

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

РЕКОМЕНДУВАННЯ ЗАКЛАДІВ ДЛЯ ЛЮБИТЕЛІВ КАВИ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS

FOR RECOMMENDING PLACES TO COFFEE LOVERS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-142м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ШМАЛЬКО Ф.А.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

2023_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШМАЛЬКА Федора Анатолійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
рекомендування закладів для любителів кави. Research and Software
Implementation of Methods for Recommending Places to Coffee Lovers
керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) _____ рекомендована література, технічне
завдання _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали
і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення
щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та
експериментальне дослідження програми. _____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість
слайдів, плакатів)

Слайди презентації _____

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)

_____ Федір ШМАЛЬКО
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
114 с., 47 рис., 12 табл., 3 дод., 14 джерел.

КАВА, ЗАКЛАД ДЛЯ ЛЮБИТЕЛІВ КАВИ, РЕКОМЕНДУВАННЯ,
КЛАСТЕРНИЙ АНАЛІЗ, СЕНТИМЕНТ-АНАЛІЗ.

Об'єкт роботи – процес пошуку і рекомендування закладів для любителів кави. Предмет роботи – методи рекомендування закладів для любителів кави. Мета роботи – розробити програмне забезпечення пошуку і рекомендування закладів для любителів кави для інформування користувачів про релевантні заклади.

Матеріали, методи та технічні засоби: мова Python для машинного навчання та програмування вебзастосунків разом з фреймворком Django, система керування MySQL для роботи з базою даних.

Результати. Розроблено програму для інформування любителів кави про заклади, де подається кава або напої з кави, про напої з кави, які подаються у цих закладах, використовуючи для цього пошук цих закладів і рекомендування закладів, подібних тому, який розглядає користувач. Розглядаються заклади будь-якого типу, включаючи стаціонарні або мобільні точки з продажу кави, в тому числі автомати.

Висновки. Наукова новизна полягає в методі рекомендування закладів для любителів кави, який розділяє заклади у певному місці на кластери, відносячи їх до подібних, використовуючи для цього виділення з рейтингів інформації за окремими критеріями, сентимент-аналіз і визначення оцінок на його основі за кожним з критеріїв у разі відсутності таких оцінок у рейтингу та виконуючи спектральний кластерний аналіз за визначеними оцінками.

Галузь використання. Інформування про заклади, які можуть бути цікавими любителям кави.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 114 pages, 47 figures, 12 tables, 3 appendixes, 14 sources.

COFFEE, PLACE FOR COFFEE LOVERS, RECOMMENDATION, CLUSTER ANALYSIS, SENTIMENT ANALYSIS.

The object of the work is the process of finding and recommending places to coffee lovers. The subject of the work is methods of recommending places to coffee lovers. The goal of the work is to develop software for finding and recommending places to coffee lovers to inform users about relevant places.

Materials, methods and technical means: Python language for machine learning and web application programming using the Django framework, MySQL management system for working with the database.

The results. Software has been developed to inform coffee lovers about places where coffee or coffee drinks are served, about the coffee drinks served in these places, using the search for these places and the recommendation of places similar to the one the user is looking at for this purpose. Places of any type are considered, including stationary or mobile points of sale of coffee, including coffee machines.

Conclusions. The scientific novelty of the work is in the method of recommending places to coffee lovers, which divides places around a certain location into clusters, classifying these places as similar, using the selection of information from the ratings by separate criteria, sentiment analysis and the evaluation of ratings based on it for each of the criteria, if such evaluations were not made by reviewer manually, and performing spectral cluster analysis based on the determined ratings.

Field of use. Information about places which may be of interest to coffee lovers.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Огляд програмного забезпечення рекомендування закладів для любителів кави	10
1.2 Порівняння аналогів	16
1.3 Висновки за розділом 1	18
2 Матеріали і методи.....	19
2.1 Задача рекомендування закладів для любителів кави.....	19
2.2 Методи кластерного аналізу	20
2.3 Метод рекомендування закладів для любителів кави.....	21
2.4 Висновки за розділом 2	24
3 Проектування програмного забезпечення	25
3.1 Вибір засобів розробки	25
3.2 Моделювання програмного забезпечення на основі функціональних вимог	26
3.3 Проектування структури бази даних.....	32
3.4 Висновки за розділом 3	38
4 Основні рішення щодо реалізації програмного забезпечення	39
4.1 Порядок роботи програмного забезпечення	39
4.2 Рішення з реалізації вебзастосунку	40
4.3 Висновки за розділом 4	54
5 Експлуатація, тестування та експериментальне дослідження програми	55
5.1 Опис програми.....	55
5.2 Експлуатація програми.....	55
5.3 Експериментальне дослідження	61
5.4 Висновки за розділом 5	73
Висновки	74

Перелік джерел посилання	75
Додаток А Технічне завдання	77
Додаток Б Текст програми	83
Додаток В Слайди презентації	107

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

URL – Uniform Resource Locator;

БД – база даних.

ВСТУП

Велика кількість людей споживає каву, при чому робить це достатньо регулярно. За аналітичними даними, які представлено у роботі [1], визначено:

- 74 % американців п'ють каву щодня, 49 % людей випивають від 3 до 5 чашок кави на день;
- 32 % людей купують каву в кав'ярні від 1 до 3 днів на тиждень;
- багато людей витрачає 20 доларів США або менше на місяць на каву, тож все це в підсумку підтверджує популярність кави, а також достатньо великі обсяги коштів, характерні для цього ринку [1].

Культура споживання кави в Україні поступово змінюється, покращується. Це призводить до важливості отримування релевантної інформації про заклади або загалом місця продажу кави, які можуть зацікавити людину.

Інформування любителів кави може відбуватися в межах організації пошуку, надання інформації про конкретний заклад, а може також і як рекомендації, що розширює в підсумку отримані результати. Програму, яка має враховувати ці два напрямки, потрібно розробити в цій роботі, забезпечивши відповідний метод підтримки рекомендації закладів для любителів кави.

Метою роботи є розробити програмне забезпечення пошуку і рекомендації закладів для любителів кави для інформування користувачів про релевантні заклади.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд програмного забезпечення рекомендації закладів для любителів кави

Рекомендування закладів для любителів кави побудовано на створенні відгуків про ці заклади. Одним з основних програмних засобів у цій галузі є Yelp (рис. 1.1) [2], який містить відгуки про різні заклади та справи, створені користувачами. Таких справ достатньо багато, вони включають ресторани, автосервіси, сервіси, пов'язані з обслуговуванням будинків та інші типи сервісів, послуг. Серед таких закладів (ресторанів) є і заклади, які продають каву. Вони не виокремлені першочергово, але якщо в рядку пошуку задати слово кава, то буде виведено за результатами пошуку саме заклади для продажу кави.

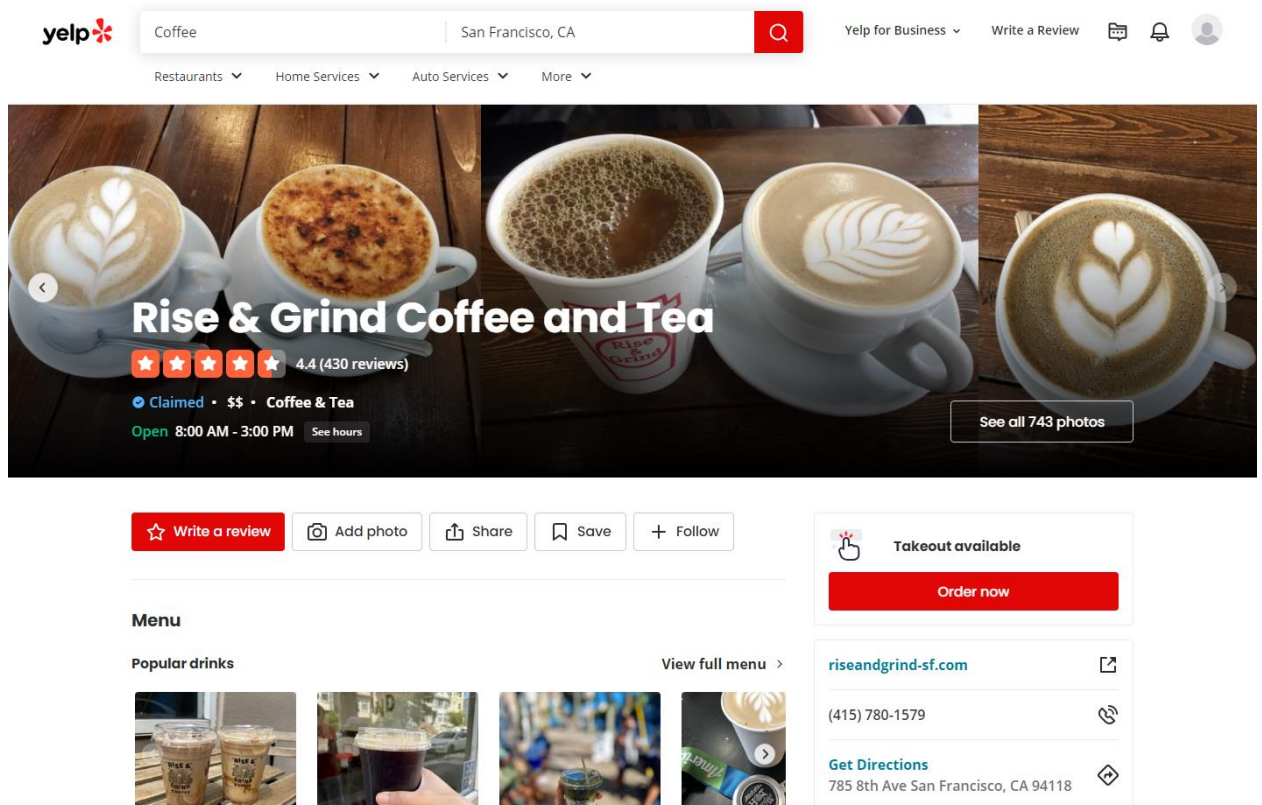


Рисунок 1.1 – Сторінка закладу в Yelp

За кожним закладом для продажу кави у Yelp надається інформація

про заклад, про години його роботи (розклад), меню, що включає відповідні напої (або страви, якщо це ресторан, а не заклад для продажу кави). За кожним напоєм надаються його фотографії, відгуки про напій. Окрім відгуків про окремі напої сторінка закладу містить також відгуки і про заклад загалом різних користувачів (рис. 1.2). Кожен відгук містить інформацію про автора, сам текст, оцінку за 5-бальною шкалою, можливо фотографії, а також набір реакцій на цей відгук. Кожна реакція містить кількість людей, які погодились з нею. Реакції включають позначку про корисність цього відгуку, подяку, любов до цього відгуку та негативне відношення.

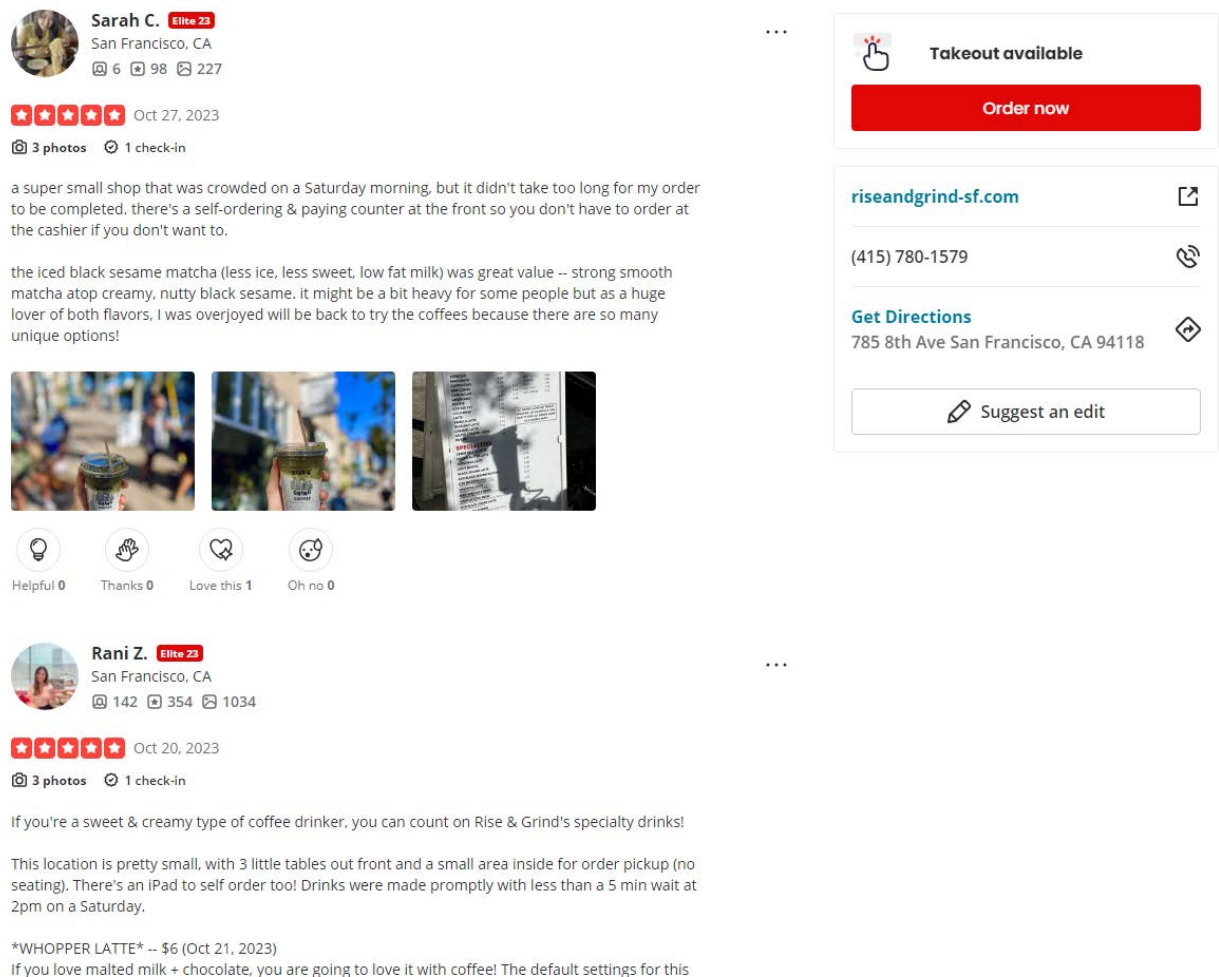


Рисунок 1.2 – Відгуки про заклад у Yelp

Програма не має системи автоматичної рекомендації за закладами, але вона містить рекламу закладів (рис. 1.3). Реклама надається за кожним

закладом, але містить тільки ті заклади, які сплатили за неї кошти. Тому вважати це повноцінною системою автоматичних рекомендацій не можна, адже вибір відбувається з обмеженого числа закладів, тому навіть звичним є те, що на заклад одного типу надаються поради про заклади взагалі іншого типу.

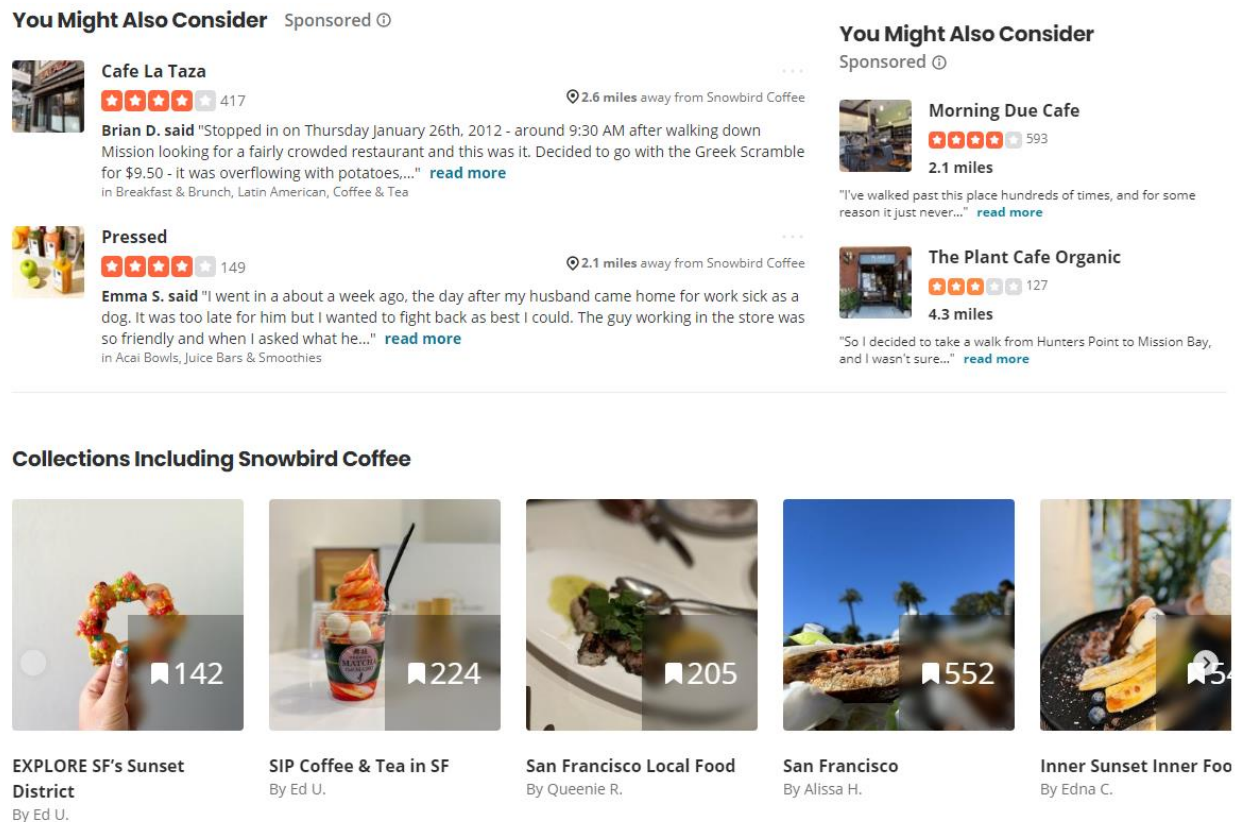


Рисунок 1.3 – Рекомендації закладів у Yelp

За результатами огляду визначено, що саме Yelp є прототипом програми, яка створюється в дипломній кваліфікаційній роботі. На користувачів програми можна підписуватися, можна надавати їм рекомендації (позначка compliment). За користувачами можна переглядати їхні відгуки. У програмі можна також виконати резервування місць у певних закладах, замовити страви, що вже не відноситься до відгуків, але розширює доступні користувачам можливості. Також реалізується можливість створити власний перелік закладів, зробити його публічним. Але переліки стосуються тільки закладів, а не страв чи напоїв.

FourSquare (рис. 1.4) [3] надає інформацію не тільки про заклади для любителів кави, а про різні заклади. Проте ніякої інформації стосовно напоїв, кави у програмі отримати неможливо. За кожним закладом надається інформація про його розташування, контактні дані, фотографії з закладу.

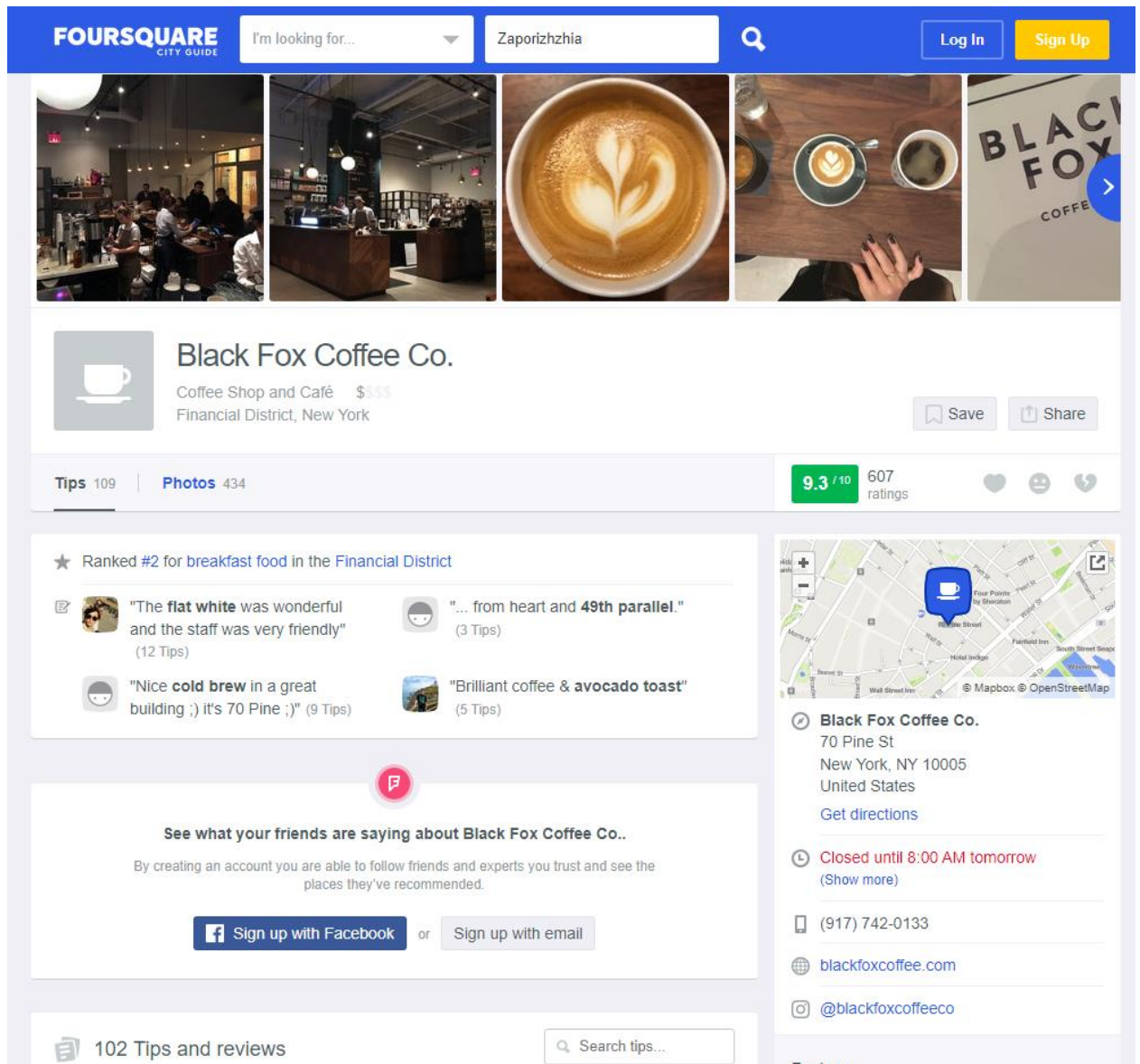


Рисунок 1.4 – Сторінка закладу в FourSquare

За кожним закладом з продажу кави приводяться рейтинги, виставлені користувачами, а також агрегується загальна оцінка. Рейтинги фактично є тільки відгуками, адже не передбачають виставлення оцінок за різними критеріями (рис. 1.5).

