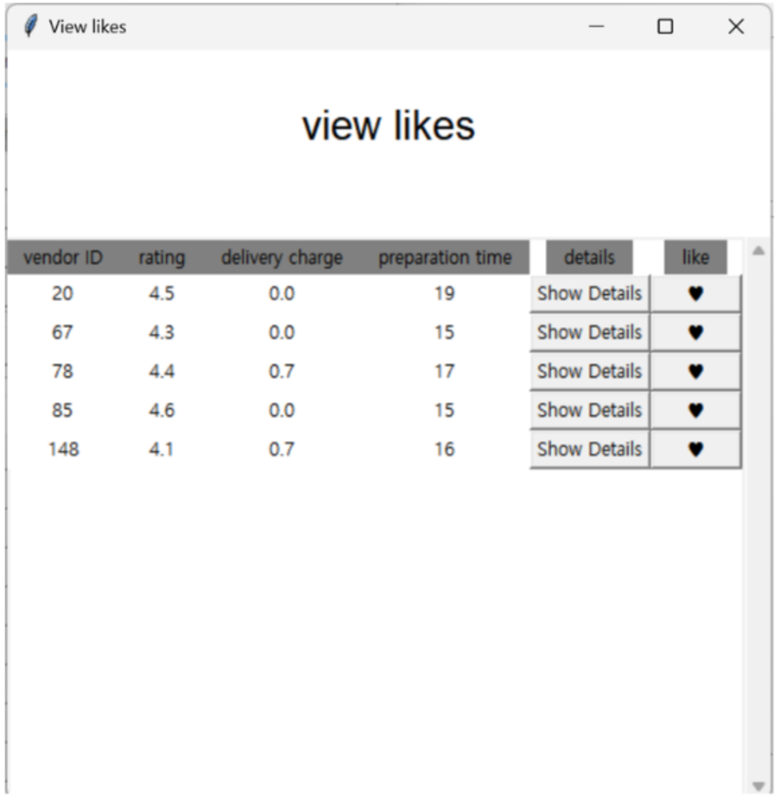


GUI DEMO

VIEW LIKE SCREEN

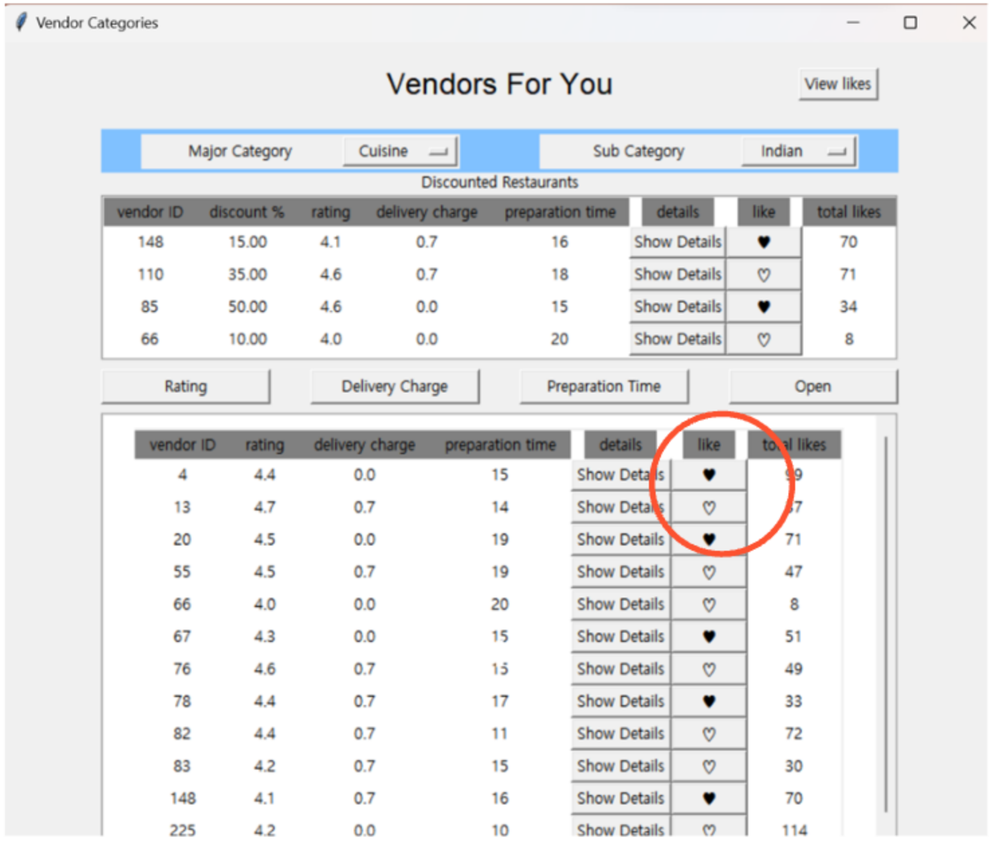
Only stoires that users like can see separately