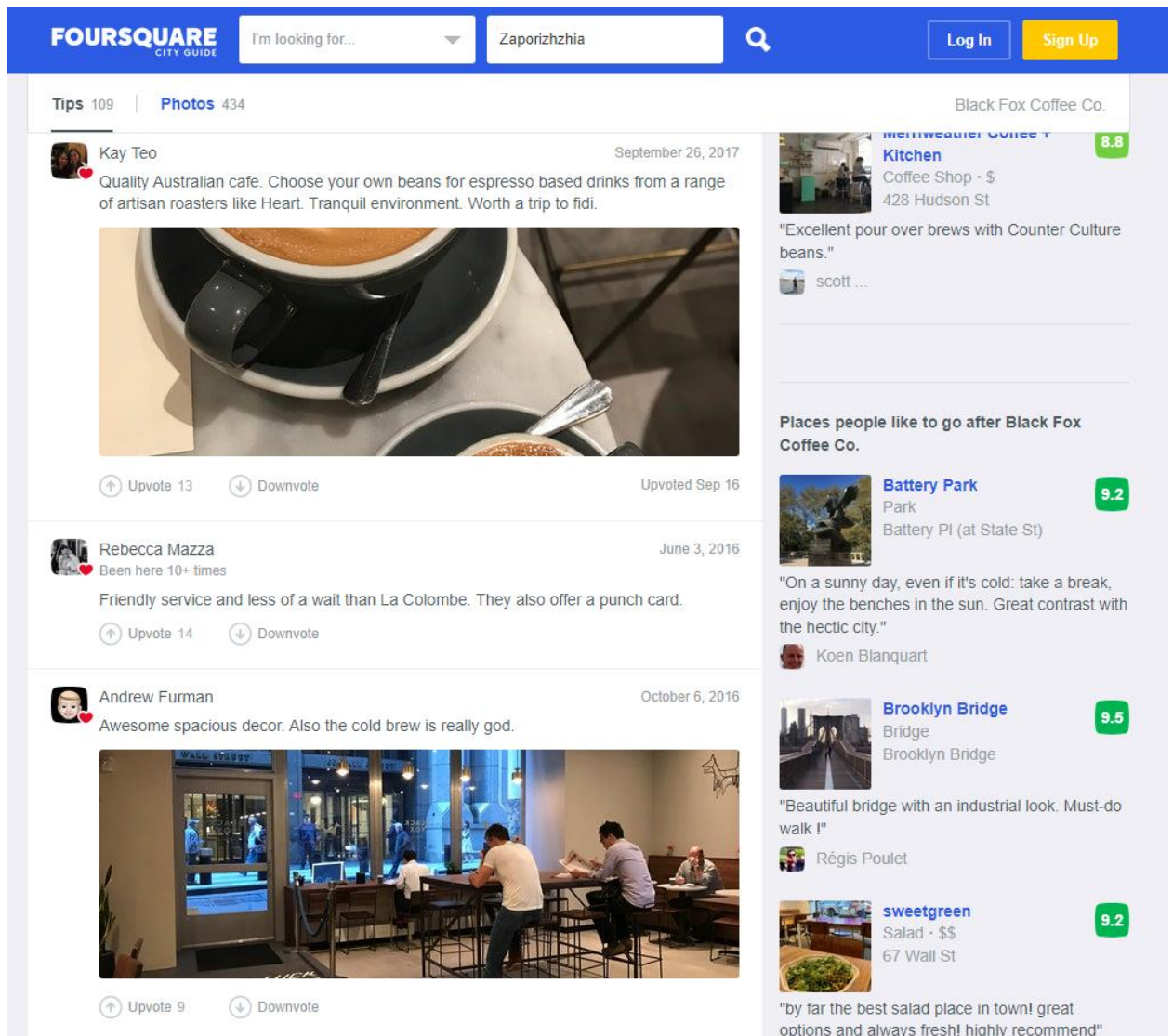


Рисунок 1.5 – Відгуки про заклад у FourSquare

У FourSquare надаються рекомендації до закладів з продажу кави, тобто можна зайти на сторінку закладу, а на ній побачити рекомендації інших закладів. У програмі також присутні можливості створення переліків закладів, підписування на сторінки інших користувачів, а також присутні можливість визначати реакції на створені іншими користувачами рейтинги (відгуки). Реакції включають позитивне голосування, негативне голосування, визначення рейтингу як спаму, нерелевантного або образливого.

Tripadvisor (рис. 1.6) [4] також дозволяє шукати заклади для любителів кави, але ці любителі кави фактично є мандрівниками. Тобто програма орієнтована на надання інформації про заклади різного типу, але в

розрізі для мандрівників. Сортування результатів пошуку, надання на сторінці закладу інформації про інші заклади виходить саме з рейтингу. Тобто рекомендаційна програма не має, але має рейтингову систему, за якою і будується весь доступ. Програма фактично рекомендує найкращі заклади, а не ті, які підходять конкретному користувачу.

Tripadvisor також не має інформації про напої з кави, меню або подібні засоби, надається інформація тільки про заклади з кави.

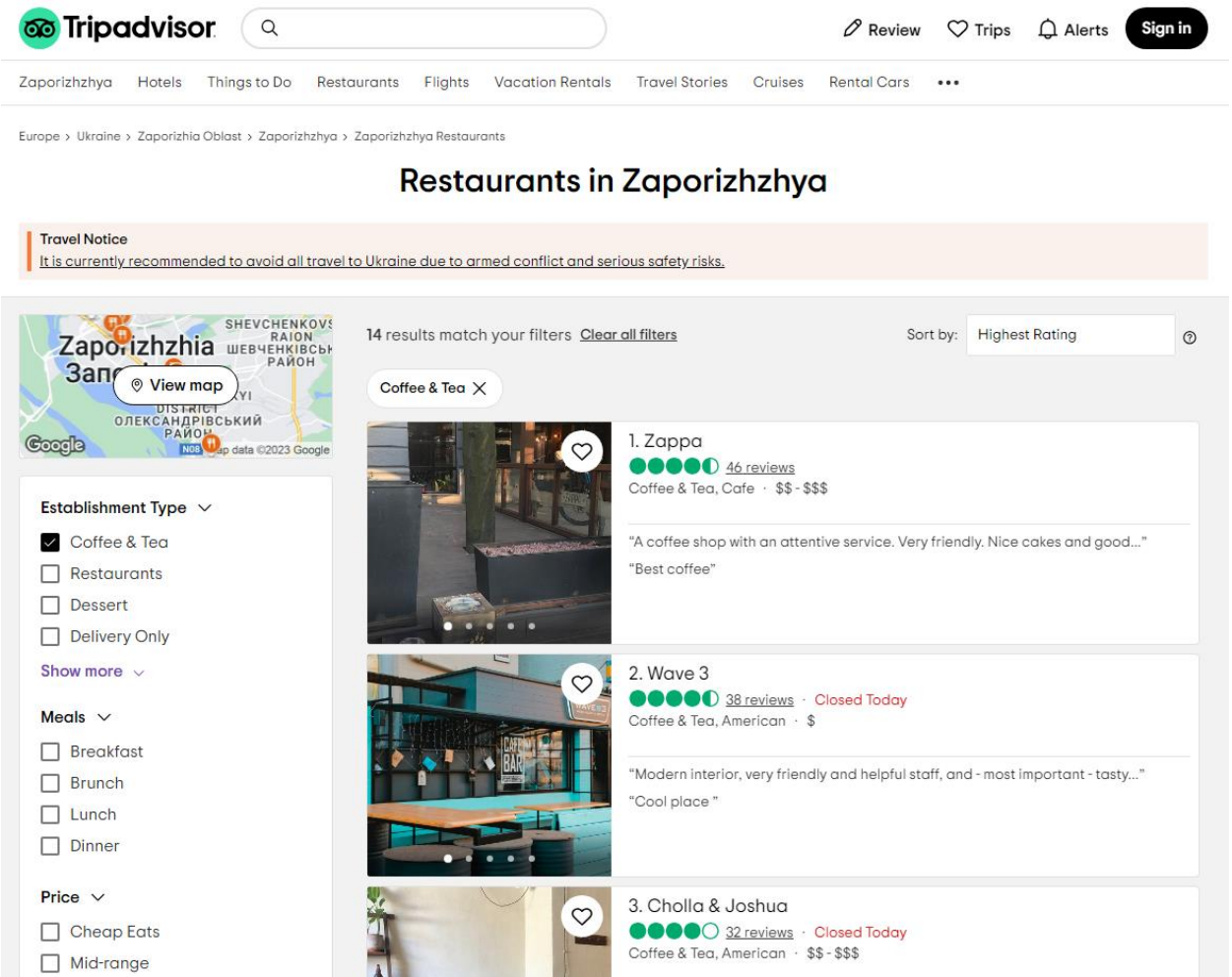


Рисунок 1.6 – Найкращі заклади у заданому місті в Tripadvisor

У програмі не можна створювати власні переліки закладів, а можна тільки надавати власні відгуки, за якими інші користувачі можуть визначати, що вони корисні, або інформувати про проблеми (рис. 1.7).

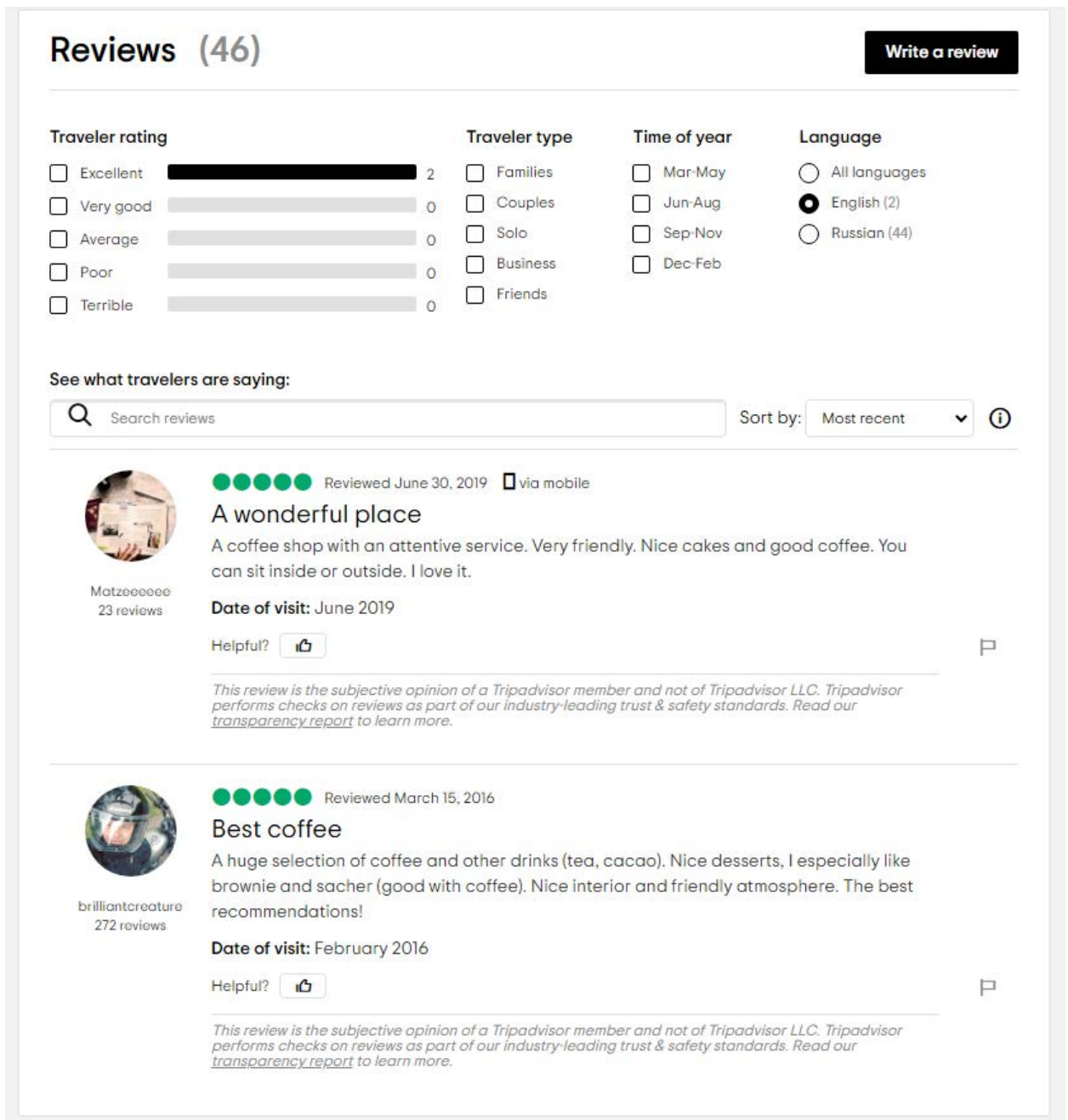


Рисунок 1.7 – Відгуки про заклад у Tripadvisor

1.2 Порівняння аналогів

Аналіз програм, які використовуються для пошуку закладів для любителів кави (табл. 1.1), включив всі програми, огляд яких було виконано, зокрема Yelp, FourSquare, Tripadvisor, та програму, яка представляється в роботі як результат.

Таблиця 1.1 – Аналіз програмного забезпечення, яке використовується для пошуку закладів для любителів кави

Критерій	Yelp	FourSquare	Tripadvisor	Представлена програма
Інформація про заклади з продажу кави	Так	Так	Так	Так
Інформація про інші місця, де продаються напої з кави	Ні	Ні	Ні	Так
Інформація про напої з кави	Так	Ні	Ні	Так
Пошук закладів	Так	Так	Так	Розширений (наприклад, за напоями)
Переліки закладів	Так	Так	Ні	Так
Переліки напоїв з кави	Ні	Ні	Ні	Так
Підписки на користувачів	Так	Так	Ні	Так
Рейтинги закладів	Так	Так	Так	Так
Рейтинги напоїв у закладах	Так	Ні	Ні	Так
Реакції на рейтинги	Так	Так	Тільки вподобання	Так, розширені
Рекомендування закладів з продажу кави	Тільки як реклама закладів загалом	Під заклади	Відсутні, тільки найкращі	Під заклади

Представлена в роботі програма відрізняється від розглянутих наявністю інформації про інші місця, де продається кава, а не тільки ресторани або кав'ярні, тобто в програмі надається інформація і про автомати з продажу кави, і про мобільні точки, будь-які кіоски та інші типи місць. Також у представленій програмі надається розширений пошук закладів, який включає можливість пошуку за датами, визначенням напоїв з кави, які мають подаватися в закладі. Окрім того тільки у Yelp надається інформація не тільки про заклади, але і про напої, на відміну від FourSquare і Tripadvisor, але у Yelp відсутні переліки напоїв з кави. Переліки напоїв з кави присутні в представленій програмі. Надається також можливість рекомендацій під окремі заклади, яка як така відсутня в Tripadvisor, а в Yelp надається фактично реклама закладів. У цей час у FourSquare відсутні багато можливостей з Yelp, які описані вище.

1.3 Висновки за розділом 1

Виконано огляд та проаналізовано програмне забезпечення, яке використовується для пошуку закладів для любителів кави, визначено те, чим має відрізнитися програма, яка має бути представлена в підсумку. У підсумку виконання розділу визначено завдання:

- проаналізувати програмне забезпечення, яке використовується для пошуку закладів для любителів кави;
- розробити метод рекомендування закладів для любителів кави і виконати експериментальне дослідження;
- виконати проєктування програми пошуку закладів для любителів кави;
- реалізувати програму пошуку закладів для любителів кави.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача рекомендації закладів для любителів кави

Задача рекомендації закладів для любителів кави може бути сформульована по-різному, адже рекомендації:

- можуть включати персональні характеристики користувача;
- можуть враховувати персональний досвід користувача стосовно попередніх закладів, створюючи таким чином послідовність отриманих результатів та прогнозуючи, що буде наступним;
- можуть базуватися на конкретному закладі і передбачати надання рекомендацій стосовно подібних закладів.

У роботі розглянуто останній з перелічених варіантів, враховуючи, що попередні 2 мають достатньо важливі недоліки, які полягають в необхідності накопичити інформацію про користувача. Тому рекомендації можна буде виконати тільки після того, як він встановить деякі оцінки іншим закладам. А у другому випадку таких даних потрібно буде ще й накопичити достатньо багато. Тож це виносить за дужки частину користувачів, які тільки зареєструвалися, а рекомендації будуть надаватися тільки після певного часу роботи.

Задача рекомендації закладів для любителів кави передбачає визначення множини доступних закладів, розбиття всієї множини на деякі підмножини закладів, а тоді в залежності від того, до якої з підмножин належить потрібний заклад, надання закладів з цієї підмножини як результату роботи.

Такі підмножини вирішено формувати на основі проведення кластерного аналізу, тож ця ідея має бути основою створеного у підсумку методу рекомендації закладів для любителів кави, який має розв'язати цю задачу в підсумку.

2.2 Методи кластерного аналізу

Одними з найбільш використовуваних методів кластеризації на даний момент є кластеризація k-середніх, де близько розташовані екземпляри відносяться до однакового кластеру, та спектральна кластеризація, де враховуються зв'язки між екземплярами замість відстані, тому не завжди найближче розташовані екземпляри опиняються в тому самому кластері [5]. Для кластеризації в межах рекомендації закладів для любителів кави кількість використовуваних критеріїв є достатньо великою, тому розглядаючи відстані в двомірному представленні важко однозначно ідентифікувати кластери, адже форма кластерів може бути достатньо складною, а фактично кажучи, є невідомою. Тому саме використання спектрального кластерного аналізу є підходящим для рекомендування закладів для любителів кави.

Спектральна кластеризація в основі використовує роботу з даними як з графом, тобто кожен екземпляр представляє собою певну вершину графа. У свою чергу граф можна представити як матрицю суміжності, де індекси рядків і стовпців представляють вершини, а конкретні записи у цій матриці представляють відсутність або наявність ребра між вузлами, тоді матриця спорідненості, яка використовується в даному варіанті кластеризації, подібна до матриці суміжності, за винятком того, що значення для пари точок виражає, наскільки ці точки схожі одна на одну, відповідно якщо точки в парі дуже різні, то спорідненість має дорівнювати 0, а якщо точки ідентичні, то спорідненість може дорівнювати 1 [6]-[7].

Перед тим, як виконувати кластерний аналіз, необхідно застосувати сентимент-аналіз для виділення емоцій, закладених у текст рейтингу, якщо числових оцінок у рейтингу не було встановлено. Для цього має застосовуватися одна з технік сентимент-аналізу. Ці техніки можуть бути побудовані на основі машинного навчання, однак тоді потребують розмічених даних, для яких визначено вручну таку оцінку. Оскільки це не

завжди можливо або не завжди може бути забезпечено, адже потребує додаткових витрат, то може бути використано більш просту техніку.

Сентимент-аналіз на основі лексики базується на визначенні емоцій у тексті рейтингу на основі слів, який текст рейтингу містить. У цьому методі для кожного слова, яким може визначатися емоційна забарвленість тексту, має бути визначено те, чи це слово є позитивним, негативним чи воно є нейтральним, а для кожної відповідності встановлено відповідне числове значення [8]. Загальна сентимент-оцінка тоді визначається, обчислюючи різницю між кількістю слів, які віднесені до позитивних і до негативних слів, поділена на їхню суму [9]. Маючи таку оцінку, загалом можна визначити, яку емоцію несе текст рейтингу (позитивну чи негативну), а також використати цю сентимент-оцінку для подальших необхідних обчислень.

2.3 Метод рекомендування закладів для любителів кави

Виконання методу рекомендування закладів для любителів кави розпочинається з визначення множини доступних закладів. До такої множини включаються всі заклади, які потенційно можуть бути релевантними. Це можуть бути заклади, які розташовані в межах одного й того самого міста або окремого його району, якщо це дуже велике місто. Це має бути визначено особами, які приймають рішення: в якому саме просторі можуть бути рекомендовані заклади. Окрім того до рекомендованих закладів можуть бути віднесені тільки заклади з загальним рейтингом не нижче заданого та з рейтингом за якістю кави окремо не нижче заданого рівня. Таким рівнем може бути 3 з 5. Виділення окремо якості кави визначається тим, що ці рекомендації виконуються саме для любителів кави, тому це є логічним.

Тоді для кожного закладу, що входить до множини доступних закладів, має бути визначено значення його характеристик, що включають:

- середній рейтинг за критерієм якості кави;

- середній рейтинг за критерієм атмосфери;
- середній рейтинг за критерієм зручності паркування;
- середній рейтинг за критерієм зручності місць у закладі;
- середній рейтинг за критерієм розташування закладу;
- середній рейтинг за критерієм обслуговування;
- середній рейтинг за критерієм цінової політики;
- середній рейтинг за критерієм рівня інтернету.

Якщо в самому рейтингу користувач такі значення встановив, то мають використовуватися саме такі значення. Якщо рейтинги, які пропонувались користувачу, не передбачали виставлення оцінок за цими критеріями, то має бути встановлено оцінки на основі тексту рейтингу (відгуку).

Для цього спочатку має бути проведено роботу з кожним рейтингом за кожним закладом і за кожним текстом рейтингу виконано оброблення тексту. Для цього мають бути спочатку відділені слова, а тоді виділені частини тексту, які стосуються кожного критерію. Для виконання цих дій спочатку за кожним критерієм має бути накопичено слова, які найчастіше його характеризують, тобто якщо мова йде про цінову політику, то відповідно це дорого, дешево, наприклад, та вартість, коштує, ціна тощо.

Тоді після проходження по кожному тексту відділяються частини з заданої максимальної кількості слів, в які входять терміни, що характеризують кожен критерій. Тоді за кожною такою порцією має бути проведено сентимент-аналіз та встановлено відповідну оцінку позитивності або негативності порції тексту відносно певного критерію.

Далі за сформованою оцінкою має бути сформовано рейтинг за критерієм. Для перетворення отриманого значення на рейтинг використовується наступне обчислення за формулою (2.1):

$$r_j^i = \begin{cases} \frac{(s_j^i - o^i)(sc^{\max} - r_{tot}^i)}{\max_i s_j^i - o^i} + r_{tot}^i, \text{ якщо } s_j^i \geq o^i; \\ r_{tot}^i - \frac{r_{tot}^i(o^i - s_j^i)}{o^i - \min_i s_j^i}, \text{ якщо } s_j^i < o^i, \end{cases} \quad (2.1)$$

де r_j^i – оцінка за критерієм з номером j текстового рейтингу з номером i ;

s_j^i – оцінка sentiment-аналізу за критерієм з номером j текстового рейтингу з номером i ;

o^i – оцінка sentiment-аналізу за загальним критерієм текстового рейтингу (тобто весь текст є позитивним або негативним висловленням) з номером i ;

$\max_i s_j^i$ – максимальна sentiment-оцінка, виставлена за критерієм з номером j серед всіх рейтингів;

$\min_i s_j^i$ – мінімальна sentiment-оцінка, виставлена за критерієм з номером j серед всіх рейтингів;

$sc^{\max}$ – максимальна можлива оцінка, тобто межа використовуваної системи оцінювання (у даному випадку, 5-бальна система, тобто 5);

r_{tot}^i – рейтинговий бал, який за цим рейтингом (відгуком) з номером i встановив сам користувач.

Після отримання таких оцінок за кожним критерієм за кожним рейтингом має бути обчислено середнє арифметичне за кожним критерієм для кожного закладу для любителів кави. Тоді ці отримані оцінки за кожним критерієм замінюють оцінки, виставлені власноруч користувачами.

На наступному етапі до всіх закладів, які пройшли попередній відбір та характеризуються середніми оцінками за кожним критерієм, застосовується спектральний кластерний аналіз з визначенням необхідної кількості кластерів. Результатом цього етапу є сформовані кластери серед всіх закладів. Тоді коли задається потрібний заклад (наприклад, це сторінка

відповідного закладу), то за ним рекомендуються заклади, які належать до того ж самого кластера.

2.4 Висновки за розділом 2

Наукова новизна роботи полягає в методі рекомендування закладів для любителів кави, який розділяє заклади у певному місці на кластери, відносячи їх до подібних, використовуючи для цього виділення з рейтингів інформації за окремими критеріями, сентимент-аналіз і визначення оцінок на його основі за кожним з критеріїв у разі відсутності таких оцінок у рейтингу та виконуючи спектральний кластерний аналіз за визначеними оцінками.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів розробки

При виборі засобів розробки потрібно одразу враховувати, що програмне забезпечення рекомендації закладів для любителів кави об'єднує в собі створення вебзастосунку в межах звужених строків, створення і використання моделей машинного навчання. Тому під час аналізу розглядалися саме фреймворки веброзробки (табл. 3.1) з подальшим додавання мови програмування та технологій, які з ними можна використати. Django [10]-[11] і Node.js [12] є одними з найчастіше застосованих фреймворків веброзробки, з яких обрано Django, адже він має більш потужний захист, дозволяє швидше завершувати розробку та підтримує мову python, для якої фреймворки машинного навчання на зараз є більш стабільними у роботі.

Таблиця 3.1 – Аналіз фреймворків веброзробки для створення програмного забезпечення рекомендації закладів для любителів кави

Критерій	Django	Node.js
Використовувана мова програмування	Python	JavaScript
Рівень безпеки даних і засобів	Високий рівень вбудованого захисту	Потребує додаткової роботи, ускладняється при збільшенні джерел даних
Час розробки співставних рішень	Менший	Вищий
Масштабованість	Висока	Висока
Стабільність фреймворків машинного навчання	Вища	Нижча

3.2 Моделювання програмного забезпечення на основі функціональних вимог

Першим типом користувачів для взаємодії з програмою є користувач без авторизації. Це користувач, який відкриває одну зі сторінок вебзастосунку, при цьому не виконавши авторизацію, якщо в нього є обліковий запис, або не маючи облікового запису.

Моделювання роботи користувача без авторизації з програмою полягало у включенні наступних сценаріїв (рис. 3.1):

- перегляд користувацьких рейтингів закладів для любителів кави;
- перегляд рейтингів за конкретним закладом для любителів кави;
- перегляд рекомендованих закладів для любителів кави за конкретним закладом;
- пошук закладів для любителів кави;
- пошук серед рейтингів закладів для любителів кави;
- пошук серед користувацьких рейтингів закладів для любителів кави;
- перегляд інформації про користувача;
- перегляд інформації про заклад для любителів кави;
- перегляд всіх фотографій закладу для любителів кави;
- перегляд фотографій кави з закладу для любителів кави за конкретним напоєм;
- перегляд користувацьких фотографій закладів для любителів кави;
- перегляд користувацьких фотографій кави з закладів для любителів кави;
- перегляд користувачів, які підписалися на користувача-оглядача;
- перегляд стрічки рейтингів від користувачів-оглядачів за підпискою;
- перегляд останніх рейтингів користувачів-оглядачів;
- перегляд напоїв за закладом для любителів кави;
- пошук користувачів-оглядачів;
- авторизація з альтернативним сценарієм реєстрації.

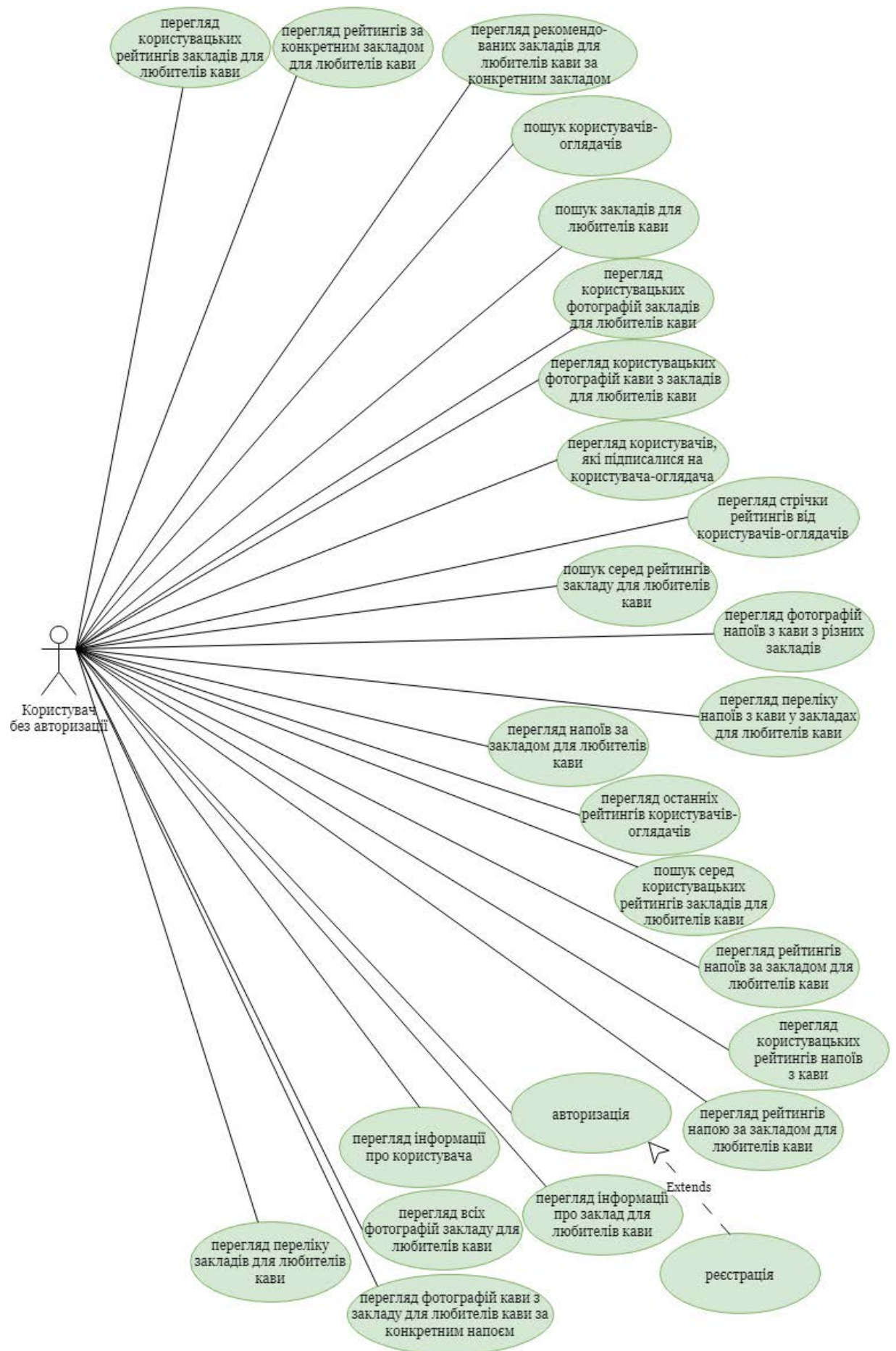


Рисунок 3.1 – Діаграма прецедентів для користувача без авторизації

Також сценарії користувача без авторизації:

- перегляд фотографій напоїв з кави з різних закладів;
- перегляд рейтингів напоїв за закладом для любителів кави;
- перегляд користувацьких рейтингів напоїв з кави;
- перегляд рейтингів напою за закладом для любителів кави;
- перегляд переліку закладів для любителів кави;
- перегляд переліку напоїв з кави з різних закладів для любителів кави.

Другим типом користувача, як і всі інші типи, є користувач після авторизації, але в даному випадку такий, що не має додаткових особливих прав доступу. Моделювання роботи стандартного користувача після авторизації з програмою полягало у включенні наступних сценаріїв (рис. 3.2):

- додавання закладу для любителів кави;
- створення користувацького переліку закладів для любителів кави, що має як альтернативний сценарій вилучення користувацького переліку закладів для любителів кави;
- надання публічного доступу до користувацького переліку закладів для любителів кави, що має як альтернативний сценарій закриття публічного доступу до користувацького переліку закладів для любителів кави;
- внесення закладу для любителів кави до користувацького переліку, що має як альтернативний сценарій вилучення закладу для любителів кави з користувацького переліку;
- визначення рейтингу закладу для любителів кави, що має як альтернативний сценарій коригування рейтингу закладів для любителів кави, що реалізується тільки тоді, коли рейтинг вже створений, але застосовується тільки в тому випадку, якщо з моменту створення рейтингу не пройшла доба (якщо пройшло більше доби, то коригувати рейтинг вже неможливо, але можна повторно визначити рейтинг, тобто фактично створити новий екземпляр рейтингу);
- повторне визначення рейтингу закладу для любителів кави.

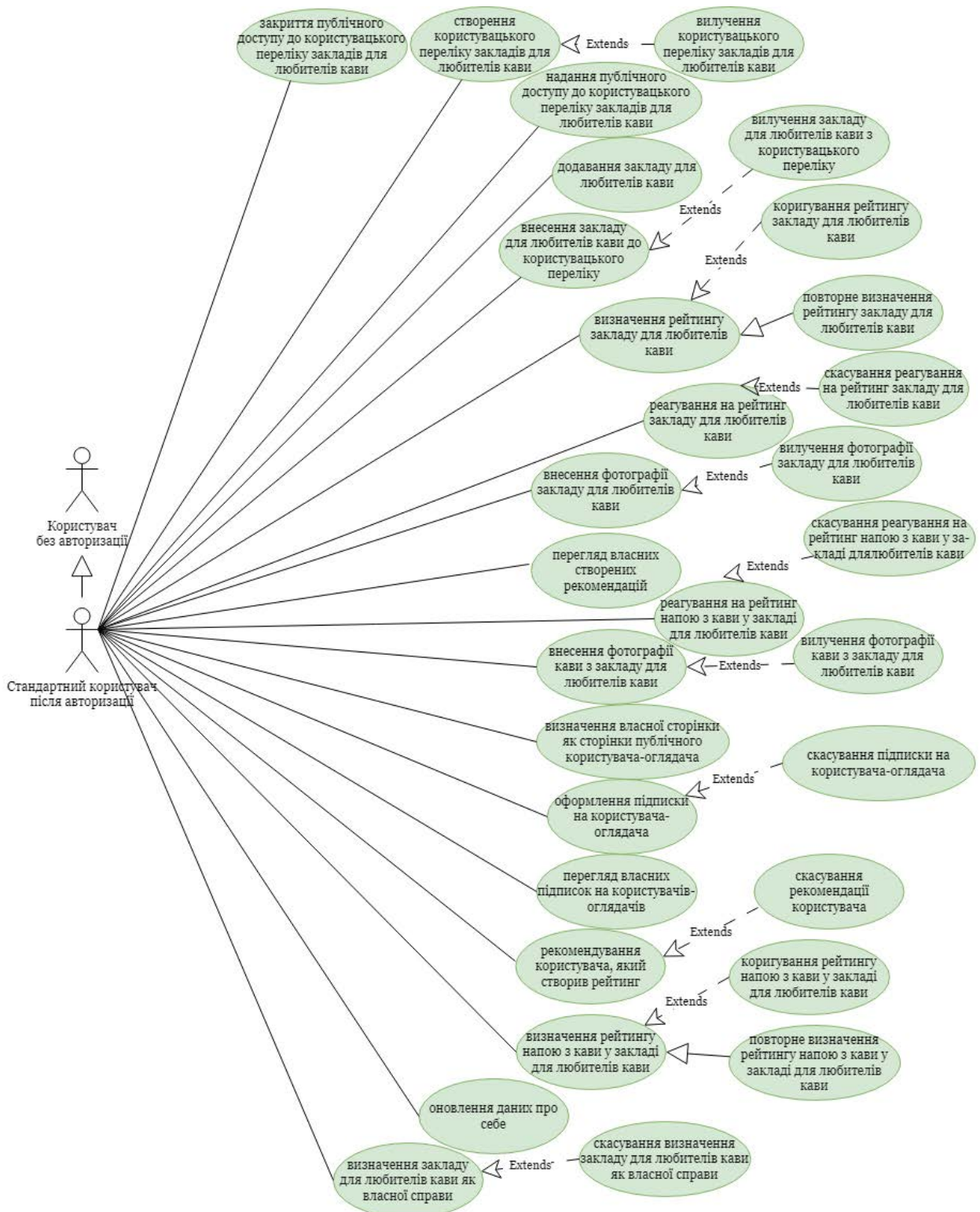


Рисунок 3.2 – Діаграма прецедентів для стандартного користувача після авторизації

Ще до сценаріїв роботи стандартного користувача після авторизації належать:

– реагування на рейтинг закладу для любителів кави, що має як

альтернативний сценарій скасування реагування на рейтинг закладу для любителів кави;

- внесення фотографії закладу для любителів кави, що має як альтернативний сценарій вилучення фотографії закладу для любителів кави;

- внесення фотографії кави з закладу для любителів кави, що має як альтернативний сценарій вилучення фотографії кави з закладу для любителів кави;

- визначення власної сторінки як сторінки публічного користувача-оглядача;

- оформлення підписки на користувача-оглядача, що має як альтернативний сценарій скасування підписки на користувача-оглядача;

- перегляд власних підписок на користувачів-оглядачів;

- рекомендування користувача, який створив рейтинг, що має як альтернативний сценарій скасування рекомендації користувача;

- перегляд власних створених рекомендацій;

- визначення закладу для любителів кави як власної справи, що має як альтернативний сценарій скасування визначення закладу для любителів кави як власної справи;

- оновлення даних про себе;

- визначення рейтингу напою з кави у закладі для любителів кави, альтернативним сценарієм для якого є коригування рейтингу напою з кави у закладі для любителів кави, що реалізується тільки тоді, коли рейтинг вже створений, але застосовується тільки в тому випадку, якщо з моменту створення рейтингу не пройшла доба (якщо пройшло більше доби, то коригувати рейтинг вже неможливо, але можна повторно визначити рейтинг, тобто фактично створити новий екземпляр рейтингу);

- повторне визначення рейтингу напою з кави у закладі для любителів кави;

- реагування на рейтинг напою з кави у закладі для любителів кави, альтернативним сценарієм для якого є скасування реагування на рейтинг

напою з кави у закладі для любителів кави;

- створення користувацького переліку напоїв з кави у закладах для любителів кави, що має як альтернативний сценарій вилучення користувацького переліку напоїв з кави у закладах для любителів кави;

- внесення напою до користувацького переліку напоїв з кави у закладах для любителів кави, що має як альтернативний сценарій вилучення напою з користувацького переліку напоїв з кави у закладах для любителів кави;

- перегляд самостійно створених переліків закладів для любителів кави;

- перегляд самостійно створених переліків напоїв з кави у закладах для любителів кави.

Третім типом користувача є власник справи. Це користувач, який авторизувався та визначив один з закладів для любителів кави як власну справу або який зареєструвався як власник справи. Моделювання роботи власника справи з програмою полягало у включенні наступних сценаріїв (рис. 3.3):

- перегляд власних бізнесів користувача;

- додавання напою з кави до закладу для любителів кави, що має як альтернативні сценарії вилучення напою з кави з закладу для любителів кави та оновлення даних напою з кави з закладу для любителів кави;

- підтвердження даних про напій з закладу для любителів кави, що має як альтернативний сценарій скасування даних про напій з закладу для любителів кави;

- оновлення даних про заклад для любителів кави.

Четвертим типом користувачів є модератор (рис. 3.4), який є авторизованим користувачем, але з додатковими правами, пов'язаними з керуванням процесами, тобто з можливостями оновлення даних вже не стосовно тільки власного закладу, як у випадку власника справи, а також з додатковими можливостями, які виділені у сценарій підтвердження закладу

для любителів кави як власної справи користувача, що має як альтернативний сценарій скасування закладу для любителів кави як власної справи користувача.

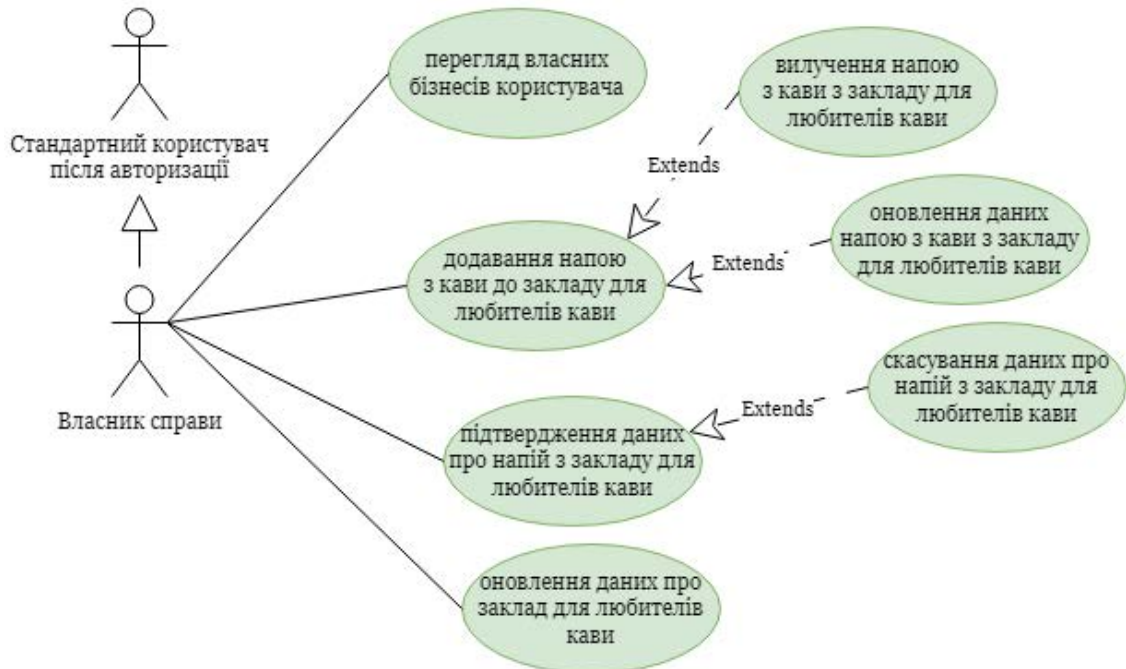


Рисунок 3.3 – Діаграма прецедентів для власника справи

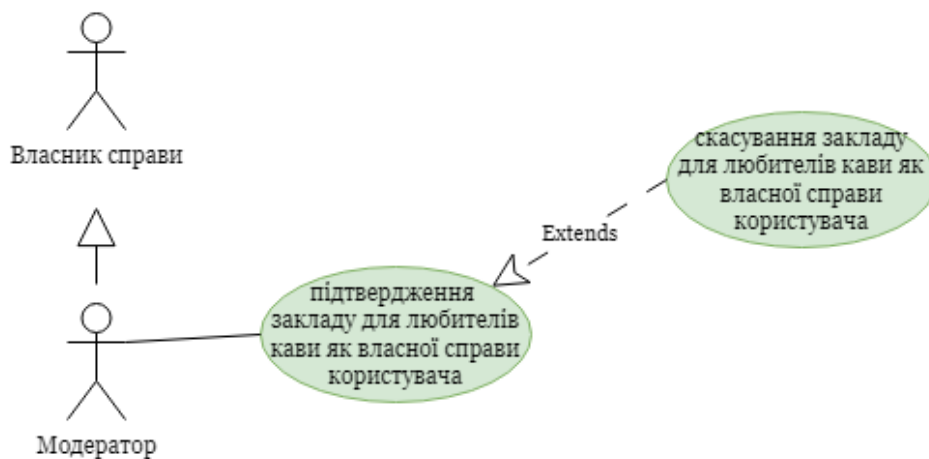


Рисунок 3.4 – Діаграма прецедентів для модератора

3.3 Проєктування структури бази даних

Не зважаючи на вибір Django та python як основних технологій розробки, існують можливості щодо вибору різних систем управління бази

даних (БД) для підтримки роботи з даними. При виконанні вибору обрано систему MySQL, адже вона є безкоштовною, широко розповсюдженою, що дозволяє забезпечити підтримку цієї системи управління БД фактично на кожному хостингу та не підвищує вимог до хостингу, окрім того ще й має високу надійність.

Розглянемо основні сутності, які було виділено в БД.

Звичайно, що зважаючи на формулювання теми роботи, однією з основних сутностей є заклад, який продає напої з кави, каву. Оскільки, як було визначено раніше, в програмі має надаватися інформацію не тільки про стандартні формалізовано виділені заклади, а до них має бути долучено заклади для любителів кави різних типів (автомати, кіоски тощо), то заклад для любителів кави використано в якості узагальнюючої сутності. Такі заклади для любителів кави винесено в сутність *Coffeeplace*. Заклади для любителів кави можуть мати різні типи, серед яких можуть бути ресторани, спеціалізовані кав'ярні, магазини для продажу кави, які додатково ще й продають напої з кави, але також це можуть бути і автомати для продажу кави, і мобільні пункти, різні кіоски або іншого типу заклади для продажу кави. Те, які саме типи закладів для любителів кави використовуються, визначається сутністю *Coffeeplacetype*.

Другою основною сутністю, яка продиктована самою темою, є звичайно любителі кави. Ці любителі кави є фактично користувачами програми і визначаються сутністю *Coffeelover*. Такий любитель кави може бути або звичайним користувачем, або користувачем-оглядачем. У другому випадку на його сторінку можна підписуватися. Відповідно фактично всі його рейтинги можна зручним чином потім переглядати серед всіх, на кого підписався авторизований користувач.

Любителі кави можуть бути не просто користувачами, які можуть створювати рейтинги, списки, а вони можуть бути власниками або представниками закладів для любителів кави, надаючи про них додаткову верифіковану ними інформацію. Інформація про те, який користувач які

заклади для любителів кави як власник цієї справи або представник представляє, відповідає сутності Businessowner.

Розклад роботи закладу для любителів кави визначається фактично робочим часом протягом одного з типових днів. Цей денний розклад визначається сутністю Timetable. Це може бути розклад на понеділок, вівторок, середу, четвер, п'ятницю, суботу, неділю або святковий день.

Кожен заклад для любителів кави може мати різні фотографії, які можуть надаватися власником, а можуть додаватися звичайними користувачами, які відвідали заклад. Відповідна сутність має назву Coffeeplacephoto.

Окрім фотографій кожен заклад може характеризуватися напоями з кави, які в ньому подаються або загалом продаються. Ця інформація за конкретним закладом може бути і відсутня, якщо сам власник цієї справи не надав такі дані. Проте якщо це так, то звичайні користувачі можуть вносити дані про те, які саме напої продаються в цьому закладі. Для того щоб не вводити в оману некоректними назвами напоїв початково необхідний певний довідник напоїв. Ці напої визначаються сутністю Drink і не стосуються конкретного закладу, а є загальними варіантами напоїв, які можуть продаватися.

Проте напої з кави є важливими для користувачів не самі по собі, а в тих варіаціях, в яких вони подаються у цих закладах для любителів кави. Конкретні напої в конкретних закладах загалом визначаються сутністю Coffeeplacedrink. Оскільки інформація про те, який саме напій в якому саме закладі продається, може бути внесена не тільки власником цього закладу, але і сторонньою особою, а тому в певних випадках може бути некоректною, то це повинно окремо зазначатися, тому кожен напій у певному закладі повинен містити мітку про те, чи інформація про нього є підтвердженою. Окрім того кожен напій у конкретному закладі може характеризуватися сортом кави, з якого він виготовляється. Ця інформація може додатково надаватися власником або представниками цього закладу, може вводитися

іншими користувачами, але загалом звичайно може бути і відсутньою. Для любителів кави це може бути додатковою важливою інформацією. Тому виникає необхідність зберігати сорти кави, створюючи певний довідник для цього, подібно до довідника напоїв з кави. Тож сорти кави визначаються сутністю Coffeesort.

Також напої у закладах для любителів кави можуть бути представлені фотографіями, які також завантажують або власники чи представники закладу, або звичайні користувачі, тобто любителі кави. Ці фотографії відповідають сутності Coffeeplacedrinkphoto.

Рейтингування закладів для любителів кави виконується за допомогою створення рейтингів звичайними користувачами. Ці рейтинги містять не тільки фактично текст, який можна разом з основною оцінкою, вважати відгуком, але і виставлені бали за різними параметрами, які характеризують цей заклад. Рейтинги закладів для любителів кави визначаються сутністю Coffeeplacering.