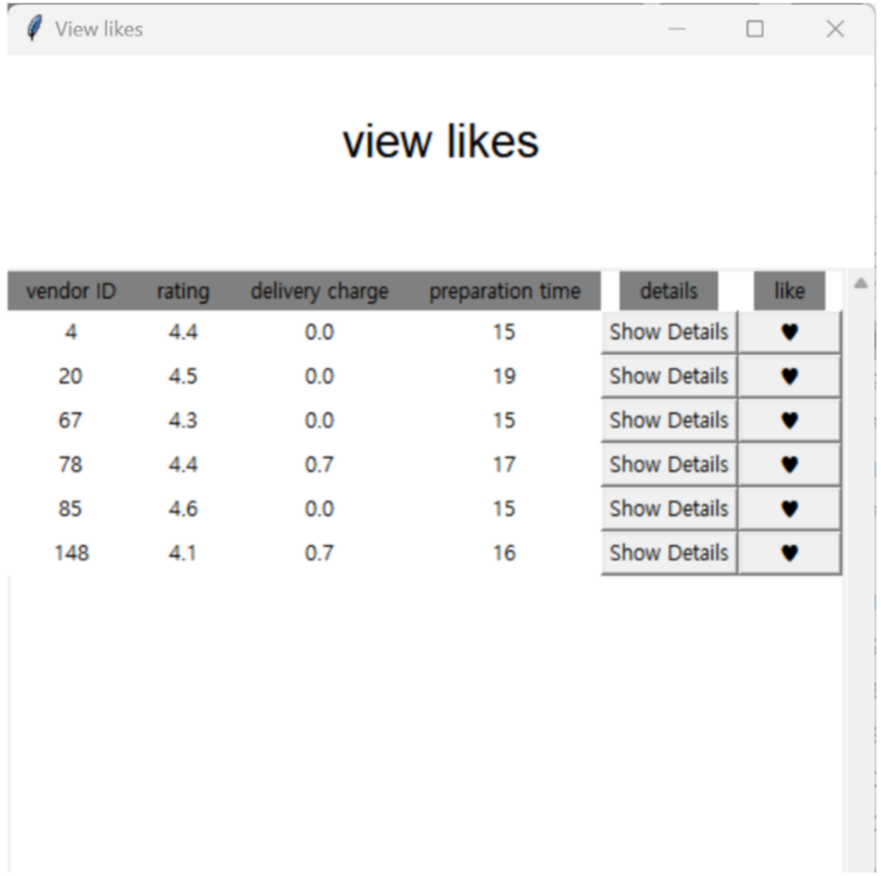




GUI DEMO

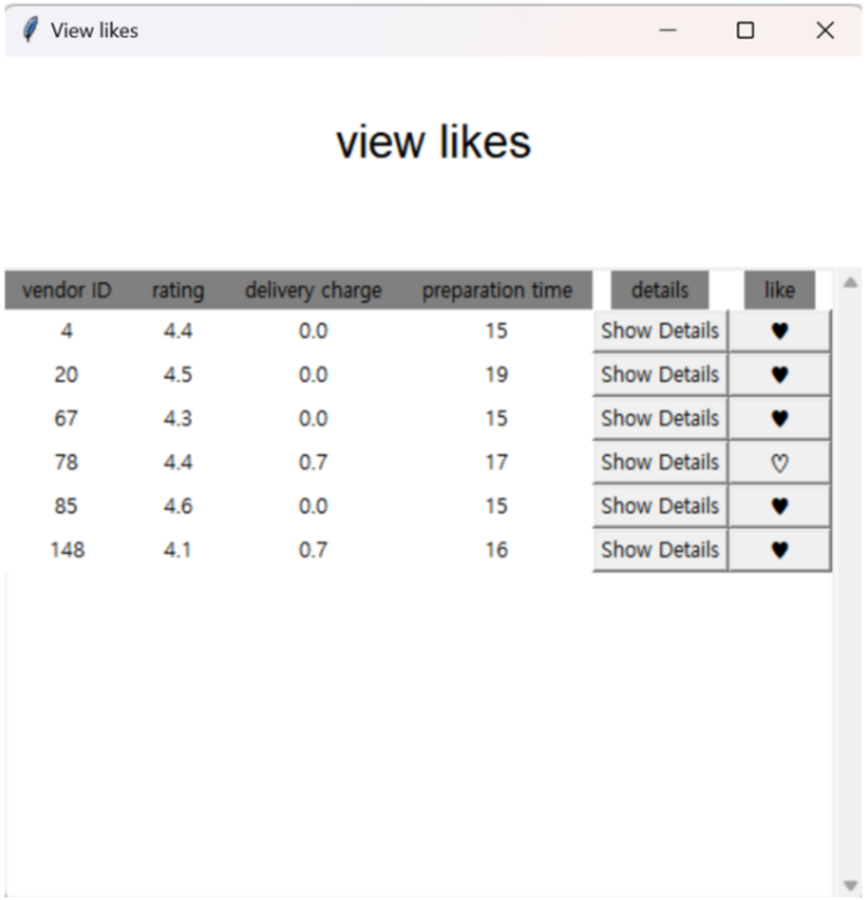
VIEW LIKE SCREEN

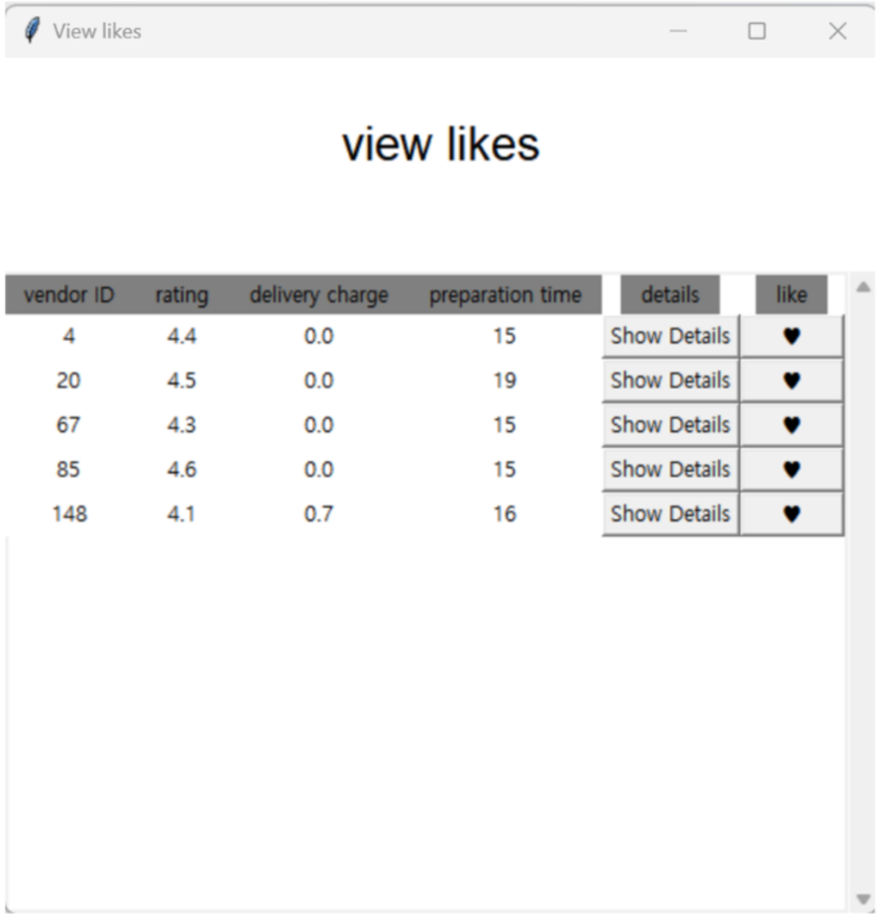




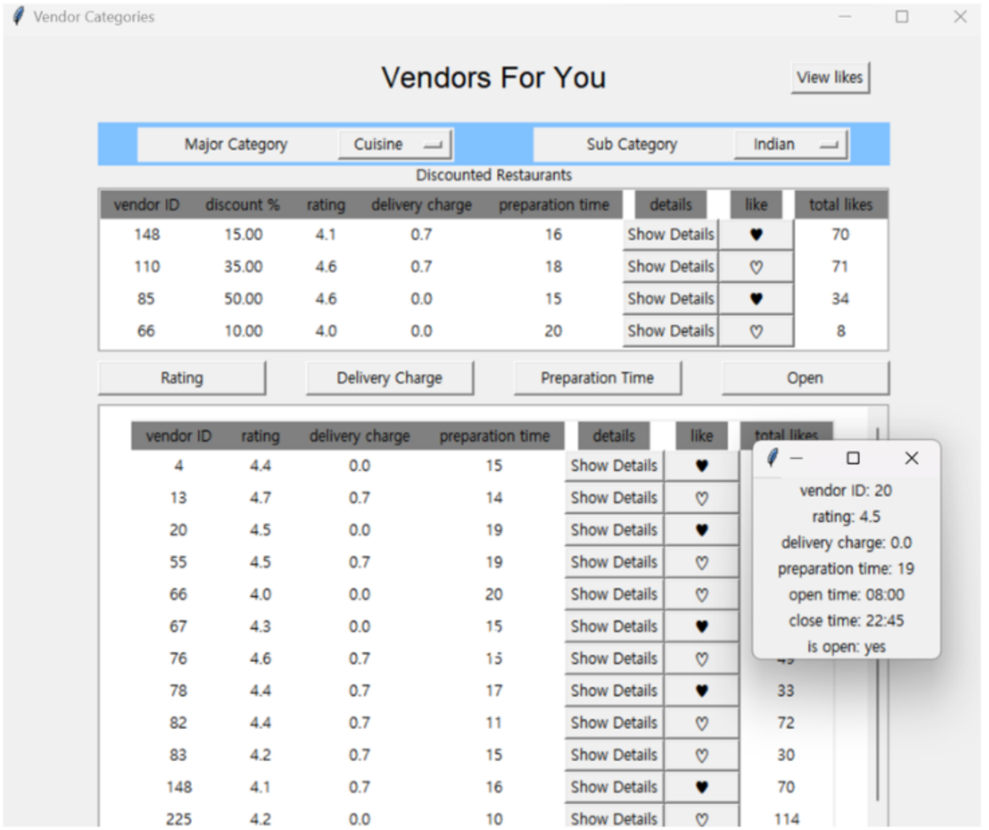
GUI DEMO

VIEW LIKE SCREEN



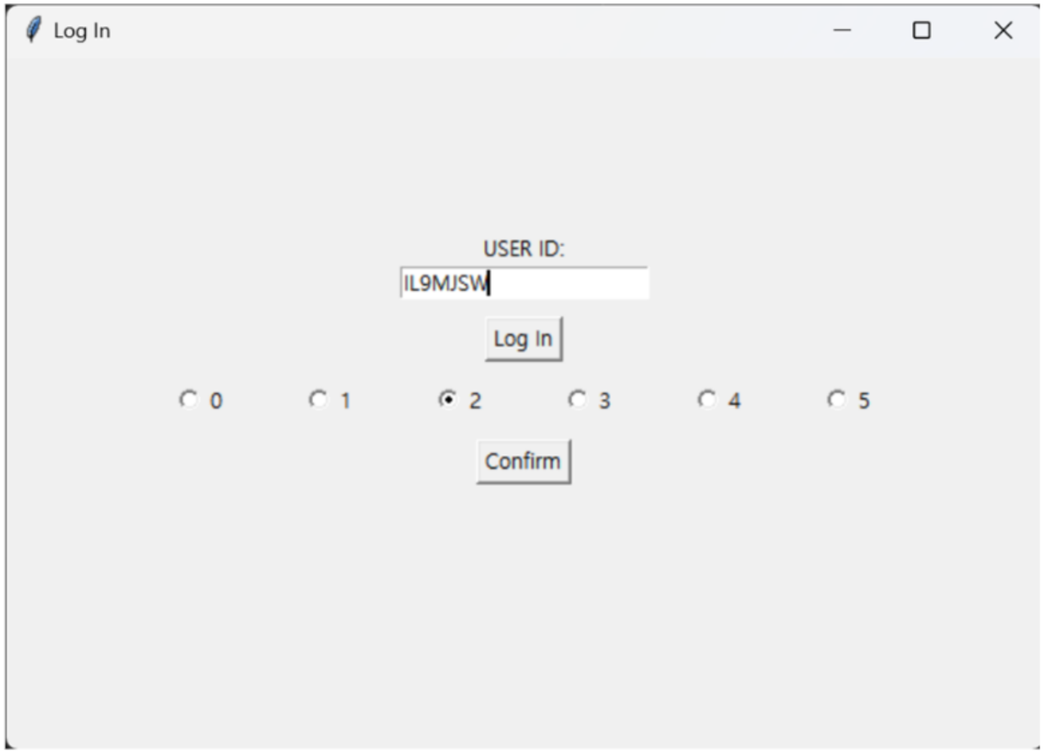


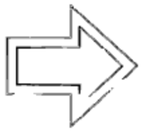
GUI DEMO

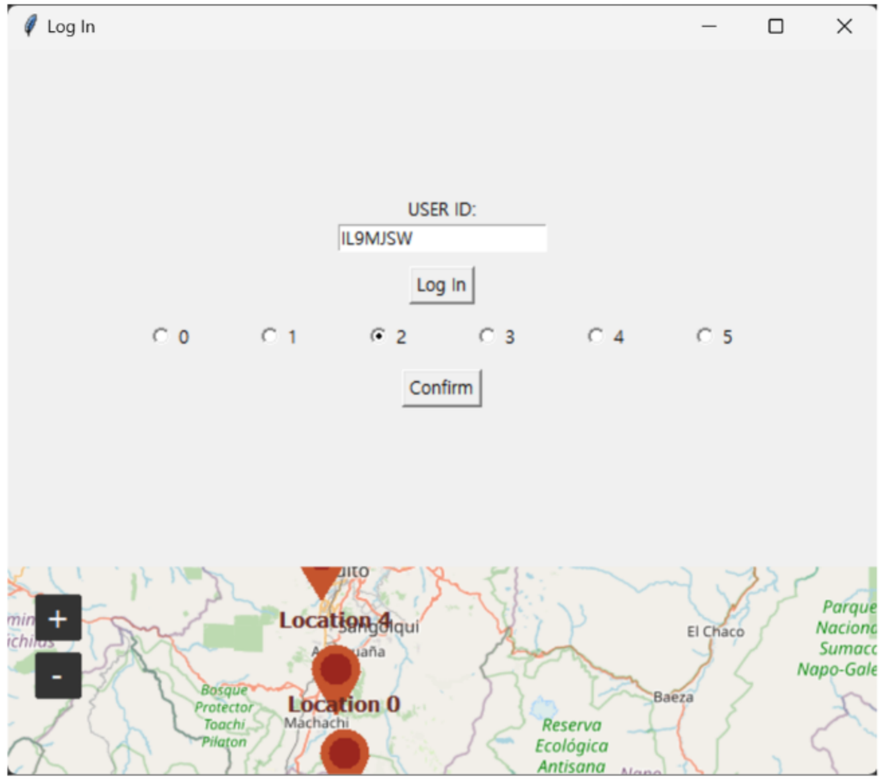


Click the Show details button to pop up information about the vendors

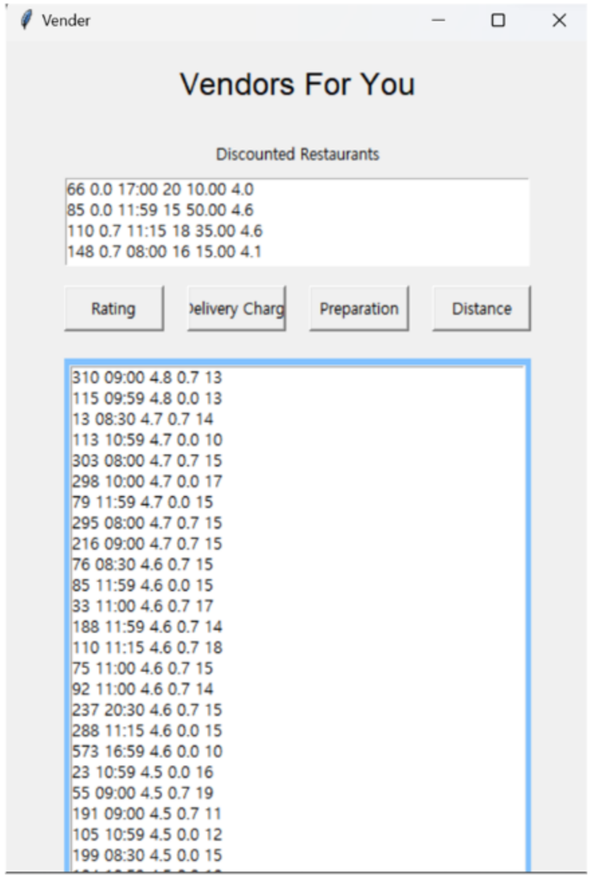
GUI CHANGES

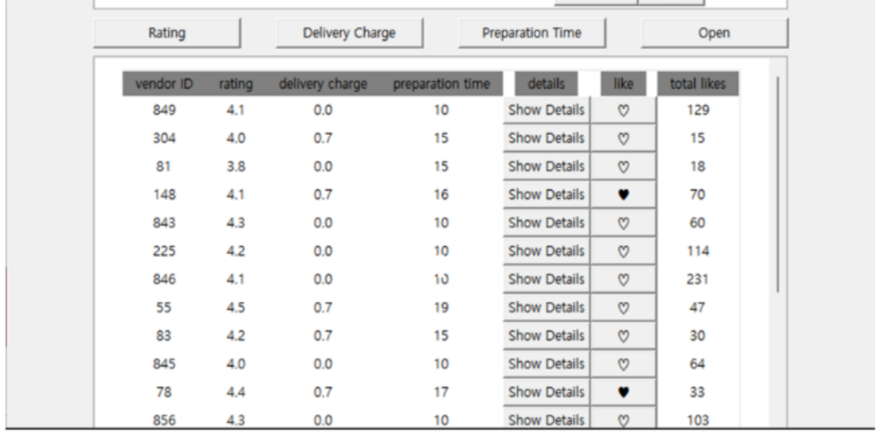

Log In
USER ID:
IL9MJSW
Log In
0
1
2
3
4
5
Confirm




Log In
USER ID:
IL9MJSW
Log In
0
1
2
3
4
5