Рейтинги є корисним джерелом інформації для інших користувачів, які вирішують, які заклади можуть їх привабити. Але важливо визначити, які з них є правдивими, корисними, які вони емоції викликають. Тобто важливою є реакція на рейтинг інших користувачів, що в підсумку дозволяє певним чином оцінити рейтинг та привернути до нього увагу або навпаки відвернути. Оскільки реакції можуть бути різних типів, а за час життєвого циклу програми може з'явитися необхідність змінити назви реакцій або набір реакцій, то необхідним є виділення можливих реакцій (або фактично типів реакцій) на рейтинги закладів для любителів кави у окрему сутність Reactiontype.

Однак окрім типів реакцій важливими є конкретні реакції на конкретні рейтинги, які було отримано від різних користувачів (фактично любителів кави). Саме ці реакції виділені у окрему сутність Coffeeplaceringreaction.

Окрім рейтингування самих закладів для любителів кави рейтингування може стосуватися і напоїв, адже загалом заклад може бути достатньо тривіальним, але певний напій є досить смачним. За наявності рейтингування на окремі напої любитель кави може отримати додаткову необхідну інформацію для нього.

Рейтингування напоїв з кави, які продаються в конкретному закладі, визначаються сутністю *Coffeeplacedrinkrating*. Ці рейтинги також можуть мати реакції від інших користувачів за тією самою логікою. Самці ці рейтинги забезпечуються сутністю *Coffeeplacedrinkratingreaction*, а типи цих реакцій відповідають типам реакцій, які використовуються також і для рейтингів закладів для любителів кави.

Окрім надання інформації про заклади для любителів кави, рейтингування цих закладів, напоїв, ще одним варіантом поширення інформації про заклади, особливо невідомі заклади любителям кави, є переліки закладів для любителів кави та переліки напоїв, які в цих закладах продаються. Таким чином користувачі-оглядачі можуть привертати увагу до окремих закладів, напоїв, які їм сподобались, а відповідно і до своїх сторінок у результатах, збільшуючи кількість підписників.

Переліки закладів для любителів кави відповідають сутності *Coffeeplacelist* та складаються з окремих закладів, які любитель кави вніс до свого переліку. Ці окремі заклади у переліку визначаються сутністю *Coffeeplacelistmember*.

Переліки напоїв у варіаціях конкретних закладів для любителів кави, які створюються любителями кави, визначаються сутністю *Coffeeplacedrinklist*. Ці окремі напої у закладах для любителів кави, що внесені в перелік, визначаються сутністю *Coffeeplacedrinklistmember*.

Якщо любитель кави визначив свій тип доступу як користувач-оглядач, то інші користувачі можуть підписуватися на нього та переглядати його рейтинги закладів для любителів кави та самих напоїв у цих закладах, а також переглядати створені ним переліки закладів та напоїв, якщо вони

визначені як публічні. Підписки любителів кави на сторінки користувачів-оглядачів відповідають сутності Coffeeloverfollow.

Відповідна схема БД представлена на рис. 3.5.

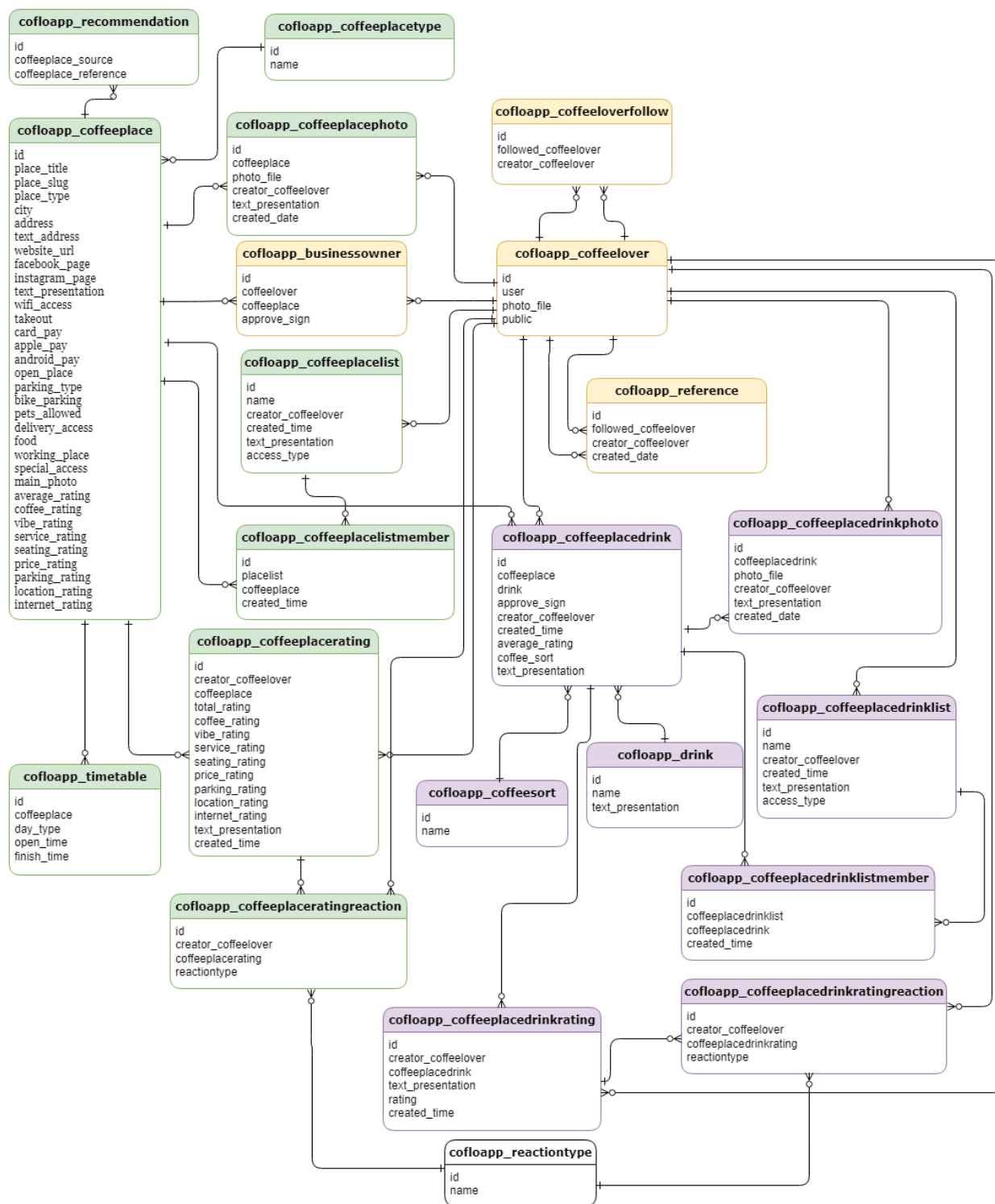


Рисунок 3.5 – Схема БД

На схемі на рис. 3.5 зеленим кольором виділено таблиці, які стосуються закладів для любителів кави, фіолетовим – напоїв з кави, а жовтим – користувачів.

Окрім стандартних реакцій на рейтинги можна також надати рекомендацію користувачу як вищу форми реакції, таким чином визначивши його здібності оцінювати заклади для любителів кави та напої. Рекомендації відповідають сутності Reference.

Рекомендації, які ж програмно формуються для закладів для любителів кави, зберігаються в таблиці `cofloapp_recommendation`.

3.4 Висновки за розділом 3

У розділі забезпечено моделювання сценаріїв для користувача без авторизації, стандартного користувача після авторизації, власника справи, модератора зі створенням діаграм прецедентів, виділено всі сутності предметної області і відповідні таблиці БД, що відносяться до закладів для любителів кави, напоїв з кави та користувачів. Реалізацію програми визначено виконувати мовою Python.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Порядок роботи програмного забезпечення

На рис. 4.1 показано схему пошуку закладів для любителів кави.

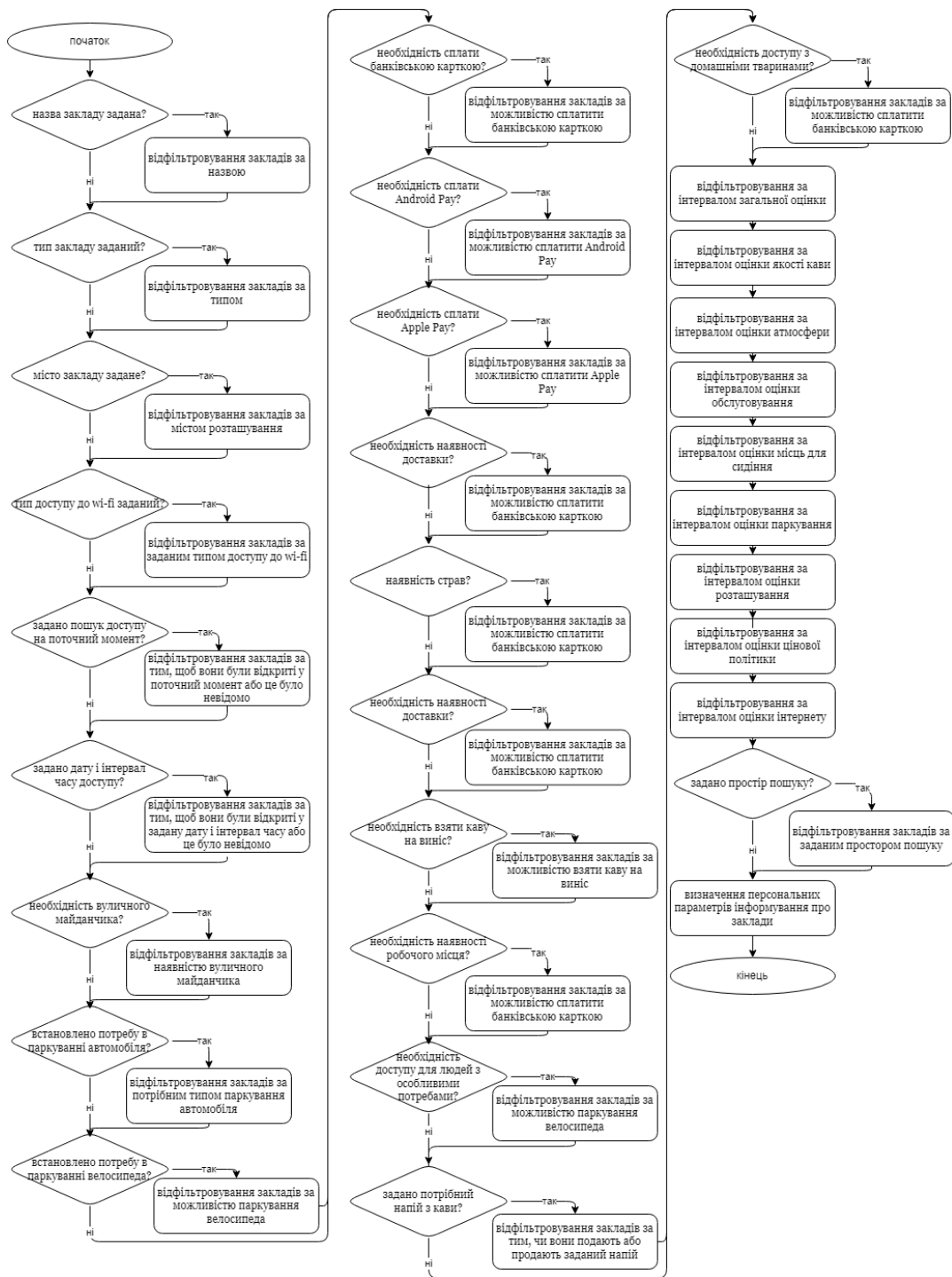


Рисунок 4.1 – Схема виконання пошуку закладів для любителів кави

4.2 Рішення з реалізації вебзастосунку

Застосунок Django, яким є програмне забезпечення, що розроблялось, мало необхідність визначення представлень застосунку та менеджера Uniform Resource Locator (URL) для взаємодії з користувачем щодо виконання функціональних вимог. Усі шаблони вебсторінок створені за шаблоном [13].

Реалізацію пошуку закладів для любителів кави виконано функцією `search_coffeeplace` на основі схеми з рис. 4.1 з визначенням значень всіх параметрів зі схеми через зчитування параметрів GET-запиту в файлі `views.py`. У цьому ж файлі аналогічно пошук серед рейтингів закладу для любителів кави створено функцією `search_coffeeplace_ratings`. Пошук серед рейтингів конкретного напою з кави, що подається у конкретному закладі, створено функцією `search_coffeeplace_drink_ratings`. Функція `coffee_lover_place_ratings` повертає набір рейтингів закладів для любителів кави, які були створені користувачем з логіном `coffeelover_username`. Функція `coffee_lover_drink_ratings` повертає набір рейтингів напоїв, що подаються у закладах для любителів кави, які були створені користувачем з логіном `coffeelover_username`. Функція `coffeeplace_ratings` повертає перелік всіх рейтингів, виставлених закладу для любителів кави з унікальним текстовим ідентифікатором `coffeeplace_slug`. Функція `drink_ratings` повертає перелік рейтингів, які стосуються закладу з унікальним текстовим ідентифікатором `coffeeplace_slug` та напою з назвою `drink_name`, яка якщо не вказана, то відповідає всім напоям, які в цьому закладі подаються. У всіх цих функціях окрім першої складовою результату, яким є словник, є дані за ключем `levels`, що є переліком, до якого входить кількість рейтингів з загальною оцінкою більше 4, більше до 3 і до 4, більше 2 і до 3 та менше 2.

Функція `coffee_place_photos` повертає перелік фотографій з закладу з текстовим ідентифікатором `coffeeplace_slug`, `coffee_place_drink_photos` – перелік фотографій за напоєм з назвою `drink_name` з закладу з текстовим

ідентифікатором `coffeeplace_slug`, `coffee_drink_photos` – перелік фотографій за напоєм з назвою `drink_name` з різних закладів, `coffee_lover_drink_photos` – перелік фотографій напоїв, зроблених користувачем з логіном `coffeelover_username`, `coffee_lover_place_photos` – перелік фотографій закладів, зроблених користувачем з логіном `coffeelover_username`.

Функція `coffee_lover_follows` повертає перелік користувачів, які підписалися на користувача з логіном `coffeelover_username`, а якщо звертається авторизований користувач, то додатково за ключем `to_follow` позначається, чи підписався на цього користувача цей авторизований користувач. Функція `new_ratings` повертає за ключами `place_ratings` і `drink_ratings` по 100 останніх створених рейтингів закладів і напоїв відповідно. Функція `place_drinks` повертає перелік усіх напоїв, які подаються у закладі з текстовим ідентифікатором `coffeeplace_slug`.

Функція `place_lists` повертає перелік всіх переліків закладів, які були створені авторизованим користувачем, а функція `drink_lists` – напоїв. Функція `coffee_lover_references` повертає перелік власних створених рекомендацій, наданих іншим користувачам. Функція `coffee_lover_businessowners` повертає перелік закладів, які визначені як справа авторизованого користувача.

Функція `new_follow_ratings` повертає перелік нових рейтингів, які були створені користувачами-оглядачами, на які підписався авторизований любитель кави. Функція `search_coffee_lover` підтримує пошук користувачів за логіном. Функція `coffee_lover_page` повертає дані для сторінки любителя кави з логіном `coffee_lover_username`, включаючи його 5 останніх рейтингів закладів, напоїв у закладах, рекомендації цього користувача, а також, якщо користувач є користувачем-оглядачем, то додаються його відкриті для доступу переліки закладів і напоїв у закладах.

Функція `place_list` повертає заклади з переліку з номером `placelist`, забороняючи доступ, якщо це закритий перелік, а авторизований користувач не є автором. Аналогічно для переліку напоїв у закладах працює функція `drink_list`.

Функція `coffee_place_page` повертає дані для сторінки закладу для любителів кави з текстовим ідентифікатором `coffee_place_slug`, включаючи до них 5 останніх рейтингів цього закладу і напоїв з закладу, 5 найкращих напоїв, 2 рекомендовані заклади, а також інформацію про сам заклад, його розклад роботи, фотографії.

Окремо винесені у файл `editorviews.py` представлення, які стосуються внесення даних у БД, зміни існуючих даних та формування рекомендацій закладів для любителів кави. Функція `coffee_place_add` реалізує додавання закладу для любителів кави, застосовуючи форму для реалізації, функція `coffee_place_list_create` реалізує створення користувацького переліку закладів для любителів кави, також застосовуючи в основі форму, а функція `coffee_place_list_add` додає до переліку з номером `place_list`, який попередньо створений, заклад з текстовим ідентифікатором `coffee_place_slug`. Інші функції додавання, редагування, видалення влаштовані подібним чином з виділенням функцій під сценарії.

Наступними створені моделі для застосунку Django.

Модель закладу для любителів кави реалізована класом `Coffeeplace` за допомогою визначення таких полів:

- `id` – унікальний номер закладу для любителів кави, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;
- `place_title` – назва закладу для любителів кави, має тип `CharField`;
- `place_slug` – унікальний текстовий ідентифікатор для сторінки закладу для любителів кави, має тип `SlugField`;
- `place_type` – тип закладу для любителів кави, є зовнішнім ключем з відповідної моделі, має тип `ForeignKey`;
- `city` – місто, де знаходиться заклад для любителів кави, має тип `CharField`, організований на основі варіантів вибору, якими є назви міст у програмі;

- address – координати адреси, де знаходиться заклад для любителів кави, має тип CharField, оскільки включає два числові значення, розділені комами;
- text_address – текстове представлення адреси, де знаходиться заклад для любителів кави, має тип CharField;
- website_url – URL вебсайту даного закладу для любителів кави, має тип URLField;
- facebook_page – URL сторінки в соціальній мережі Facebook закладу для любителів кави, має тип URLField;
- instagram_page – URL сторінки в соціальній мережі Instagram закладу для любителів кави, має тип URLField;
- text_presentation – опис текстом особливостей закладу для любителів кави, має тип TextField;
- wifi_access – наявність або відсутність доступу до Wi-Fi у закладі для любителів кави, має тип CharField, організований на основі варіантів вибору (відсутність, наявність без пароля, наявність з паролем, платно);
- takeout – наявність або відсутність можливості придбати каву на винос у даному закладі для любителів кави, має тип BooleanField;
- card_pay – наявність або відсутність можливості сплатити за замовлення банківською картою у даному закладі для любителів кави, має тип BooleanField;
- apple_pay – наявність або відсутність можливості сплатити за замовлення через сервіс Apple Pay у даному закладі для любителів кави, має тип BooleanField;
- android_pay – наявність або відсутність можливості сплатити за замовлення через сервіс Adroid Pay у даному закладі для любителів кави, має тип BooleanField;
- open_place – наявність або відсутність відкритого майданчика у закладі для любителів кави, має тип BooleanField;

- parking_type – тип паркування, наявний прямо поруч з закладом для любителів кави, має тип CharField, організований на основі варіантів вибору;
- bike_parking – наявність або відсутність паркування для велосипедів перед даним закладом для любителів кави, має тип BooleanField;
- pets_allowed – наявність або відсутність можливості відвідати заклад для любителів кави з домашньою твариною, має тип BooleanField;
- delivery_access – наявність або відсутність можливості здійснити доставку замовлення з закладу для любителів кави додому, має тип BooleanField;
- food – наявність або відсутність можливості придбати їжу, а не тільки напої з кави, у закладі для любителів кави, має тип BooleanField;
- working_place – наявність або відсутність робочого місця у закладі для любителів кави, має тип BooleanField;
- special_access – наявність або відсутність можливості доступу людей з обмеженими можливостями до закладу для любителів кави, має тип BooleanField;
- main_photo – основна фотографія закладу для любителів кави, має тип ImageField;
- average_rating – середній рейтинг, який було виставлено закладу для любителів кави користувачами, має тип FloatField;
- coffee_rating – середній рейтинг закладу для любителів кави в розрізі якості кави, має тип FloatField;
- vibe_rating – середній рейтинг закладу для любителів кави в розрізі атмосфери закладу, має тип FloatField;
- service_rating – середній рейтинг закладу для любителів кави в розрізі якості сервісу, має тип FloatField;
- seating_rating – середній рейтинг закладу для любителів кави в розрізі зручності місць для сидіння, має тип FloatField;
- price_rating – середній рейтинг закладу для любителів кави в розрізі цінової політики (вартості кави), має тип FloatField;

- parking_rating – середній рейтинг закладу для любителів кави в розрізі зручності паркування поруч, має тип FloatField;
- location_rating – середній рейтинг закладу для любителів кави в розрізі зручності розташування, має тип FloatField;
- internet_rating – середній рейтинг закладу для любителів кави в розрізі якості доступу до інтернету, має тип FloatField.

Модель типу закладів для любителів кави реалізована класом Coffeemplacetype за допомогою визначення таких полів:

- id – унікальний номер типу закладу для любителів кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;
- name – назва типу закладу, має тип CharField.

Модель розкладу роботи закладу для любителів кави реалізована класом Timetable за допомогою визначення таких полів:

- id – унікальний номер позиції розкладу роботи закладу для любителів кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;
- coffeeplace – заклад для любителів кави, якого стосується розклад роботи в такий день, є зовнішнім ключем з відповідної моделі, має тип ForeignKey;
- day_type – тип дня (або номер дня тижня за порядком, або спеціальний тип, що визначає святковий день), має тип SmallIntegerField;
- open_time – час відкриття для роботи закладу для любителів кави у зазначений день, має тип TimeField;
- finish_time – час завершення роботи закладу для любителів кави у зазначений день, має тип TimeField.

Модель фотографії закладу для любителів кави реалізована класом Coffeemplacephoto за допомогою визначення таких полів:

- id – унікальний номер фотографії закладу для любителів кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;

- `coffeerplace` – заклад для любителів кави, якого стосується ця фотографія, є зовнішнім ключем з відповідної моделі, має тип `ForeignKey`;
- `photo_file` – сама фотографія, яка представляє цей заклад для любителів кави, має тип `ImageField`;
- `creator_coffeeloover` – обліковий запис любителя кави, який додав цю фотографію, є зовнішнім ключем з моделі `Coffeeloover` та має тип `ForeignKey`;
- `text_presentation` – опис текстом повідомлення до цієї фотографії закладу для любителів кави, має тип `TextField`;
- `created_date` – дата, в яку було завантажено цю фотографію закладу для любителів кави, має тип `DateField`.

Модель рейтингування закладу для любителів кави реалізована класом `Coffeeplacerating` за допомогою визначення таких полів:

- `id` – унікальний номер рейтингування закладу для любителів кави, який створений автоматично як елемент моделі `Django`, має тип поля `BigAutoField`;
- `creator_coffeeloover` – обліковий запис любителя кави, який створив цей рейтинг, є зовнішнім ключем з моделі `Coffeeloover` та має тип `ForeignKey`;
- `coffeerplace` – заклад для любителів кави, для якого користувач створив цей рейтинг, є зовнішнім ключем з відповідної моделі, має тип `ForeignKey`;
- `total_rating` – сумарна оцінка закладу для любителів кави, має тип `PositiveSmallIntegerField`;
- `coffee_rating` – оцінка закладу для любителів кави в розрізі якості кави, має тип `PositiveSmallIntegerField`;
- `vibe_rating` – оцінка закладу для любителів кави в розрізі атмосфери закладу, має тип `PositiveSmallIntegerField`;
- `service_rating` – оцінка закладу для любителів кави в розрізі якості сервісу, має тип `PositiveSmallIntegerField`;
- `seating_rating` – оцінка закладу для любителів кави в розрізі зручності місць для сидіння, має тип `PositiveSmallIntegerField`;

- price_rating – оцінка закладу для любителів кави в розрізі вартості кави, має тип PositiveSmallIntegerField;
- parking_rating – оцінка закладу для любителів кави в розрізі зручності паркування поруч, має тип PositiveSmallIntegerField;
- location_rating – оцінка закладу для любителів кави в розрізі зручності розташування, має тип PositiveSmallIntegerField;
- internet_rating – оцінка закладу для любителів кави в розрізі якості доступу до інтернету, має тип PositiveSmallIntegerField;
- text_presentation – опис текстом враження від цього закладу для любителів кави, що може включати і деталізацію за переліченими критеріями та загалом, має тип TextField;
- created_time – час, в який було створено це рейтингування, має тип DateTimeField.

Модель типу реакції на рейтингування реалізована класом Reactiontype за допомогою визначення таких полів:

- id – унікальний номер типу реакції на рейтингування, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;
- name – назва типу реакції, має тип CharField.

Модель конкретної реакції на рейтингування закладу для любителів кави реалізована класом Coffeeplaceratingreaction за допомогою визначення таких полів:

- id – унікальний номер реакції на рейтингування закладу для любителів кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;
- creator_coffeelover – обліковий запис любителя кави, який створив цю реакцію, є зовнішнім ключем з моделі Coffeelover та має тип ForeignKey;
- coffeeplacerating – рейтинг закладу для любителів кави, на який визначено цю реакцію, є зовнішнім ключем з моделі Coffeeplacerating, має тип ForeignKey;

– `reactiontype` – тип реакції, який було визначено користувачем на це рейтингування, є зовнішнім ключем з моделі `Reactiontype`, має тип `ForeignKey`.

Модель переліку закладів для любителів кави реалізована класом `Coffeeplacelist` за допомогою визначення таких полів:

– `id` – унікальний номер переліку закладів для любителів кави, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;

– `name` – назва переліку, має тип `CharField`;

– `creator_coffeelover` – обліковий запис любителя кави, який створив цей перелік закладів для любителів кави, є зовнішнім ключем з моделі `Coffeelover` та має тип `ForeignKey`;

– `created_time` – час, в який було створено цей перелік, має тип `DateTimeField`;

– `text_presentation` – опис текстом переліку закладів для любителів кави, має тип `TextField`;

– `access_type` – тип доступу до переліку закладів для любителів кави (публічний або приватний), має тип `BooleanField`.

Модель запису про входження закладу для любителів кави до одного з переліків таких закладів реалізована класом `Coffeeplacelistmember` за допомогою визначення таких полів:

– `id` – унікальний номер запису про входження закладу для любителів кави до одного з переліків таких закладів, створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;

– `coffeeplacelist` – перелік закладів для любителів кави, до якого входить цей заклад, є зовнішнім ключем з моделі `Coffeeplacelist`, має тип `ForeignKey`;

– `coffeeplace` – заклад для любителів кави, який входить до даного переліку, є зовнішнім ключем з моделі `Coffeeplace`, має тип `ForeignKey`;

– `created_time` – час, в який було додано цей заклад для любителів кави до цього переліку, має тип `DateTimeField`.

Модель напою з кави реалізована класом Drink за допомогою визначення таких полів:

- id – унікальний номер напою з кави, створений автоматично як елемент моделі Django, має тип поля BigAutoField;
- name – назва напою з кави, має тип CharField;
- text_presentation – опис текстом напою з кави, має тип TextField.

Модель конкретного напою з кави, що подається в конкретному закладі для любителів кави, реалізована класом Coffeepacedrink за допомогою визначення таких полів:

- id – унікальний номер конкретного напою з кави, що подається в конкретному закладі для любителів кави, має тип поля BigAutoField;
- coffeepace – заклад для любителів кави, де подається цей напій, є зовнішнім ключем з моделі Coffeepace, має тип ForeignKey;
- drink – напій з кави, який подається в цьому закладі для любителів кави, є зовнішнім ключем з моделі Drink, має тип ForeignKey;
- approve_sign – підтвердження інформації про цей напій з кави, який подається в цьому закладі, має тип CharField, організований на основі варіантів вибору (невідомий, підтверджена істинність, підтверджена неправдивість);
- creator_coffeeloover – обліковий запис любителя кави, який вніс інформацію про цей напій з кави у цьому закладі, є зовнішнім ключем з моделі Coffeeloover та має тип ForeignKey;
- created_time – час, в який було внесено інформацію про те, що цей напій з кави подається в цьому закладі для любителів кави, має тип DateField;
- coffee_sort – сорт кави, з якої робиться цей напій, є зовнішнім ключем з моделі Coffeesort, має тип ForeignKey;
- text_presentation – опис текстом особливостей цього напою з кави в цьому закладі для любителів кави, має тип TextField;

– `average_rating` – середній рейтинг, який було виставлено даному напою з кави в цьому закладі для любителів кави користувачами, має тип `FloatField`.

Модель конкретної реакції на рейтингування напою з кави, що подається в певному закладі для любителів кави, реалізована класом `Coffeeplacedrinkratingreaction` за допомогою визначення таких полів:

– `id` – унікальний номер реакції на рейтингування напою з кави у варіанті певного закладу для любителів кави, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;

– `creator_coffeelover` – обліковий запис любителя кави, який створив цю реакцію, є зовнішнім ключем з моделі `Coffeelover` та має тип `ForeignKey`;

– `coffeeplacedrinkrating` – рейтинг напою з кави, на який визначено цю реакцію, є зовнішнім ключем з моделі `Coffeeplacedrinkrating`, має тип `ForeignKey`;

– `reactiontype` – тип реакції, який було визначено користувачем на це рейтингування, є зовнішнім ключем з моделі `Reactiontype`, має тип `ForeignKey`.

Модель сорту кави реалізована класом `Coffeesort` за допомогою визначення таких полів:

– `id` – унікальний номер сорту кави, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;

– `name` – назва сорту кави, має тип `CharField`.

Модель переліку напоїв з кави реалізована класом `Coffeeplacedrinklist` за допомогою визначення таких полів:

– `id` – унікальний номер переліку напоїв з кави, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;

– `name` – назва переліку, має тип `CharField`;

– `creator_coffeelover` – обліковий запис любителя кави, який створив цей перелік напоїв з кави, є зовнішнім ключем з моделі `Coffeelover` та має тип `ForeignKey`;

- `created_time` – час, в який було створено цей перелік, має тип `DateTimeField`;
- `text_presentation` – опис текстом переліку напоїв з кави, має тип `TextField`;
- `access_type` – тип доступу до переліку напоїв у закладах для любителів кави (публічний або приватний), має тип `BooleanField`.

Модель запису про входження напою з кави до одного з переліків таких напоїв реалізована класом `Coffeeplacedrinklistmember` за допомогою визначення таких полів:

- `id` – унікальний номер запису про входження напою з кави до одного з переліків таких напоїв, створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;
- `coffeeplacedrinklist` – перелік напоїв з кави, до якого входить цей напій, є зовнішнім ключем з моделі `Coffeeplacedrinklist`, має тип `ForeignKey`;
- `coffeeplacedrink` – напій з кави, який входить до даного переліку, є зовнішнім ключем з моделі `Coffeeplacedrink`, має тип `ForeignKey`;
- `created_time` – час, в який було додано цей напій з кави до цього переліку, має тип `DateTimeField`.

Модель рейтингування напою з кави реалізована класом `Coffeeplacedrinkrating` за допомогою визначення таких полів:

- `id` – унікальний номер рейтингування напою з кави, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;
- `creator_coffeelover` – обліковий запис любителя кави, який створив цей рейтинг, є зовнішнім ключем з моделі `Coffeelover` та має тип `ForeignKey`;
- `coffeeplacedrink` – напій з кави у закладі для любителів кави, рейтинг якого визначено, є зовнішнім ключем з моделі `Coffeeplacedrink`, має тип `ForeignKey`;
- `text_presentation` – опис текстом враження від цього напою з кави у цьому закладі для любителів кави, має тип `TextField`;

- rating – оцінка, виставлена напою з кави, має тип PositiveIntegerField;

- created_time – час, в який було створено рейтинг напою з кави, має тип DateTimeField.

Модель фотографії напою з кави у закладі для любителів кави реалізована класом Coffeepplacedrinkphoto за допомогою визначення таких полів:

- id – унікальний номер фотографії напою з кави у закладі для любителів кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;

- coffeepplacedrink – напій з кави, якого стосується ця фотографія, є зовнішнім ключем з моделі Coffeepplacedrink, має тип ForeignKey;

- photo_file – сама фотографія, яка представляє цей напій з кави, має тип ImageField;

- creator_coffeelover – обліковий запис любителя кави, який додав цю фотографію, є зовнішнім ключем з моделі Coffeelover та має тип ForeignKey;

- text_presentation – опис текстом повідомлення до цієї фотографії напою з кави у цьому закладі для любителів кави, має тип TextField;

- created_date – дата, в яку було завантажено цю фотографію напою з кави у цьому закладі для любителів кави, має тип DateField.

Модель облікового запису любителя кави реалізована класом Coffeelover за допомогою визначення таких полів:

- id – унікальний номер облікового запису любителя кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;

- user – обліковий запис користувача за стандартною моделлю User з Django, є зовнішнім ключем з відповідної стандартної моделі Django;

- public – обліковий запис відповідає сторінці користувача-оглядача або звичайного користувача, має тип BooleanField;

- photo_file – фотографія, яка представляє цього користувача, має тип ImageField;

– follows – перелік користувачів, які підписалися на цю сторінку користувача-оглядача, має тип ManyToManyField на основі моделі Coffeelover;

Модель визначення облікового запису користувача (любителя кави) як власника певної справи реалізована класом Businessowner за допомогою визначення таких полів:

– id – унікальний номер облікового запису користувача (любителя кави) як власника певної справи, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;

– coffeelover – обліковий запис любителя кави, який володіє або представляє цей заклад для любителів кави, є зовнішнім ключем з моделі Coffeelover та має тип ForeignKey;

– coffeeplace – заклад для любителів кави, який визначено як власну справу цього користувача, є зовнішнім ключем з моделі Coffeeplace, має тип ForeignKey;

– approve_sign – підтвердження інформації про те, що цей користувач представляє або володіє цим закладом для любителів кави як власною справою, має тип CharField, організований на основі варіантів вибору (невідомий, підтверджена істинність, підтверджена неправдивість).

Модель рекомендацій користувачів-оглядачів реалізована класом Reference за допомогою визначення таких полів:

– id – унікальний номер рекомендації користувача-оглядача кави, який створений автоматично як елемент моделі Django, має тип поля BigAutoField;

– followed_coffeelover – користувач-оглядач, рекомендацію щодо якого було створено цим любителем кави, є зовнішнім ключем з моделі Coffeelover;

– creator_coffeelover – обліковий запис любителя кави, який надав цю рекомендацію, є зовнішнім ключем з моделі Coffeelover та має тип ForeignKey;

- `created_date` – дата, в яку було надано цю рекомендацію користувачу-оглядачу, має тип `DateTimeField`.

Для реалізації запропонованого методу рекомендування закладів для любителів кави використано засоби пакетів `scikit-learn` та `nltk`. Сама кластеризація виконується класом `SpectralClustering`, до якого передаються методом `fit` дані, а потім у `labels_` отримуються мітки кластерів. Отримані результати зберігаються в моделі, клас якої має назву `Recommendation` та створений за допомогою визначення таких полів:

- `id` – унікальний номер запису, який створений автоматично як елемент моделі Django, має тип поля `BigAutoField`;
- `coffeeplace_source` – заклад для любителів кави, для якого визначено рекомендацію, є зовнішнім ключем з моделі `Coffeeplace`;
- `coffeeplace_reference` – заклад для любителів кави, який рекомендовано на цей заклад, є зовнішнім ключем з моделі `Coffeeplace`.

Функція представлення `coffee_place_recommend` повертає для закладу для любителів кави з текстовим ідентифікатором `coffee_place_slug` перелік закладів, які для нього вважаються подібними і рекомендованими.

4.3 Висновки за розділом 4

Представлено схему виконання пошуку закладів для любителів кави, яка забезпечує підтримку пошуку саме закладів, але також і логіку, яка подібно застосована для пошуку інших даних у відповідності зі сценаріями роботи. Усі сценарії роботи реалізовані та представлено те, яким чином це зроблено, у розділі.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

Основним призначенням програми є інформування любителів кави про заклади, де подається кава або напої з кави. Розглядаються заклади будь-якого типу, включаючи стаціонарні або мобільні точки з продажу кави, в тому числі автомати. Для забезпечення інформування складовими є пошук закладів для любителів кави та рекомендування закладів продажу кави користувачам, що дозволяє розширити пул закладів, які використовує кожна окрема людина, що є користувачем програми.

5.2 Експлуатація програми

Експлуатація програми через розміщення її на вебсервері дозволила перевірити заплановані сценарії роботи. Частина з них доступна через меню, а частина доступна через окремі сторінки.

Меню розміщено згори кожної сторінки та включає цілий ряд розділів та підрозділів, які можуть збільшуватись в залежності від ролі користувача. Розділи меню включають:

- розділ Заклади, з підрозділів для користувача без авторизації доступний тільки Пошук (тому це не розділ, а пункт), а для інших ролей сюди входять Додати для внесення даних закладу, Переліки для доступу до власних переліків закладів (тому для них це розділ, а не пункт);
- розділ Оглядачі, де є Шукати, Нові рейтинги (від користувачів-оглядачів), що доступні користувачам без авторизації, а для інших додатково: Стрічка для перегляду рейтингів від користувачів, на яких підписався користувач, Підписки з переліком таких користувачів, Рекомендації з переліком рекомендацій, які були надані цим користувачем іншим;

– розділ Напої з підрозділом Усі (фотографії напоїв з різних закладів – єдине, що доступно без авторизації) та Переліки напоїв;

– розділ Бізнеси, що доступний після авторизації та для власника справи надає доступ до його справ (закладів), а для модератора дозволяє перейти до пошуку користувачів, звідки можна одразу переглянути заклади користувача;

– розділ Персона, що дозволяє вносити зміни у обліковий запис.

Кнопка Доеднатися перенаправляє до авторизації.

Сторінка закладу (рис. 5.1) містить основну фотографію (інші фотографії нижче з кнопкою для переходу на окрему сторінку зі всіма фотографіями), тип закладу, назву, адресу, загальну оцінку, розклад роботи, кнопку для створення рейтингу (без авторизації буде переведено до авторизації спочатку), додавання до переліків (нижче фото, але тільки після авторизації, включаючи перелік та посилання додати або вилучити для додавання або вилучення з обраного переліку цього закладу).

CofloApp

ЗАКЛАДИ ОГЛЯДАЧІ НАПОЇ

Доеднатися →



Заклад

Великий ресторан

Coffee Stock

Запоріжжя вул. Гончаренка, 32

Оцінка: 5.0

Ми продаємо, подаємо і смакуємо каву разом з вами. У нас можна придбати різні сорти кави, спробувати наймачніші напої з цієї кави та просто провести класно час на великому вуличному майданчику та внутрішньому залі на 50 місць.

☑ Понеділок: 10:00 - 22:00

☑ Вівторок: 10:00 - 22:00

☑ Середа: 10:00 - 22:00

☑ Четвер: 10:00 - 22:00

☑ П'ятниця: 10:00 - 23:00

☑ Субота: 12:00 - 23:00

☑ Неділя: 14:00 - 23:00

☑ Святкові дні: 15:00 - 20:00

Створити рейтинг

Рисунок 5.1 – Сторінка закладу для любителів кави

Сторінка закладу для любителів кави містить також деталізацію оцінок за критеріями в середньому, 5 останніх рейтингів закладу (рис. 5.2) з кнопкою Показати всі для того, щоб відкрити сторінку зі всіма створеними рейтингами про заклад, 5 останніх рейтингів напоїв, що подаються у закладі, влаштовані аналогічно, 5 найкращих за рейтингами напоїв, що подаються у закладі з кнопкою Показати меню для відображення сторінки з переліком напоїв у меню.

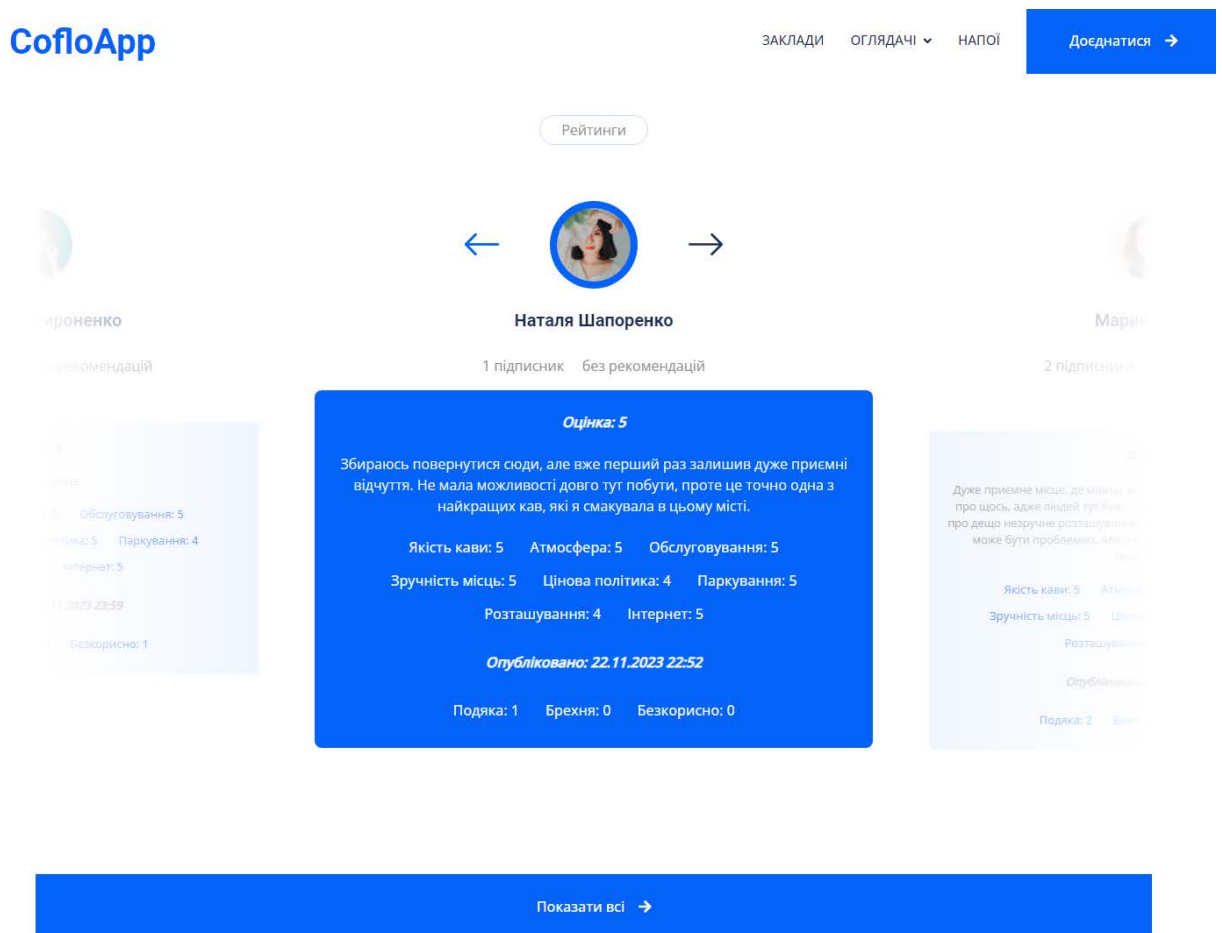


Рисунок 5.2 – Рейтинги закладу для любителів кави користувача без авторизації

У варіанті цієї сторінки після авторизації додається ряд описаних можливостей, а для рейтингів надаються позначки плюс та мінус для додавання або скасування підписки, рекомендації, реакцій на рейтинг (рис. 5.3).

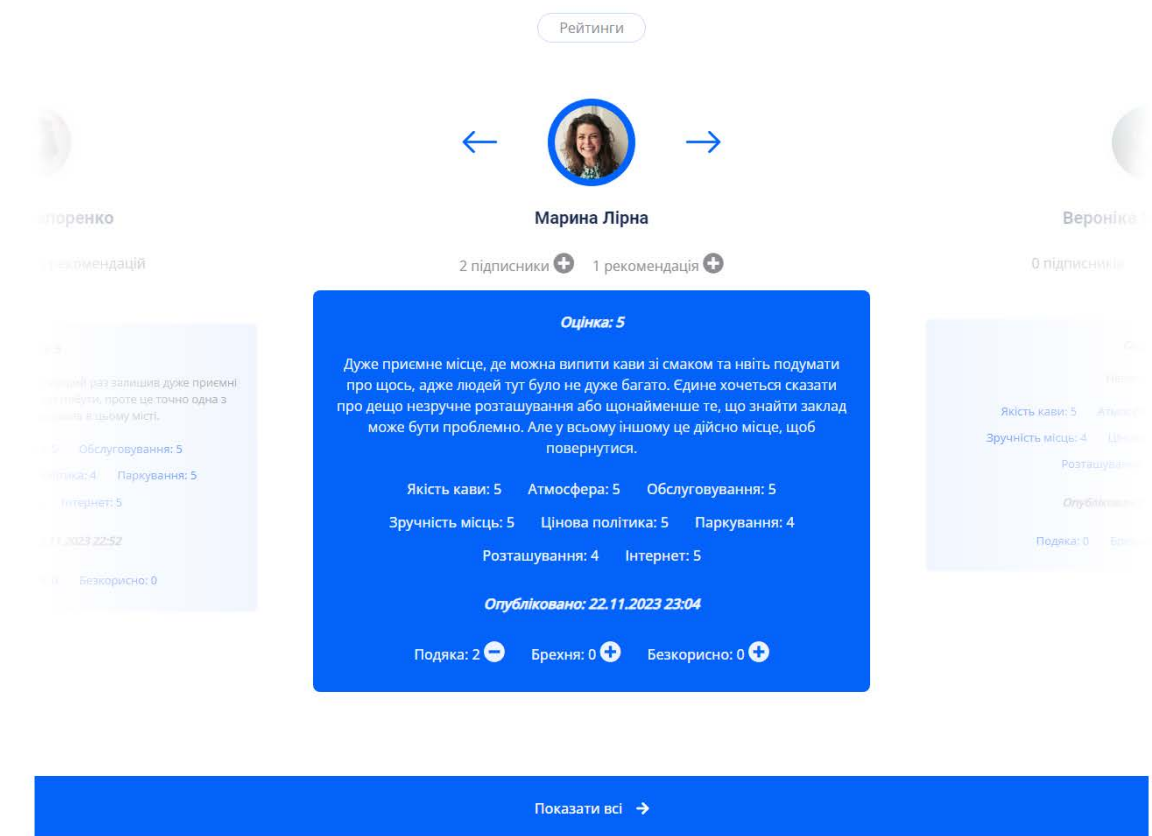


Рисунок 5.3 – Рейтинги закладу для любителів кави користувача після авторизації

Внизу на сторінці також є позначка Мій бізнес, щоб відзначити, що цей заклад керується певним користувачем. Якщо хтось це позначив, то далі модератор може підтвердити це або скасувати. Якщо підтверджено, то користувач зможе оновлювати дані закладу.

Зі сторінки фотографій доступна кнопка для додавання власної фотографії.

З напоїв можна додавати напій до власного переліку.

На сторінці безпосередньо під відгуками розташовані рекомендації подібних закладів, сформовані запропонованим методом (рис. 5.4). Це типовий спосіб приведення переліку закладів. Звідси можна перейти до сторінки закладу, а кнопкою Показати всі, що нижче, відкрити цілу таку сторінку.