Confirm
Location 4
Location 0
Machachi
El Chaco
Baeza
Reserva
Ecológica
Antisana
Bosque
Protector
Toachi
Pilaton

GUI CHANGES


Vender
Vendors For You
Discounted Restaurants
66 0.0 17:00 20 10.00 4.0
85 0.0 11:59 15 50.00 4.6
110 0.7 11:15 18 35.00 4.6
148 0.7 08:00 16 15.00 4.1
Rating
elivery Charg
Preparation
Distance
310 09:00 4.8 0.7 13
115 09:59 4.8 0.0 13
13 08:30 4.7 0.7 14
113 10:59 4.7 0.0 10
303 08:00 4.7 0.7 15
298 10:00 4.7 0.0 17
79 11:59 4.7 0.0 15
295 08:00 4.7 0.7 15
216 09:00 4.7 0.7 15
76 08:30 4.6 0.7 15
85 11:59 4.6 0.0 15
33 11:00 4.6 0.7 17
188 11:59 4.6 0.7 14
110 11:15 4.6 0.7 18
75 11:00 4.6 0.7 15
92 11:00 4.6 0.7 14
237 20:30 4.6 0.7 15
288 11:15 4.6 0.0 15
573 16:59 4.6 0.0 10
23 10:59 4.5 0.0 16
55 09:00 4.5 0.7 19
191 09:00 4.5 0.7 11
105 10:59 4.5 0.0 12
199 08:30 4.5 0.0 15


Vendor Categories
Vendors For You
View likes
Major Category
Cuisine
Sub Category
Indian
Discounted Restaurants
vendor ID | discount % | rating | delivery charge | preparation time | details | like | total likes
148 | 15.00 | 4.1 | 0.7 | 16 | Show Details | ♥ | 70
110 | 35.00 | 4.6 | 0.7 | 18 | Show Details | ♡ | 71
85 | 50.00 | 4.6 | 0.0 | 15 | Show Details | ♥ | 34
66 | 10.00 | 4.0 | 0.0 | 20 | Show Details | ♡ | 8
Rating
Delivery Charge
Preparation Time
Open
vendor ID | rating | delivery charge | preparation time | details | like | total likes
849 | 4.1 | 0.0 | 10 | Show Details | ♡ | 129
304 | 4.0 | 0.7 | 15 | Show Details | ♡ | 15
81 | 3.8 | 0.0 | 15 | Show Details | ♡ | 18
148 | 4.1 | 0.7 | 16 | Show Details | ♥ | 70
843 | 4.3 | 0.0 | 10 | Show Details | ♡ | 60
225 | 4.2 | 0.0 | 10 | Show Details | ♡ | 114
846 | 4.1 | 0.0 | 10 | Show Details | ♡ | 231
55 | 4.5 | 0.7 | 19 | Show Details | ♡ | 47
83 | 4.2 | 0.7 | 15 | Show Details | ♡ | 30
845 | 4.0 | 0.0 | 10 | Show Details | ♡ | 64
78 | 4.4 | 0.7 | 17 | Show Details | ♥ | 33
856 | 4.3 | 0.0 | 10 | Show Details | ♡ | 103