Рекомендації



Вдомано

5.0 з 4 рейтингів



Італ'яно

4.83 з 6 рейтингів



Уява і смак

4.75 з 4 рейтингів



Для друзів

4.75 з 4 рейтингів

Рисунок 5.4 – Рекомендації до закладу для любителів кави

Зі сторінки переліку доступний перелік самих закладів (рис. 5.5). Якщо це користувач, який створив цей перелік, то йому надаються кнопки Зробити публічним, Редагувати для зміни назви, опису переліку, рівня доступу, Вилучити. За кожним закладом такий користувач при наведенні може побачити спливаюче посилання вилучити, яке дозволяє одразу вилучити заклад з переліку.

Якщо користувач виконає пошук закладів, то сторінка пошуку є формою з полями, які відповідають вказаній у роботі схемі, а результати показуються на окремій сторінці (рис. 5.6), звідки кнопкою Шукати знов можна повернутися до форми пошуку.

За результатами пошуку можна одразу навести курсор на один з закладів, тоді з'явиться перелік зі всіма створеними користувачем, який авторизувався, переліками, звідки можна обрати потрібний, а тоді з'явиться посилання додати для внесення закладу до переліку або вилучити для вилучення закладу з цього переліку, якщо він вже був доданий до переліку до цього.

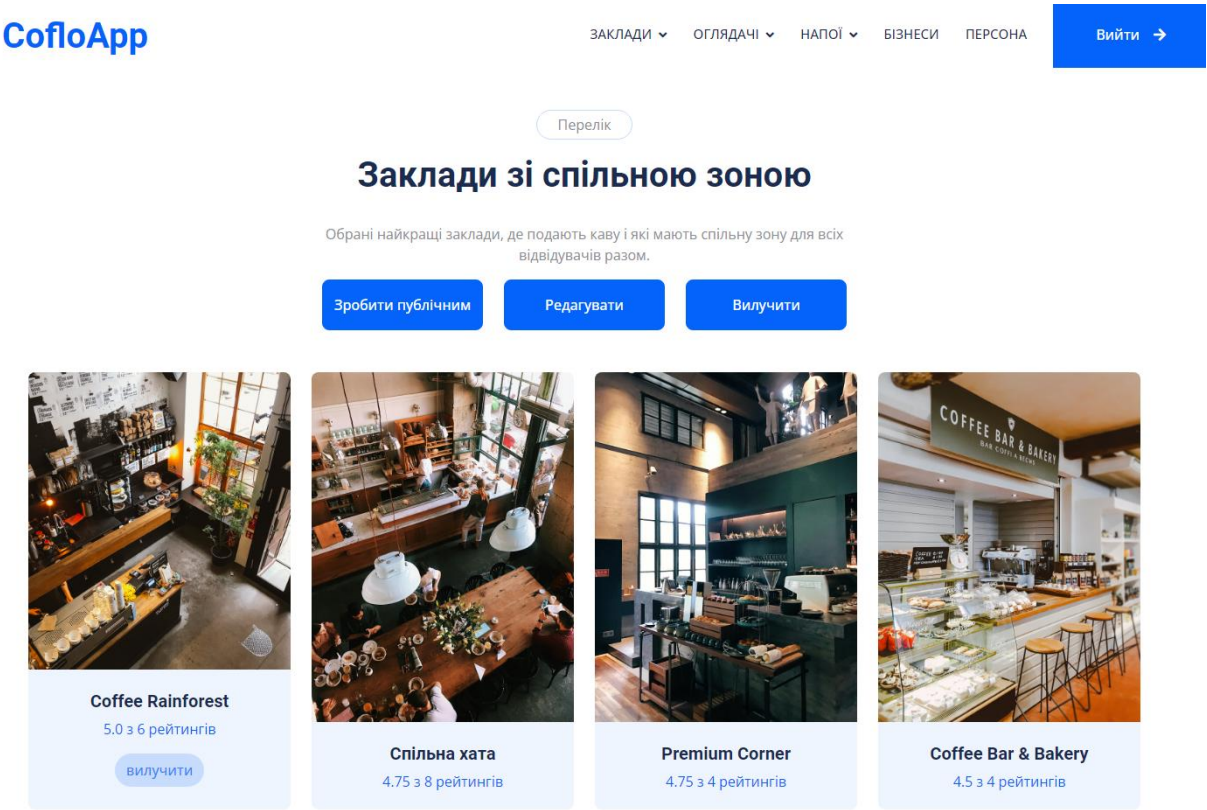


Рисунок 5.5 – Перелік закладів

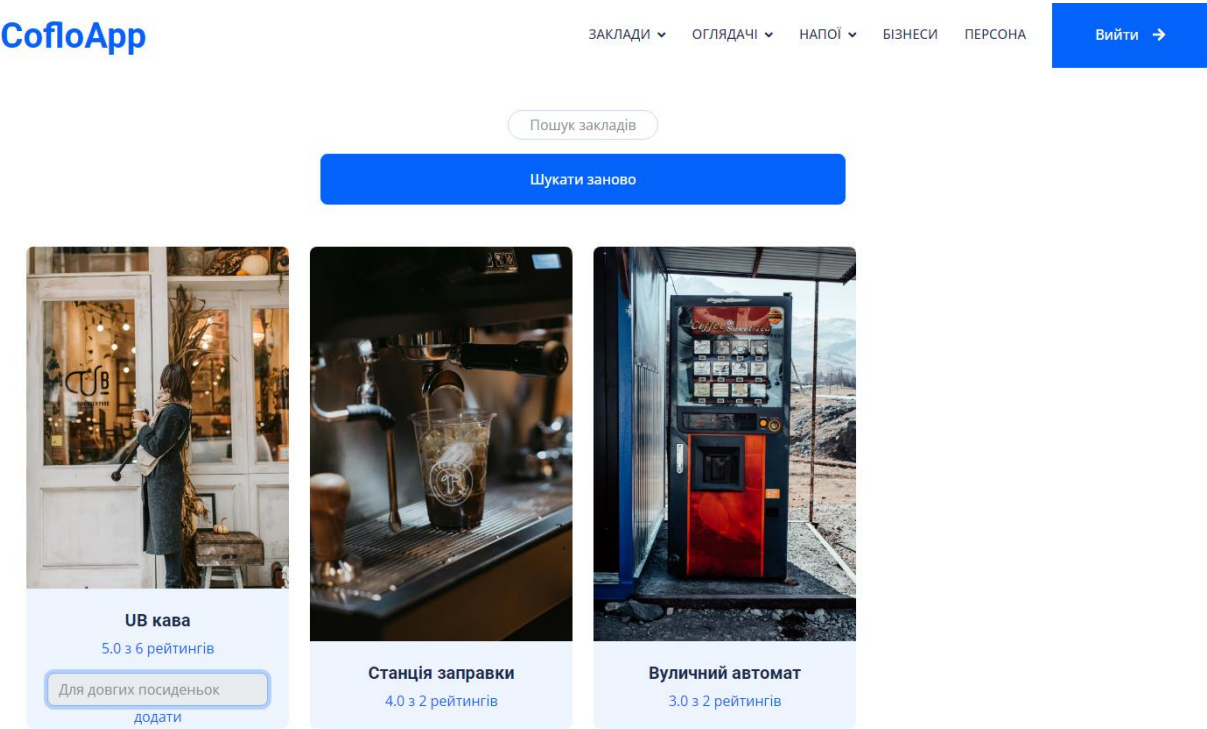


Рисунок 5.6 – Результати пошуку закладів для любителів кави

5.3 Експериментальне дослідження

У вибірці даних [14] представлені відгуки про заклади для любителів кави. Всього у вибірці 7616 відгуків і вони стосуються 78 закладів для любителів кави. Використовуючи цю вибірку, виконано експериментальне дослідження проблеми рекомендування закладів для любителів кави.

На рис. 5.7 представлено те, яким чином розподілені заклади на 2 кластери з використанням запропонованого методу, вказуючи за ординатою оцінку якості кави, а за абсцисою – оцінку за інтернет у закладі.

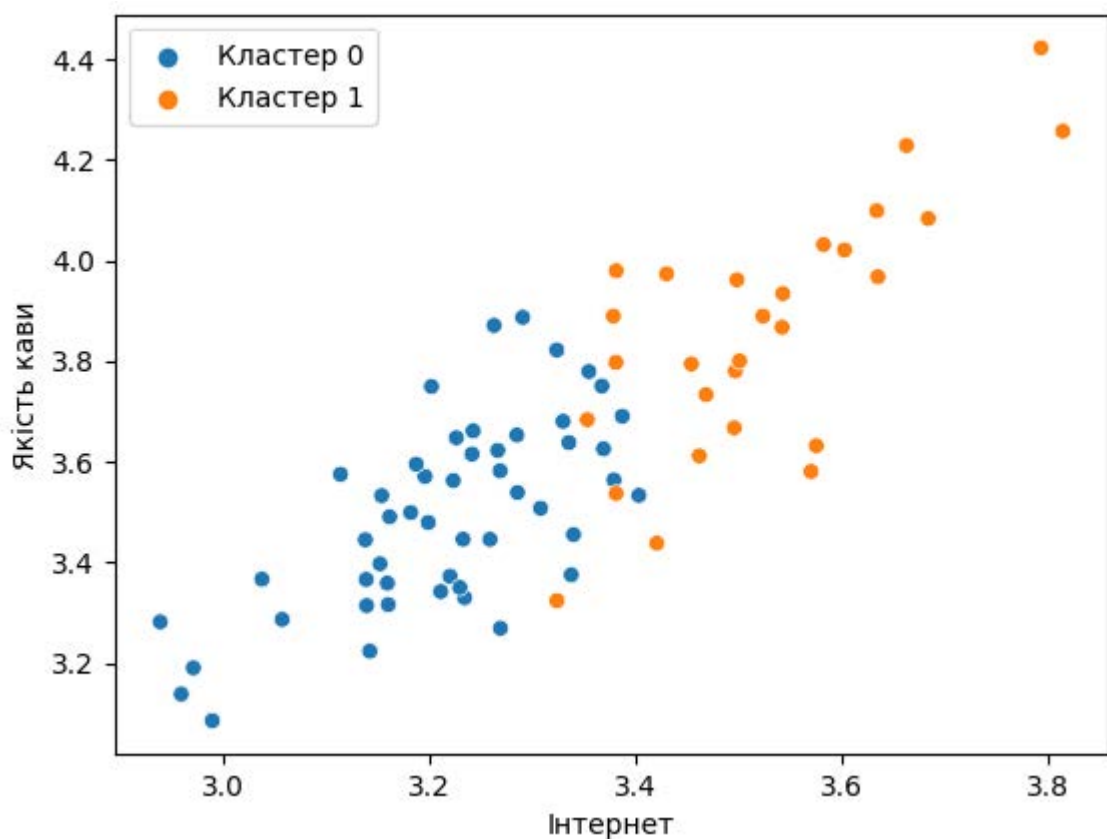


Рисунок 5.7 – Розподіл закладів на 2 кластери за якістю кави та інтернету

На рис. 5.8 і 5.9 представлено те, яким чином розподілені заклади на 2 кластери з використанням запропонованого методу, вказуючи за ординатою оцінку якості кави, а за абсцисою – відповідно оцінку розташування закладу та зручності паркування. Створено розподіл на 2 кластери з оцінками цінової

політики (рис. 5.10), зручності місць (рис. 5.11).

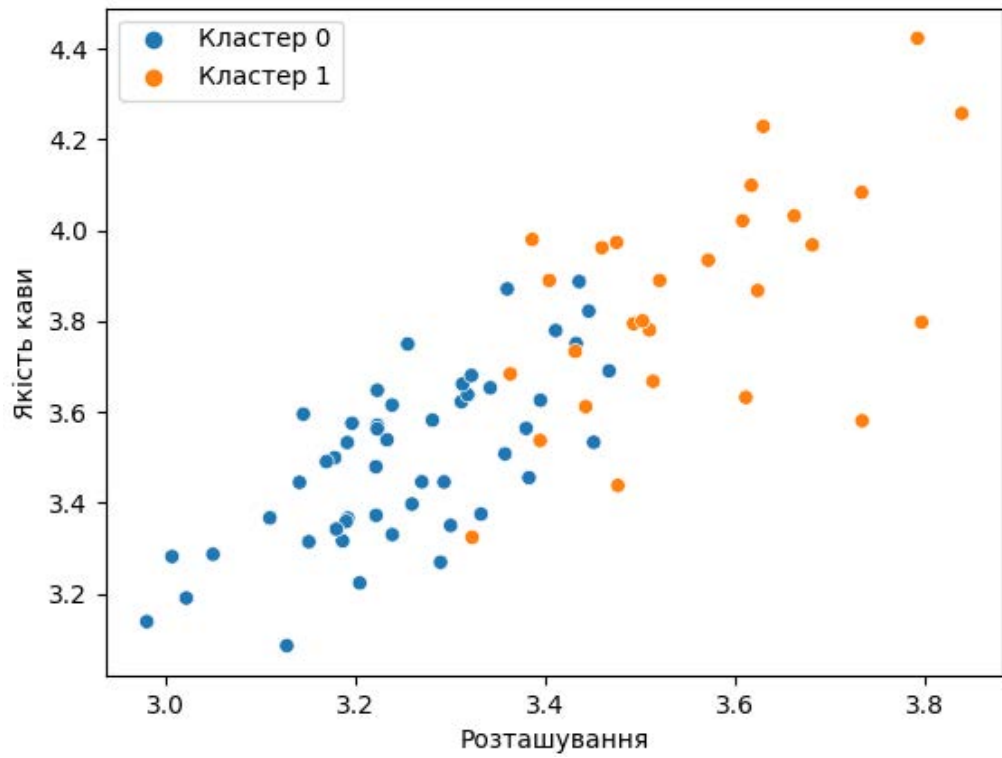


Рисунок 5.8 – Розподіл закладів на 2 кластери за якістю кави та розташування

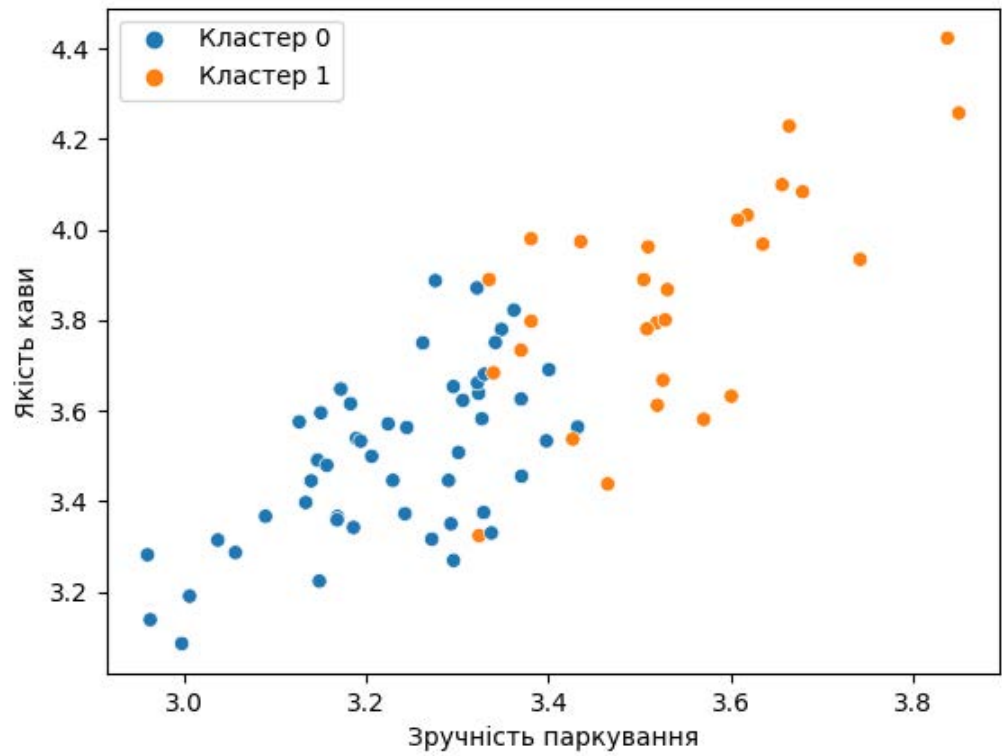


Рисунок 5.9 – Розподіл закладів на 2 кластери за якістю кави та паркування

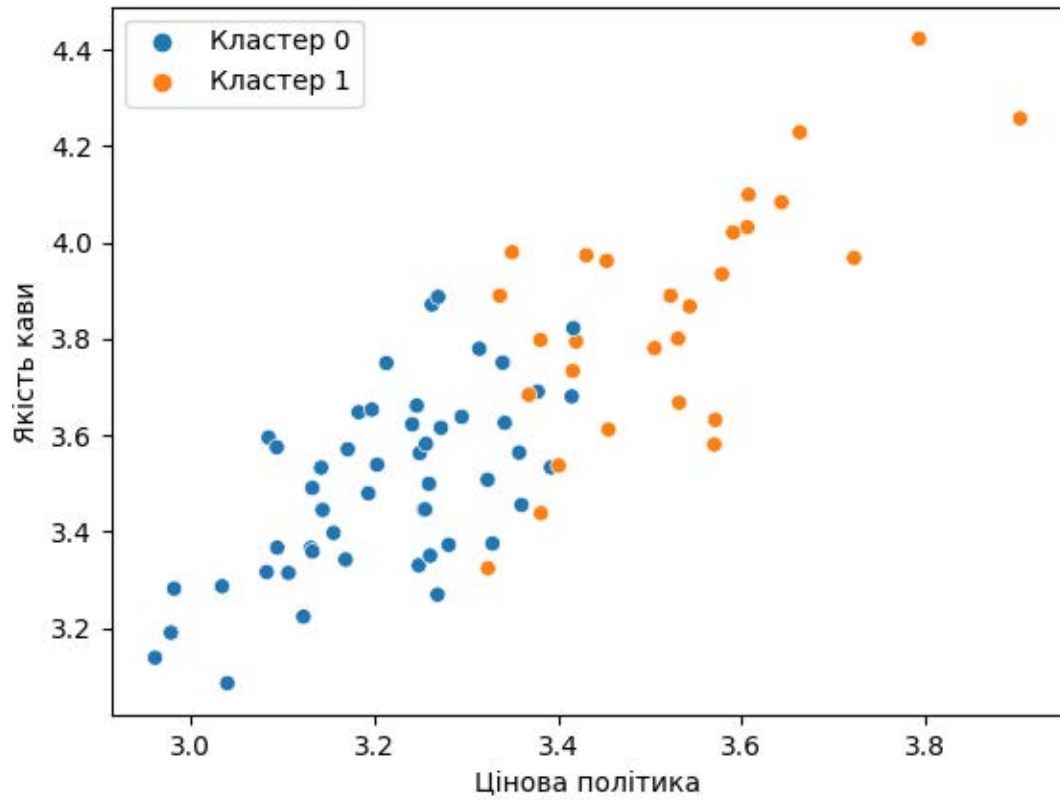


Рисунок 5.10 – Розподіл закладів на 2 кластери за якістю кави та вартістю

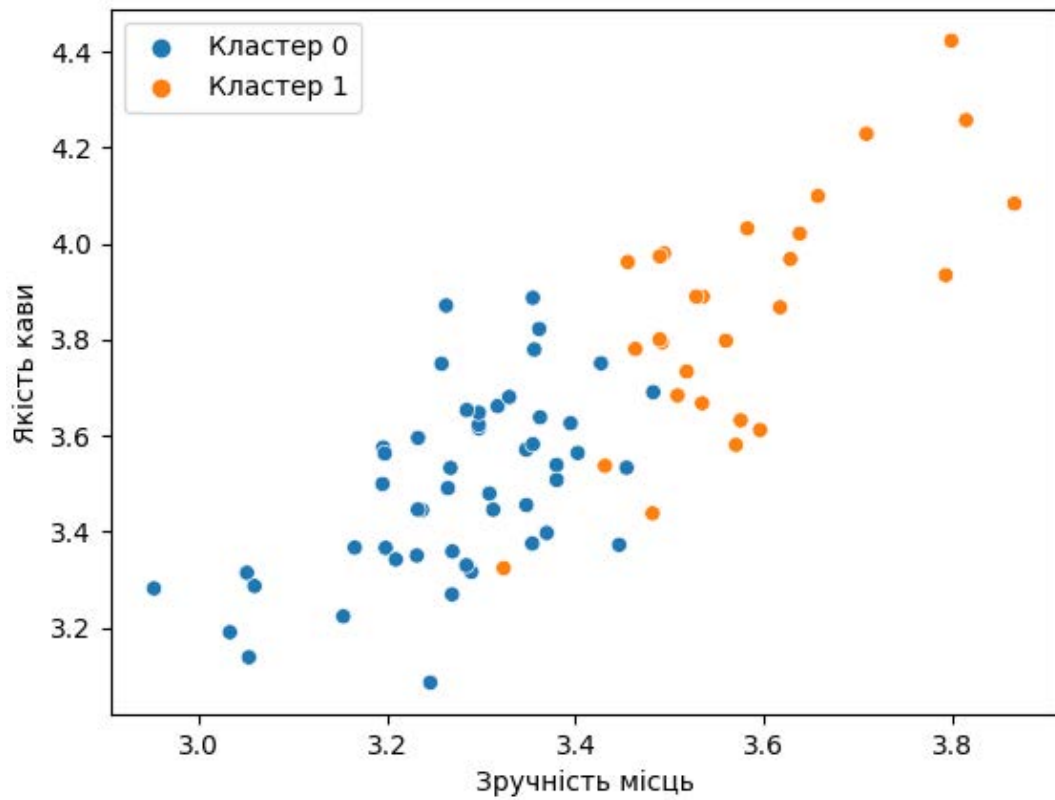


Рисунок 5.11 – Розподіл закладів на 2 кластери за якістю кави та зручністю місць

Створено розподіл на 2 кластери з оцінками обслуговування (рис. 5.12), атмосфери (рис. 5.13) за абсцисою.