LESSONS LEARNED

CHANLLENGES

- vendors' ratings that cannot be obtained by averaging users' ratings
- difficult to show only vendors available at the current time
- queries that depend on the existence of premary key values in the database
- anonymization of longitude and latitude data
- unspecified values on 'location type' column

LESSONS LEARNED

CHANLLENGES

- vendors' ratings that cannot be obtained by averaging users' ratings

The vendors' ratings don't match to the average of customers' rating

-> It's hard to update the vendors' rating when customer's rate the vendors

average of customers' ratings

vendor_id	vendor_rating
4	1.751085
13	2.125490
20	1.859794
23	2.111111
28	1.647351
...	...
849	4.173285
855	4.215962
856	4.234568
858	4.245989
907	4.460993

≠

vendor' own ratings

	id	vendor_rating
0	4	4.4
1	13	4.7
2	20	4.5
3	23	4.5
4	28	4.4
...	...	...
95	849	4.1
96	855	4.2
97	856	4.3
98	858	4.2
99	907	4.3

Then,
how can users express or record their preference?

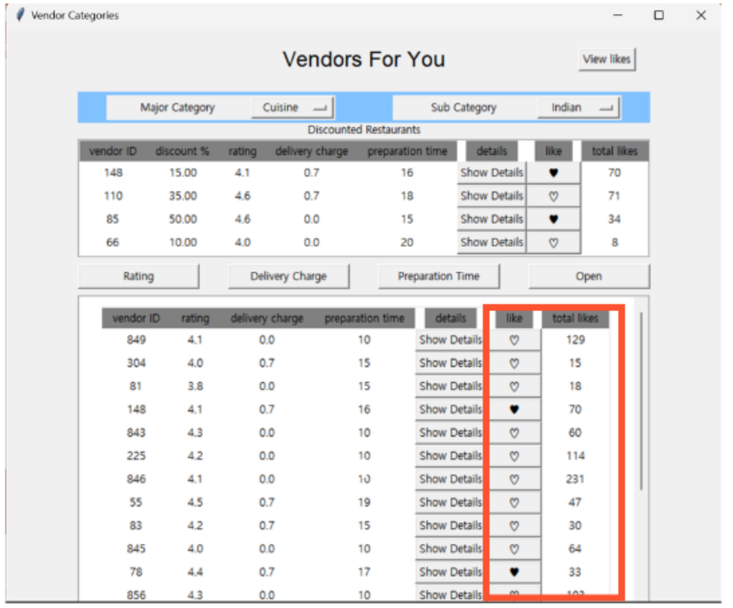
LESSONS LEARNED

SOLUTION

- decided to make other preference feature "like"

: If the user gives a rating of 5 points, it will be interpreted as "like"
and the vendor's sum of "like"(total like) can be shown to whole users

	customer_id	vendor_id	like
101	OH64IO0	310	1
159	FCPLE31	157	1
453	3P9113W	85	1
531	WB681BO	90	1
560	AA31G37	85	1
...	...	...	...
135238	Q68JJC7	845	1
135247	M825VSK	84	1
135268	RP97LI8	356	1
135283	3S6VG6R	199	1
135286	AVD95T0	106	1



LESSONS LEARNED

CHANLLENGES

- difficult to show only vendors available at the current time

It is hard to show only the restaurants that are currently open because the hours of operation are given as a single string

OpeningTime
11:00AM-11:30PM
08:30AM-10:30PM
08:00AM-10:45PM
10:59AM-10:30PM
11:00AM-11:45PM

LESSONS LEARNED

SOLUTIONS

- separate opening time and closing time

: If only the current time is entered, it becomes possible to create a query indicating whether or not the vendors are open.

open_time	end_time
11:00	23:30
08:30	22:30
08:00	22:45
10:59	22:30
11:00	23:45

LESSONS LEARNED

CHANLLENGES

- queries that depend on the existence of premary key values in the database

: Duplicate primary key values are not allowed, so if a user has a record of clicking "like" for a certain vendor, an 'UPDATE' query should be used,
and if there is no record, an 'INSERT' query should be used

```
# function for updating 'like' status
def update_like_main(values,user,btn):
    vendor_id = values[0]
        # if a vendor's status is 'like', the status is switched to 'normal'
    if btn.cget('text') == '♥':
        update_query = f"UPDATE likes SET `like` = 0 WHERE vendor_id = {vendor_id} && customer_id = '{user}'"
        myCursor.execute(update_query)
        mydb.commit()
        btn['text'] = '♡'
        # if a vendor's status is 'normal', the status is switched to 'like'
    else:
        try: # Use "UPDATE" if the customer have ever pressed "like" before
            like_dic[vendor_id]
            update_query = f"UPDATE likes SET `like` = 1 WHERE vendor_id = {vendor_id} && customer_id = '{user}'"
        except: # Use "INSERT" if the customer have never pressed "like" before
            update_query = f"INSERT INTO likes VALUES ('{user}',{vendor_id},1)"
            like_dic[vendor_id] = 1
        myCursor.execute(update_query)
        mydb.commit()
        btn['text'] = '♥'
```