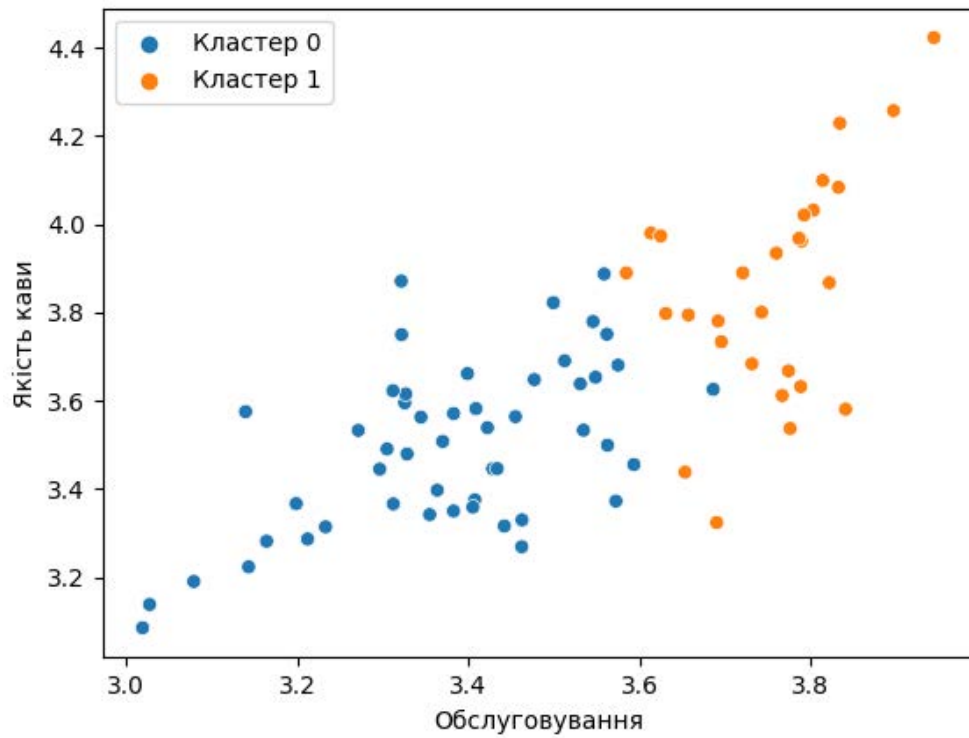


Рисунок 5.12 – Розподіл закладів на 2 кластери за якістю кави та обслуговування

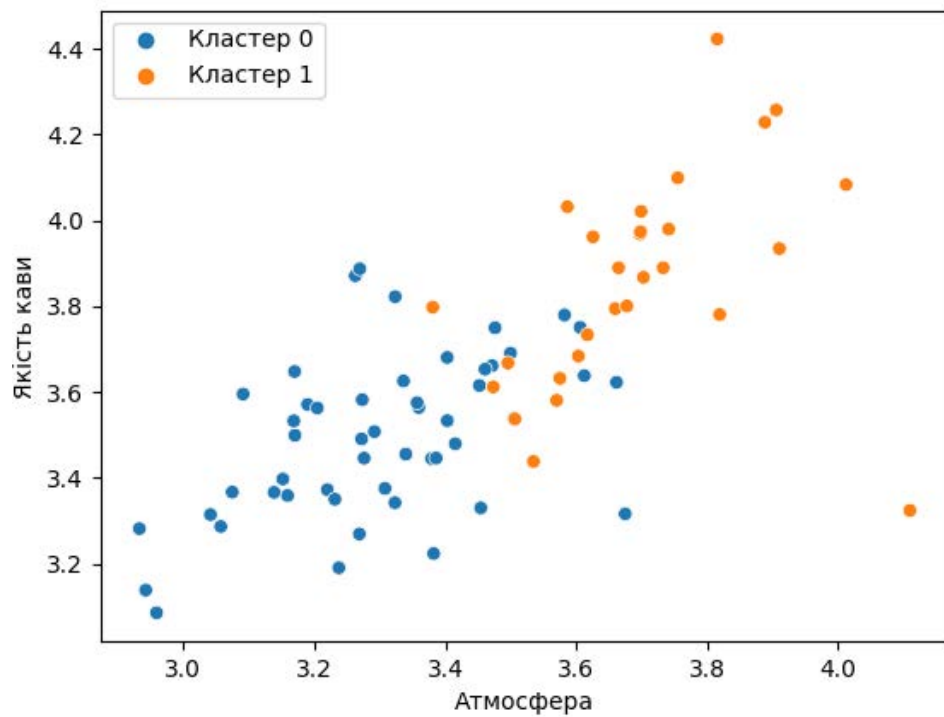


Рисунок 5.13 – Розподіл закладів на 2 кластери за якістю кави та атмосферою

Створено розподіл на 6 кластерів з оцінками інтернету (рис. 5.14), розташування (рис. 5.15), паркування (рис. 5.16), вартістю (рис. 5.17) за абсцисою та оцінкою якості кави за ординатою.

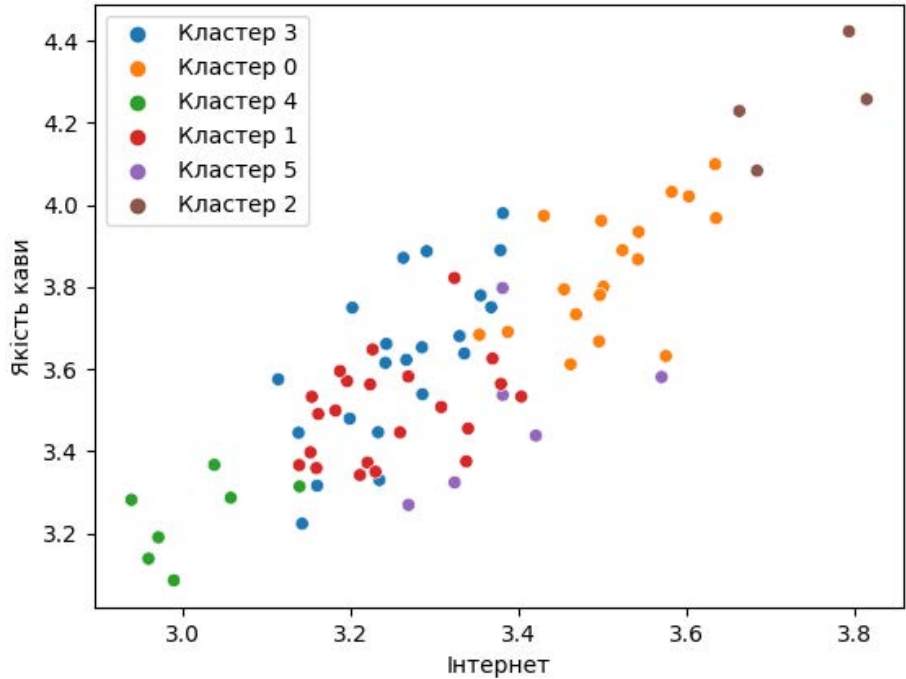


Рисунок 5.14 – Розподіл закладів на 6 кластерів за якістю кави та інтернету

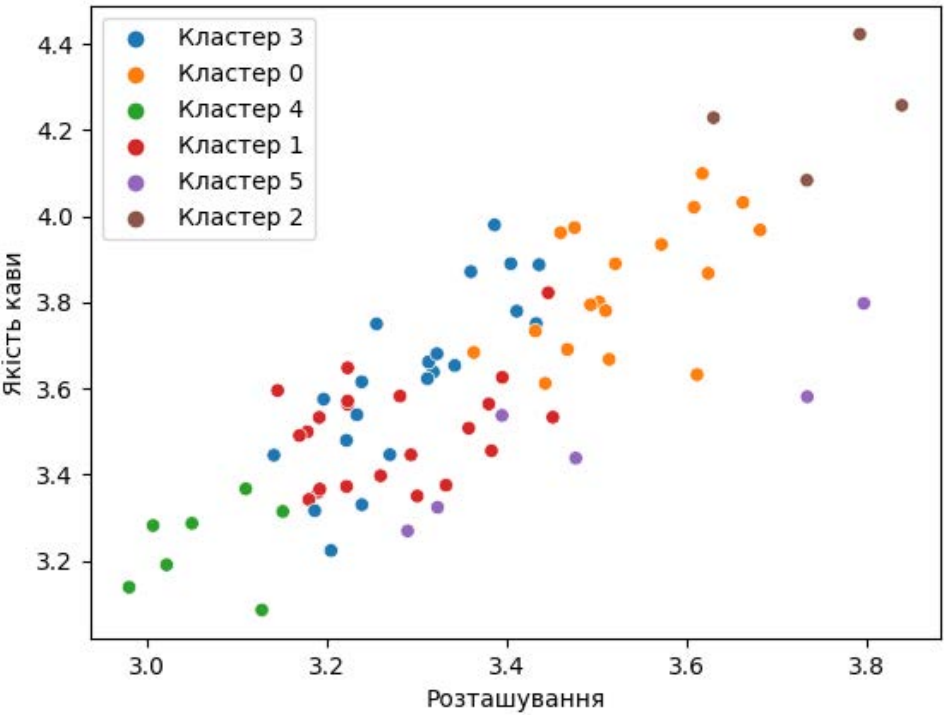


Рисунок 5.15 – Розподіл закладів на 6 кластерів за якістю кави та розташування

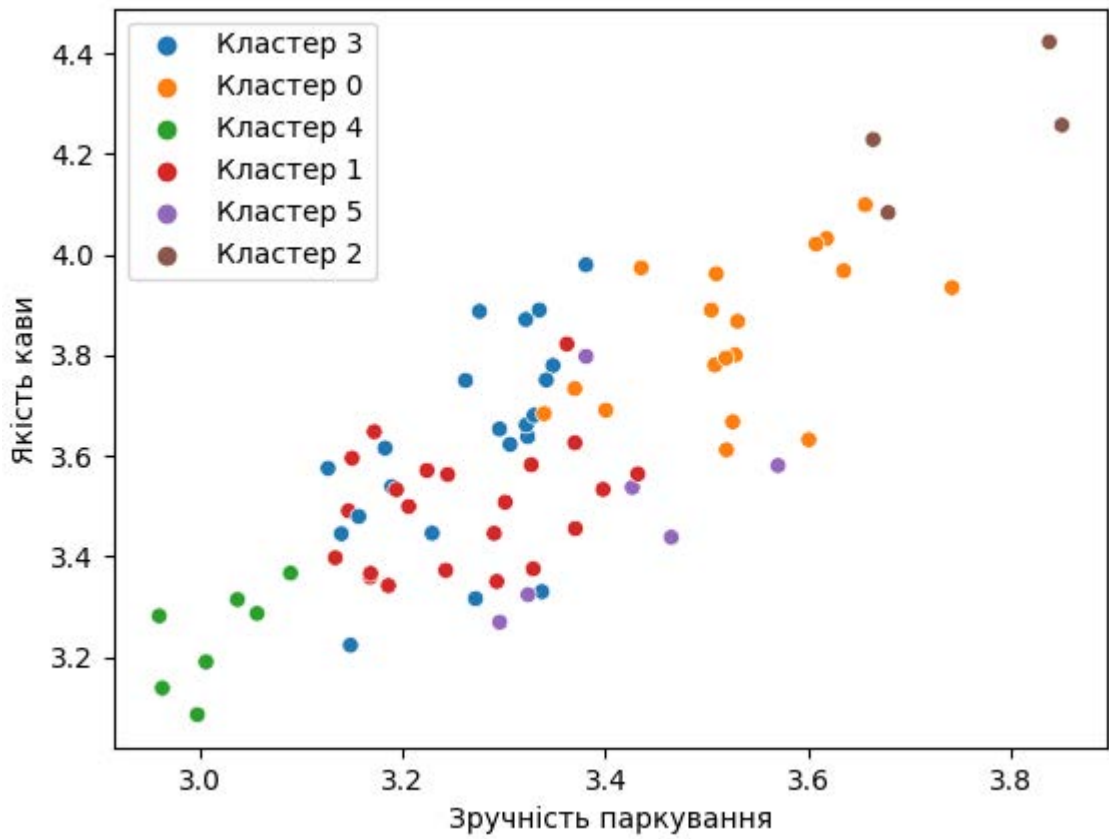


Рисунок 5.16 – Розподіл закладів на 6 кластерів за якістю кави та паркування

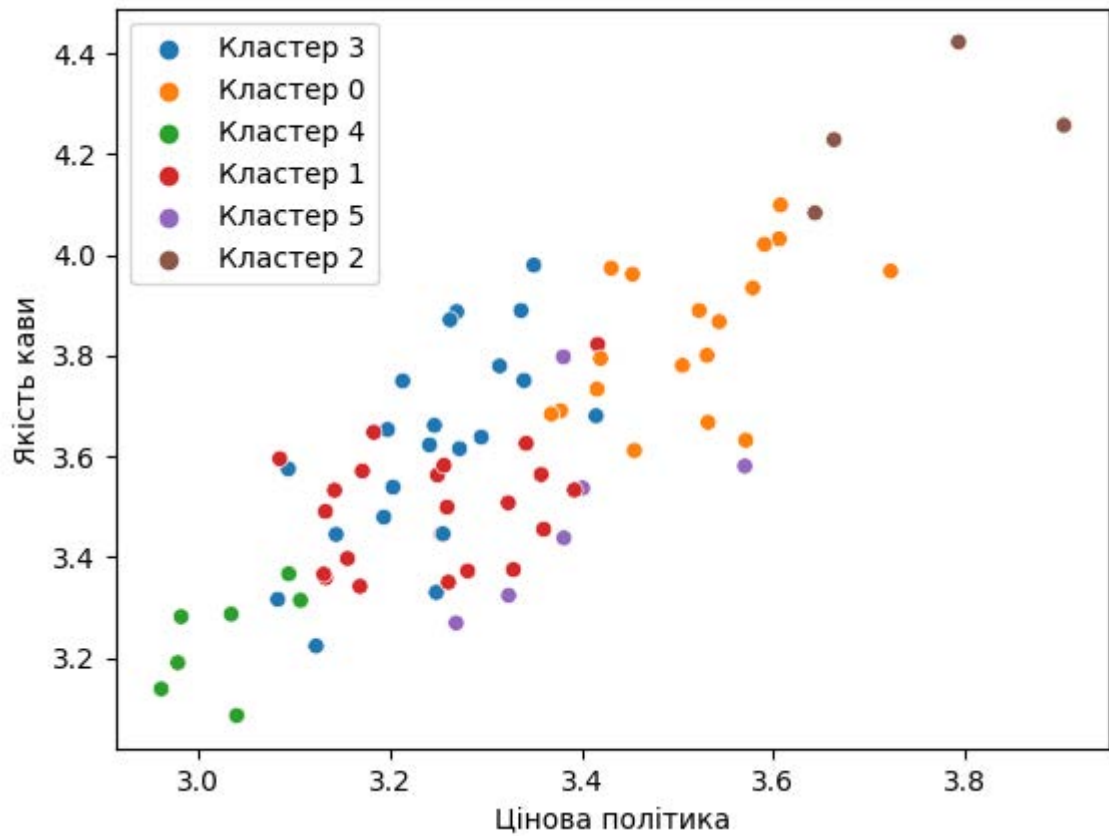


Рисунок 5.17 – Розподіл закладів на 6 кластерів за якістю кави та вартістю

Створено розподіл на 6 кластерів з оцінками зручності місць (рис. 5.18), обслуговування (рис. 5.19), атмосфери (рис. 5.20) за абсцисою.

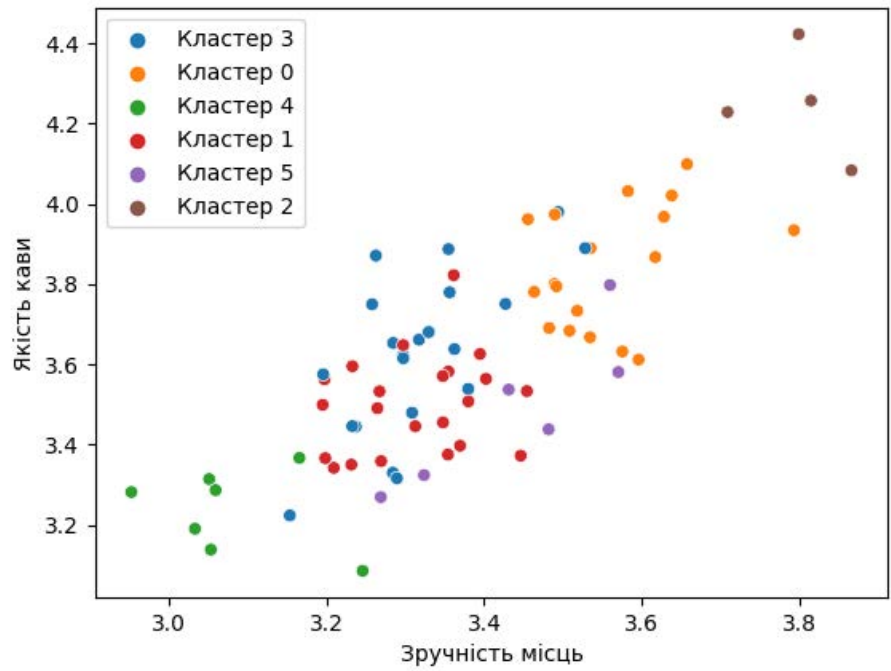


Рисунок 5.18 – Розподіл закладів на 6 кластерів за якістю кави та зручністю місць

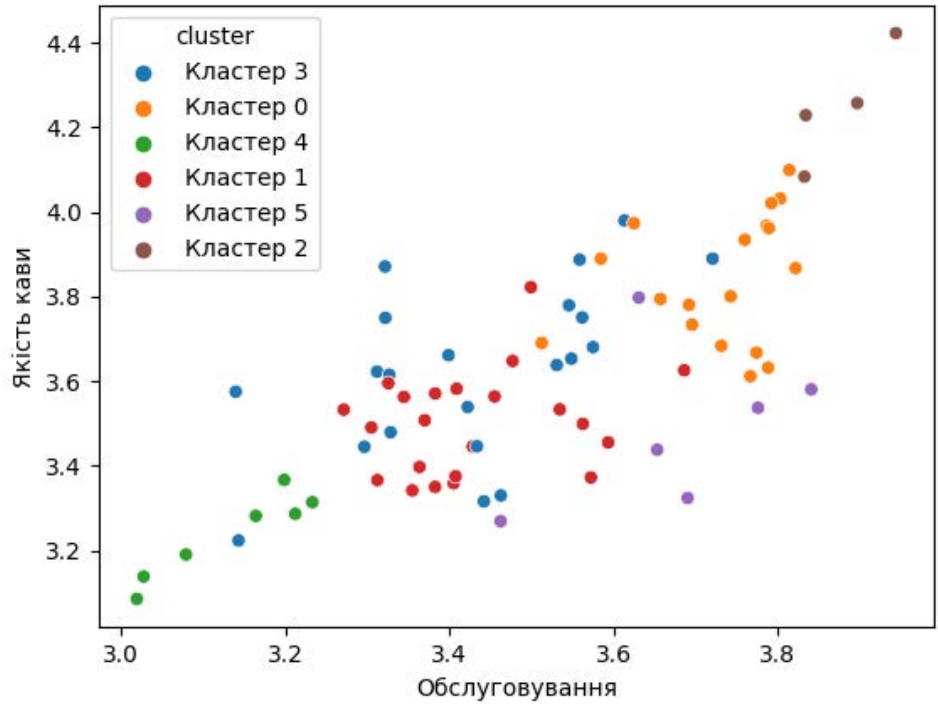


Рисунок 5.19 – Розподіл закладів на 6 кластерів за якістю кави та обслуговування

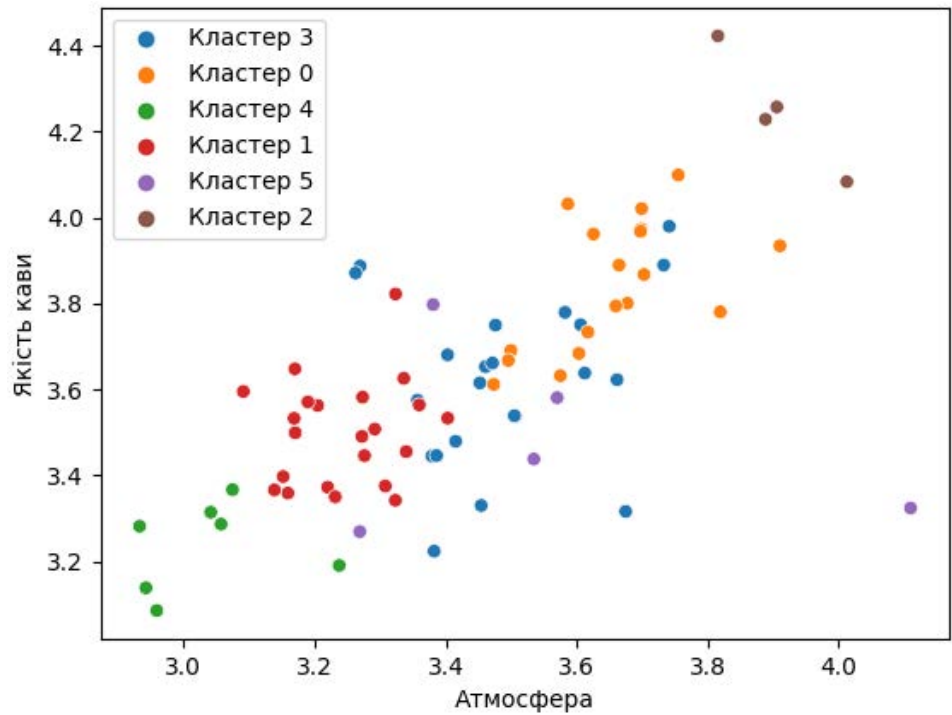


Рисунок 5.20 – Розподіл закладів на 6 кластерів за якістю кави та атмосферою

Обчислена відстань між об’єктами всередині кожного кластера за умов формування двох або трьох кластерів представлена у табл. 5.1 та 5.2.

Таблиця 5.1 – Відстань між об’єктами всередині кластера за двох кластерів

Номер кластера	Метод рекомендації закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,03245683627830732	0,08471546885760779
2	0,05031104335792566	0,06478012330532197

Таблиця 5.2 – Відстань між об’єктами всередині трьох кластерів

Номер кластера	Метод рекомендації закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,04281628724057797	0,0645249893393524
2	0,028181754403302214	0,06832378677024864
3	0,026550565406542697	0,12135900672465272

Обчислена відстань між об'єктами всередині кожного кластера за умов формування чотирьох або п'яти кластерів представлена у табл. 5.3 та 5.4 для методу рекомендування закладів для любителів кави, який запропоновано, та подібного рішення, але яке використовує напряду результати сентимент-аналізу без перетворення отриманих результатів на оцінку за відповідним критерієм.

Таблиця 5.3 – Відстань між об'єктами всередині кластера за чотирьох кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,032956772261741304	0,07207931375503486
2	0,028476058190046907	0,1202299058023305
3	0,07094980639697727	0,09323826881173951
4	0,026538858279767336	0,0381538084355378

Таблиця 5.4 – Відстань між об'єктами всередині кластера за п'ятьох кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,02986868532258074	0,04400750240048761
2	0,02979766849524352	0,04951738097073407
3	0,0330364127259454	0,07393819511145151
4	0,08878682017429758	0,1317810063337407
5	0,02653885827976732	0,06867605865432697

У табл. 5.5 і 5.6 приведена обчислена відстань між об'єктами всередині кожного кластера за умов формування шістьох або семи кластерів.

Таблиця 5.5 – Відстань між об'єктами всередині кластера за шістьох кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,029425571063932936	0,11042822706033446
2	0,01997160574279903	0,06360665463687741
3	0,02979766849524353	0,12531373536300444
4	0,03347522622004192	0,04626739327129402
5	0,026538858279767364	0,09899724851323888
6	0,08878682017429756	—

Таблиця 5.6 – Відстань між об'єктами всередині кластера за сімох кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,02581699824199029	0,060998338624502
2	0,03745129516681276	0,08951064085170382
3	0,03142932844267621	0,0714526434143487
4	0,03995961542563616	0,11301134135402381
5	0,021413517405816956	0,12514527455652566
6	0,04970941951088417	—
7	0,026538858279767333	0,04078419447241903

У табл. 5.7 і 5.8 приведена обчислена відстань між об'єктами всередині кожного кластера за умов формування восьми або дев'яти кластерів для методу рекомендування закладів для любителів кави, який запропоновано, та подібного рішення без перетворення отриманих результатів сентимент-аналізу на оцінку за відповідним критерієм.

Таблиця 5.7 – Відстань між об'єктами всередині кластера за вісьмох кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,02282979176563795	0,03478189353030913
2	0,017186555177626604	0,053683117949288546
3	0,03575312910456345	0,09083601464693287
4	0,02766288424322392	—
5	0,10293320312499993	0,08951064085170382
6	0,03420792954392445	0,1037369912356242
7	0,026538858279767354	0,12881644207659884
8	0,031970950655423505	0,06989419608455355

Таблиця 5.8 – Відстань між об'єктами всередині кластера за дев'ятьох кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,0224468812063635	0,046808173752835296
2	0,01559772262859223	0,09561263268727721
3	0,05470022262236738	0,08073674222745586
4	0,03745129516681275	0,20043576726171225
5	—	0,08951064085170382
6	0,026538858279767347	0,03881259264598431
7	0,027921265464215497	—
8	0,022209720655819363	0,044101106383538834
9	0,029848140909491793	0,03882016612525117

У табл. 5.9 приведені відповідні результати для десятих кластерів.