LESSONS LEARNED

SOLUTIONS

- create a dictionary including the current "like" status of vendors that have clicked "like" at least once

–> If a vendor exists within this dictionary, use an UPDATE query, otherwise use an INSERT query.

```python
def get_likes(customer_id):
    mydb, myCursor = connectDB("project")
    query = f"SELECT * FROM likes WHERE customer_id='{customer_id}'"
    myCursor.execute(query)
    results = myCursor.fetchall()
    like_dic = dict()
        # make dictionary containing vendors' status 1(like)/0(normal)
        # the dictionary is used to get information if the customer hav
        # only if the ID exists as a key value is important
    for r in results:
        like_dic[r[1]] = r[2]
    return like_dic
```

```python
# function for updating 'like' status
def update_like_main(values,user,btn):
    vendor_id = values[0]
        # if a vendor's status is 'like', the status is switched to 'normal'
    if btn.cget('text') == '♥':
        update_query = f"UPDATE likes SET `like` = 0 WHERE vendor_id = {vendor_id} && customer_id = '{user}'"
        myCursor.execute(update_query)
        mydb.commit()
        btn['text'] = '♡'
        # if a vendor's status is 'normal', the status is switched to 'like'
    else:
        try: # Use "UPDATE" if the customer have ever pressed "like" before
            like_dic[vendor_id]
            update_query = f"UPDATE likes SET `like` = 1 WHERE vendor_id = {vendor_id} && customer_id = '{user}'"
        except: # Use "INSERT" if the customer have never pressed "like" before
            update_query = f"INSERT INTO likes VALUES ('{user}',{vendor_id},1)"
            like_dic[vendor_id] = 1
        myCursor.execute(update_query)
        mydb.commit()
        btn['text'] = '♥'
```

LESSONS LEARNED

CHANLLENGES

- unspecified values on 'location type' column
  - Most of the values were 'Other' or null.
  - Uncomfortable for users because they do not know which place the number refers to

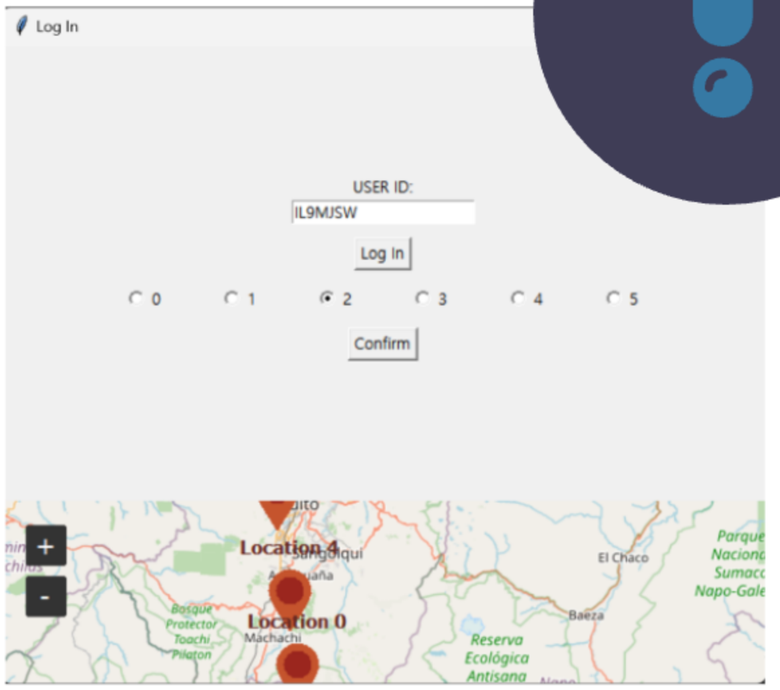
JD179G0	0	Work	-1.25606	-0.0904
UX1LKED	0		-1.74433	-0.34983
MRF3NLR	0	Home	0.07537	0.00057
FQ6BD1J	1		0.04351	-0.0085
8S4TR8P	1		-0.30303	0.6885944994204449
8S4TR8P	2	Other	-0.25725	0.5351635424340915
8S4TR8P	3	Other	-0.40665	0.6331922816876427
WEW8WR	1	Home	-0.89672	0.1460546160286092
QPS70WF	1		-0.19681	0.53584
YUET89W	0	Home	-1.09444	0.1012502956119417
R8SXYD0	0	Other	0.12783	0.5435047727272645
2KH6OSP	1		-1.86602	-0.00398
2KH6OSP	2		-0.03245	0.6199950927772784
XGE7RDS	1		-0.05805	-0.00846
XGE7RDS	2		-0.0354	0.02076
MWCJAZ1	0	Home	-0.09365	-0.00768

LESSONS LEARNED

SOLUTION

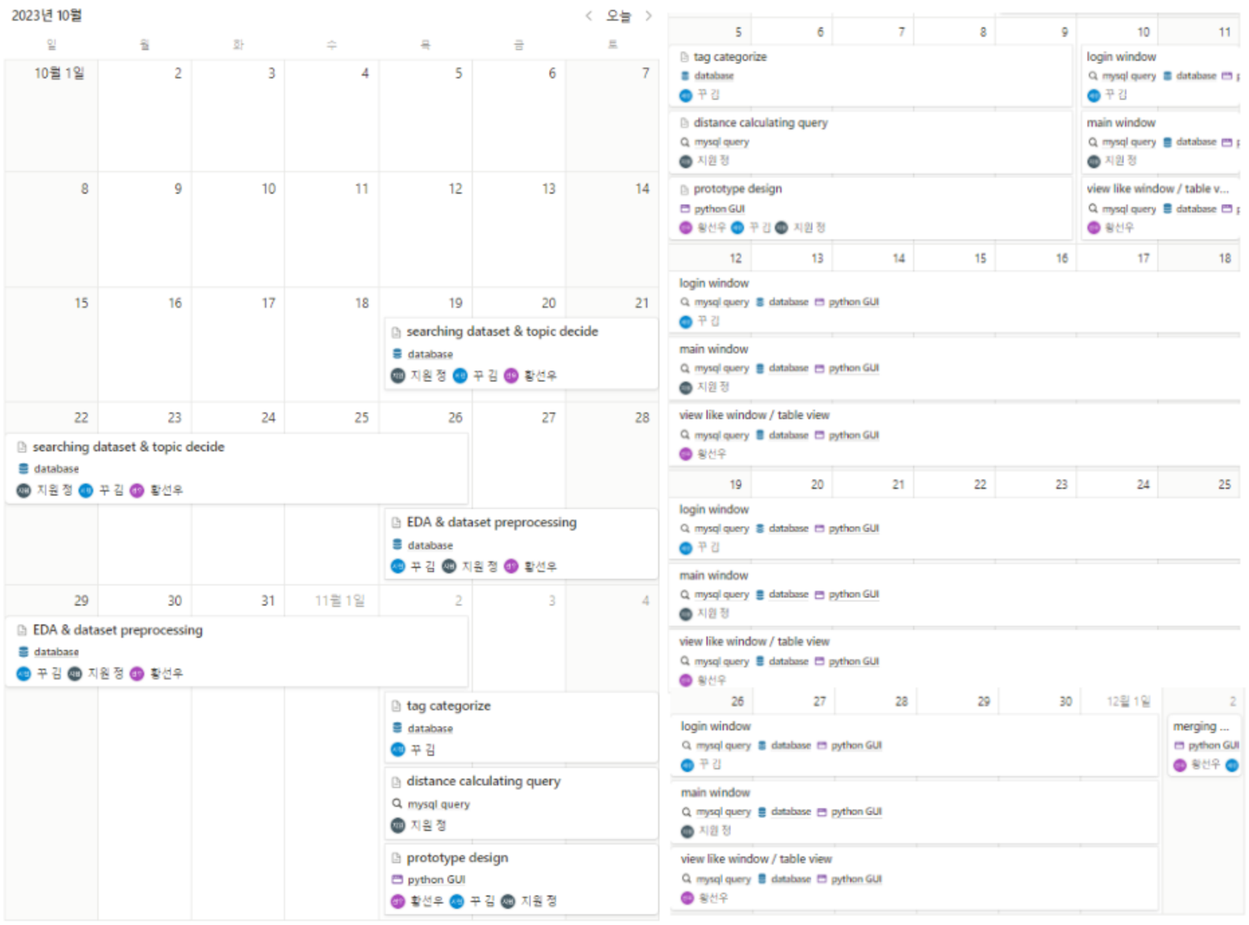
Using the given longitude and latitude, we deicided to show them on the map, so that the user can distinguish the location numbers in the visual way.

JD179G0	0	Work	-1.25606	-0.0904
UX1LKED	0		-1.74433	-0.34983
MRF3NLR	0	Home	0.07537	0.00057
FQ6BD1J	1		0.04351	-0.0085
8S4TR8P	1		-0.30303	0.6885944994204449
8S4TR8P	2	Other	-0.25725	0.5351635424340915
8S4TR8P	3	Other	-0.40665	0.6331922816876427
WEW8WR	1	Home	-0.89672	0.1460546160286092
QPS70WF	1		-0.19681	0.53584
YUET89W	0	Home	-1.09444	0.1012502956119417
R8SXYD0	0	Other	0.12783	0.5435047727272645
2KH6OSP	1		-1.86602	-0.00398
2KH6OSP	2		-0.03245	0.6199950927772784
XGE7RDS	1		-0.05805	-0.00846
XGE7RDS	2		-0.0354	0.02076
MWCJAZ1	0	Home	-0.09365	-0.00768



LESSONS LEARNED

TIMELINE



- week 8–9: searching dataset & topic decide
- week 9–10 : EDA & dataset preprocessing
- week 10–11: tag categorize, distance algorithm, prototype
- week 11–14: importing, GUI
- week 14–15: merging code