Середні значення відстані між об'єктами всередині кластерів для випадків наявності від 2 до 10 кластерів зведені до табл. 5.10.

Таблиця 5.9 – Відстань між об'єктами всередині кластера за десятих кластерів

Номер кластера	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
1	0,01794884255437999	0,08951064085170382
2	0,01997222971295537	0,05833655895009058
3	0,015174618155148903	0,05042341593612541
4	0,04078068193530856	0,07424779016960385
5	0,026538858279767333	0,15504775153395262
6	0,025355887038733364	0,03697379970078874
7	–	0,1238498008423291
8	0,011610937499999984	0,06202206212508094
9	0,027647813647764904	0,051437996979081246
10	0,056650200252975055	–

Таблиця 5.10 – Середня відстань між об'єктами всередині кластера

Кількість кластерів	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
2	0,04138394	0,074747796
3	0,032516202	0,084735928
4	0,039730374	0,080925324
5	0,041605689	0,073584029
6	0,037999292	0,07410221
7	0,033188433	0,07155749
8	0,037385413	0,071407412
9	0,026301567	0,070537536
10	0,024168007	0,070184982

Середні результати, як і результати за кожним кластером серед сформованого набору, вказують, що визначення оцінок за критеріями

оцінювання замість використання результатів сентимент-аналізу напряду дозволило зменшити середню квадратичну відстань між об'єктами всередині кластерів. Тож метод рекомендування закладів для любителів кави, який було запропоновано дозволяє сформувати групи подібних закладів, які є більш близькими між собою всередині груп (кластерів).

5.4 Висновки за розділом 5

Експериментальне дослідження проблеми рекомендування закладів для любителів кави реалізовано через вибірку з 7616 відгуків, які стосуються 78 закладів для любителів кави. Для дослідження використано метод рекомендування закладів для любителів кави, запропонований у роботі, та окремо застосовано сентимент-аналіз і спектральну кластеризацію. Запропонований метод дозволив створити кластери, відстань між об'єктами всередині яких є меншою ніж у кластерах, сформованих другим способом. Запропонований метод було програмно реалізовано та внесено до складу розробленого програмного забезпечення, принципи роботи з яким описано.

ВИСНОВКИ

За результатами виконання роботи досліджено проблему рекомендації закладів для любителів кави та розроблено програму для інформування любителів кави про заклади, де подається кава або напої з кави, про напої з кави, які подаються у цих закладах, використовуючи для цього пошук цих закладів і рекомендації закладів, подібних тому, який розглядає користувач. Для цього надається можливість створення рейтингів через оцінювання закладів та напоїв, які в них подаються. Розглядаються заклади будь-якого типу, включаючи стаціонарні або мобільні точки з продажу кави, в тому числі автомати. Користувачі можуть надавати інформацію про заклади, які є власним бізнесом, або про інші заклади, якщо цю інформацію не надає власник, що сприяє актуалізації інформації. Користувачі можуть визначати свої сторінки як сторінки користувачів-оглядачів, що дозволяє підписуватися на їхні сторінки, переглядати створені ними рейтинги. Доступною можливістю є створення переліків закладів та напоїв, що подаються або продаються в закладах.

Наукова новизна полягає в методі рекомендації закладів для любителів кави, який розділяє заклади у певному місці на кластери, відносячи їх до подібних, використовуючи для цього виділення з рейтингів інформації за окремими критеріями, сентимент-аналіз і визначення оцінок на його основі за кожним з критеріїв у разі відсутності таких оцінок у рейтингу та виконуючи спектральний кластерний аналіз за визначеними оцінками.

Експериментальне дослідження проблеми рекомендації закладів для любителів кави реалізовано через вибірку з 7616 відгуків, які стосуються 78 закладів для любителів кави. Для дослідження використано метод рекомендації закладів для любителів кави, запропонований у роботі, та окремо застосовано сентимент-аналіз і спектральну кластеризацію. Запропонований метод дозволив створити кластери, відстань між об'єктами всередині яких є меншою ніж у кластерах, сформованих другим способом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Allen, L. Coffee Statistics [Electronic resource] : Consumption, Preferences, & Spending / L. Allen. – Access mode : <https://www.driveresearch.com/market-research-company-blog/coffee-survey/>.
2. Restaurants, Dentists, Bars, Beauty Salons, Doctors – Yelp [Electronic resource]. – Access mode : <https://www.yelp.com/>.
3. Unlocking Powerful Consumer Connections with Location. FSQ [Electronic resource]. – Access mode : <https://foursquare.com/>.
4. Tripadvisor [Electronic resource] : Over a billion reviews & contributions for Hotels, Attractions, Restaurants, and more. – Access mode : <https://www.tripadvisor.com/>.
5. Gandhi, V. Spectral Clustering – Detailed Explanation [Electronic resource] / V. Gandhi. – Access mode : <https://www.kaggle.com/code/vipulgandhi/spectral-clustering-detailed-explanation>.
6. Chatterjee, M. Introduction to Spectral Clustering [Electronic resource] / M. Chatterjee. – Access mode : <https://www.mygreatlearning.com/blog/introduction-to-spectral-clustering/>.
7. Luxburg, U. A Tutorial on Spectral Clustering [Electronic resource] / U. Luxburg // Statistics and Computing. – 2007. – № 17. – Pp. 395-416. – Access mode : <https://arxiv.org/abs/0711.0189>.
8. Hassan, M. Sentiment Analysis – Tools, Techniques and Examples [Electronic resource] / M. Hassan. – Access mode : <https://researchmethod.net/sentiment-analysis/>.
9. Bessa, A. Lexicon-Based Sentiment Analysis [Electronic resource] : A Tutorial / A. Bessa. – Access mode : <https://www.knime.com/blog/lexicon-based-sentiment-analysis>.
10. Django overview. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/start/overview/>.
11. What is Django? [Electronic resource]. – Access mode :

<https://tutorial.djangogirls.org/en/django/>.

12. Documentation. Node.js [Electronic resource]. – Access mode :
<https://nodejs.org/en/docs>.

13. Klinik – Clinic Website Template [Electronic resource]. – Access
mode : <https://htmlcodex.com/clinic-website-template/>.

14. Austin Coffee Yelp Reviews [Electronic resource]. – Access mode :
<https://data.world/rdowns26/austin-coffee-yelp-reviews>.

ДОДАТОК А
Технічне завдання

Вступ

Велика кількість людей споживає каву, при чому робить це достатньо регулярно. Культура споживання кави в нашій країні поступово змінюється, покращується. Це призводить до важливості отримання релевантної інформації про заклади для любителів кави, які можуть зацікавити людину. Таке інформування може відбуватися в межах організації пошуку, надання інформації про конкретний заклад, а може також і як рекомендації, що розширює в підсумку отримані результати. Програму, яка має враховувати ці два напрямки, потрібно розробити в цій роботі, забезпечивши відповідний метод підтримки рекомендації закладів для любителів кави.

A.1 Підстави для розробки

Наказ № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка» з зазначенням теми роботи «Дослідження та програмна реалізація методів рекомендації закладів для любителів кави» та завдання на роботу є підставами для розробки.

A.2 Призначення розробки

Основним призначенням розробки є інформування любителів кави про заклади, де подається кава або напої з кави. Розглядаються заклади будь-якого типу, включаючи стаціонарні або мобільні точки з продажу кави, в тому числі автомати. Для забезпечення інформування складовою є також і рекомендації закладів продажу кави користувачам, що дозволяє розширити пул закладів, які використовує кожна окрема людина, що є користувачем програми.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Забезпечення функціональних характеристик повинно вкладатися в наступний набір функцій:

- додавання закладу для любителів кави, яким може бути зокрема заклад;
- оновлення даних про заклад для любителів кави;
- створення користувацького переліку закладів для любителів кави;
- надання публічного доступу до користувацького переліку закладів для любителів кави;
- закриття публічного доступу до користувацького переліку закладів для любителів кави;
- вилучення користувацького переліку закладів для любителів кави;
- внесення закладу для любителів кави до користувацького переліку;
- вилучення закладу для любителів кави з користувацького переліку;
- визначення рейтингу закладу для любителів кави;
- повторне визначення рейтингу закладу для любителів кави;
- перегляд користувацьких рейтингів закладів для любителів кави;
- перегляд рейтингів за конкретним закладом для любителів кави;
- коригування рейтингу закладу для любителів кави;
- перегляд рекомендованих закладів для любителів кави за конкретним закладом;
- реагування на рейтинг закладу для любителів кави;
- скасування реагування на рейтинг закладу для любителів кави;
- пошук закладів для любителів кави;
- пошук серед рейтингів закладу для любителів кави;
- пошук серед користувацьких рейтингів закладів для любителів кави;
- внесення фотографії закладу для любителів кави;
- вилучення фотографії закладу для любителів кави;

- внесення фотографії кави з закладу для любителів кави;
- вилучення фотографії кави з закладу для любителів кави;
- перегляд інформації про користувача;
- пошук користувачів-оглядачів;
- перегляд інформації про заклад для любителів кави;
- перегляд всіх фотографій закладу для любителів кави;
- перегляд фотографій кави з закладу для любителів кави за конкретним напоєм;
- перегляд користувацьких фотографій закладів для любителів кави;
- перегляд користувацьких фотографій кави з закладів для любителів кави;
- визначення власної сторінки як сторінки публічного користувача-оглядача;
- оформлення підписки на користувача-оглядача;
- скасування підписки на користувача-оглядача;
- перегляд користувачів, які підписалися на користувача-оглядача;
- перегляд власних підписок на користувачів-оглядачів;
- перегляд стрічки рейтингів від користувачів-оглядачів за підпискою;
- перегляд останніх рейтингів користувачів-оглядачів;
- рекомендування користувача, який створив рейтинг;
- скасування рекомендації користувача;
- перегляд власних створених рекомендацій;
- визначення закладу для любителів кави як власної справи;
- скасування визначення закладу для любителів кави як власної справи;
- підтвердження закладу для любителів кави кави як власної справи користувача;
- скасування закладу для любителів кави як власної справи користувача;
- перегляд власних бізнесів користувача;

- додавання напою з кави до закладу для любителів кави;
- перегляд напоїв за закладом для любителів кави;
- вилучення напою з кави з закладу для любителів кави;
- оновлення даних напою з кави з закладу для любителів кави;
- підтвердження даних про напій з закладу для любителів кави;
- скасування даних про напій з закладу для любителів кави;
- перегляд фотографій напоїв з кави з різних закладів;
- авторизація користувачів;
- реєстрація користувачів;
- оновлення даних користувача про себе;
- визначення рейтингу напою з кави у закладі для любителів кави;
- коригування рейтингу напою з кави у закладі для любителів кави;
- повторне визначення рейтингу напою з кави у закладі для любителів кави;
- реагування на рейтинг напою з кави у закладі для любителів кави;
- скасування реагування на рейтинг напою з кави у закладі для любителів кави;
- перегляд рейтингів напоїв за закладом для любителів кави;
- перегляд користувацьких рейтингів напоїв з кави;
- перегляд рейтингів напою за закладом для любителів кави;
- створення користувацького переліку напоїв з кави у закладах для любителів кави;
- внесення напою до користувацького переліку напоїв з кави у закладах для любителів кави;
- вилучення напою з користувацького переліку напоїв з кави у закладах для любителів кави;
- вилучення користувацького переліку напоїв з кави у закладах для любителів кави;
- перегляд самостійно створених переліків закладів для любителів кави;

- перегляд самостійно створених переліків напоїв з кави у закладах для любителів кави;
- перегляд переліку закладів для любителів кави;
- перегляд переліку напоїв з кави з різних закладів.

A.3.2 Умови експлуатації

Програмне забезпечення в підсумку має підтримуватись під час експлуатації через:

- організацію хостингу для роботи з сервером;
- розміщення на сервері вебзастосунку з базовим налаштуванням Django;
- наявність інтерпретатора python 3.

Або у складі цього серверу, або як окремий сервер БД має бути використано і налаштовано систему MySQL.

A.3.3 Вимоги до складу та параметрів технічних засобів

Вимоги до організації технічних засобів сервера включають мінімально процесор Dual Intel Xeon, який має частоту 3,2 ГГц, 4 Гб доступної оперативної пам'яті, 10 Гб доступної пам'яті на жорсткому диску.

A.4 Порядок контролю та приймання

Строки виконання дипломної кваліфікаційної роботи магістра визначені в завданні на роботу безпосередньо, включаючи строк подання роботи, календарний план виконання роботи з зазначеними етапами, результатами та строками. Захист роботи відбувається за оголошеним завчасно графіком захисту дипломних кваліфікаційних робіт і відповідно графіком роботи екзаменаційної комісії.

ДОДАТОК Б
Текст програми

```

from .models import Coffeeplace, Coffeeplacetype, Drink, Coffeeplacedrink, Timetable,
Coffeeover, Coffeeplacering, Coffeeplacelistmember, Reactiontype, Coffeeplaceringreaction,
Coffeeplacedrinkrating, Coffeeplacedrinkratingreaction, Reference, Coffeeplacelist,
Coffeeplacedrinklist, Businessowner

import datetime
import haversine
import holidays
from .countrycities import get_country, get_full_address
from geopy.geocoders import Nominatim
from django.shortcuts import render

def search_coffeeplace(request):
    coffeeplaces = Coffeeplace.objects.all()
    title_of_coffeeplace = self.request.GET.get('name')
    if title_of_coffeeplace:
        coffeeplaces = coffeeplaces.filter(place_title__icontains = title_of_coffeeplace)
    type_of_place = self.request.GET.get('ptype')
    if type_of_place:
        type_of_coffee_place = Coffeeplacetype.objects.get(name = type_of_place)
        coffeeplaces = coffeeplaces.filter(place_type = type_of_coffee_place)
    city_of_coffeeplace = self.request.GET.get('city')
    if city_of_coffeeplace:
        coffeeplaces = coffeeplaces.filter(city = city_of_coffeeplace)
    wifi_access = int(self.request.GET.get('wifi'))
    if wifi_access:
        coffeeplaces = coffeeplaces.filter(wifi_access = wifi_access)
    takeout_possibility = self.request.GET.get('takeout')
    if takeout_possibility:
        coffeeplaces = coffeeplaces.filter(takeout = True)
    card_pay_availability = self.request.GET.get('card')
    if card_pay_availability:
        coffeeplaces = coffeeplaces.filter(card_pay = True)
    apple_pay_availability = self.request.GET.get('apple')
    if apple_pay_availability:
        coffeeplaces = coffeeplaces.filter(apple_pay = True)
    android_pay_availability = self.request.GET.get('android')
    if android_pay_availability:
        coffeeplaces = coffeeplaces.filter(android_pay = True)
    open_place_access = self.request.GET.get('open_place')
    if open_place_access:
        coffeeplaces = coffeeplaces.filter(open_place = True)
    parking_type = self.request.GET.get('parking')
    if parking_type:
        coffeeplaces = coffeeplaces.filter(parking_type = parking_type)
    bike_parking_availability = self.request.GET.get('bike')
    if bike_parking_availability:
        coffeeplaces = coffeeplaces.filter(bike_parking = True)
    pets_allowed = self.request.GET.get('pets')
    if pets_allowed:
        coffeeplaces = coffeeplaces.filter(pets_allowed = True)
    delivery_availability = self.request.GET.get('delivery')
    if delivery_availability:
        coffeeplaces = coffeeplaces.filter(delivery_access = True)
    food_availability = self.request.GET.get('food')
    if food_availability:
        coffeeplaces = coffeeplaces.filter(food = True)

```

```

working_place_availability = self.request.GET.get('work')
if working_place_availability:
    coffeeplaces = coffeeplaces.filter(working_place = True)
special_access_possibility = self.request.GET.get('special')
if special_access_possibility:
    coffeeplaces = coffeeplaces.filter(special_access = True)
min_rating_of_coffeeplace = float(self.request.GET.get('minrating'))
if not min_rating_of_coffeeplace or min_rating_of_coffeeplace < 0 or
min_rating_of_coffeeplace > 5:
    min_rating_of_coffeeplace = 0
    coffeeplaces = coffeeplaces.filter(average_rating__gte = min_rating_of_coffeeplace)
max_rating_of_coffeeplace = float(self.request.GET.get('maxrating'))
if not max_rating_of_coffeeplace or max_rating_of_coffeeplace > 5 or
max_rating_of_coffeeplace < 0:
    max_rating_of_coffeeplace = 5.0
    coffeeplaces = coffeeplaces.filter(average_rating__lte = max_rating_of_coffeeplace)
min_coffee_rating = float(self.request.GET.get('mincoffee'))
if not min_coffee_rating or min_coffee_rating < 0 or min_coffee_rating > 5:
    min_coffee_rating = 0
    coffeeplaces = coffeeplaces.filter(coffee_rating__gte = min_coffee_rating)
max_coffee_rating = float(self.request.GET.get('maxcoffee'))
if not max_coffee_rating or max_coffee_rating < 0 or max_coffee_rating > 5:
    max_coffee_rating = 5.0
    coffeeplaces = coffeeplaces.filter(coffee_rating__lte = max_coffee_rating)
min_vibe_rating = float(self.request.GET.get('minvibe'))
if not min_vibe_rating or min_vibe_rating < 0 or min_vibe_rating > 5:
    min_vibe_rating = 0
    coffeeplaces = coffeeplaces.filter(vibe_rating__gte = min_vibe_rating)
max_vibe_rating = float(self.request.GET.get('maxvibe'))
if not max_vibe_rating or max_vibe_rating < 0 or max_vibe_rating > 5:
    max_vibe_rating = 5.0
    coffeeplaces = coffeeplaces.filter(vibe_rating__lte = max_vibe_rating)
min_service_rating = float(self.request.GET.get('minservice'))
if not min_service_rating or min_service_rating < 0 or min_service_rating > 5:
    min_service_rating = 0
    coffeeplaces = coffeeplaces.filter(service_rating__gte = min_service_rating)
max_service_rating = float(self.request.GET.get('maxservice'))
if not max_service_rating or max_service_rating < 0 or max_service_rating > 5:
    max_service_rating = 5.0
    coffeeplaces = coffeeplaces.filter(service_rating__lte = max_service_rating)
min_seating_rating = float(self.request.GET.get('minseating'))
if not min_seating_rating or min_seating_rating < 0 or min_seating_rating > 5:
    min_seating_rating = 0
    coffeeplaces = coffeeplaces.filter(seating_rating__gte = min_seating_rating)
max_seating_rating = float(self.request.GET.get('maxseating'))
if not max_seating_rating or max_seating_rating < 0 or max_seating_rating > 5:
    max_seating_rating = 5.0
    coffeeplaces = coffeeplaces.filter(seating_rating__lte = max_seating_rating)
min_price_rating = float(self.request.GET.get('minprice'))
if not min_price_rating or min_price_rating < 0 or min_price_rating > 5:
    min_price_rating = 0
    coffeeplaces = coffeeplaces.filter(price_rating__gte = min_price_rating)
max_price_rating = float(self.request.GET.get('maxprice'))
if not max_price_rating or max_price_rating < 0 or max_price_rating > 5:
    max_price_rating = 5.0
    coffeeplaces = coffeeplaces.filter(price_rating__lte = max_price_rating)

```

```

min_parking_rating = float(self.request.GET.get('minparking'))
if not min_parking_rating or min_parking_rating < 0 or min_parking_rating > 5:
    min_parking_rating = 0
coffeepplaces = coffeepplaces.filter(parking_rating__gte = min_parking_rating)
max_parking_rating = float(self.request.GET.get('maxparking'))
if not max_parking_rating or max_parking_rating < 0 or max_parking_rating > 5:
    max_parking_rating = 5.0
coffeepplaces = coffeepplaces.filter(parking_rating__lte = max_parking_rating)
min_location_rating = float(self.request.GET.get('minlocation'))
if not min_location_rating or min_location_rating < 0 or min_location_rating > 5:
    min_location_rating = 0
coffeepplaces = coffeepplaces.filter(location_rating__gte = min_location_rating)
max_location_rating = float(self.request.GET.get('maxlocation'))
if not max_location_rating or max_location_rating < 0 or max_location_rating > 5:
    max_location_rating = 5.0
coffeepplaces = coffeepplaces.filter(location_rating__lte = max_location_rating)
min_internet_rating = float(self.request.GET.get('mininternet'))
if not min_internet_rating or min_internet_rating < 0 or min_internet_rating > 5:
    min_internet_rating = 0
coffeepplaces = coffeepplaces.filter(internet_rating__gte = min_internet_rating)
max_internet_rating = float(self.request.GET.get('maxinternet'))
if not max_internet_rating or max_internet_rating < 0 or max_internet_rating > 5:
    max_internet_rating = 5.0
coffeepplaces = coffeepplaces.filter(internet_rating__lte = max_internet_rating)
drink_name = self.request.GET.get('drink')
if drink_name:
    drink_object = Drink.objects.get(name = drink_name)
    coffeepplaces_with_drink = Coffeepplacedrink.filter(drink =
drink_object).select_related('coffeepplace').values_list('coffeepplace__id')
    coffeepplaces = coffeepplaces.filter(id__in = coffeepplaces_with_drink)
coffeepplaces = coffeepplaces.select_related('place_type').values()
current_moment = self.request.GET.get('now')
period_date_str = self.request.GET.get('date')
period_date = datetime.datetime.strptime(period_date_str, '%m.%d.%Y').date()
start_period_time = self.request.GET.get('mintime')
if ':' in start_period_time:
    time_split = start_period_time.split(':')
    start_period_time = datetime.time(int(time_split[0]), int(time_split[1]), 0)
else:
    start_period_time = datetime.time(0, 0, 0)
finish_period_time = self.request.GET.get('maxtime')
if ':' in finish_period_time:
    time_split = finish_period_time.split(':')
    finish_period_time = datetime.time(int(time_split[0]), int(time_split[1]), 0)
else:
    finish_period_time = datetime.time(23, 59, 59)
if current_moment or period_date:
    full_datetime_for_now = datetime.datetime.now()
    hour_now = full_datetime_for_now.hour
    minute_now = full_datetime_for_now.minute
    time_now = datetime.time(hour_now, minute_now, 0)
    day_of_the_week = full_datetime_for_now.weekday()
    holiday_day = False
    current_date = datetime.date(full_datetime_for_now.year,
full_datetime_for_now.month, full_datetime_for_now.day)
    if period_date:

```

```

    day_of_the week = period_date.weekday()
coffeeplaces_search_results = []
for coffee_place in coffeeplaces:
    country_of_city = get_country(coffee_place.city)
    if country_of_city == 'AL'
        holidays_dates = holidays.AL()
    elif country_of_city == 'DZ'
        holidays_dates = holidays.DZ()
    elif country_of_city == 'AS'
        holidays_dates = holidays.AS()
    elif country_of_city == 'AD'
        holidays_dates = holidays.AD()
    elif country_of_city == 'AO'
        holidays_dates = holidays.AO()
    elif country_of_city == 'AR'
        holidays_dates = holidays.AR()
    elif country_of_city == 'AM'
        holidays_dates = holidays.AM()
    elif country_of_city == 'AW'
        holidays_dates = holidays.AW()
    elif country_of_city == 'AU'
        holidays_dates = holidays.AU()
    elif country_of_city == 'AT'
        holidays_dates = holidays.AT()
    elif country_of_city == 'AZ'
        holidays_dates = holidays.AZ()
    elif country_of_city == 'BS'
        holidays_dates = holidays.BS()
    elif country_of_city == 'BH'
        holidays_dates = holidays.BH()
    elif country_of_city == 'BD'
        holidays_dates = holidays.BD()
    elif country_of_city == 'BB'
        holidays_dates = holidays.BB()
    elif country_of_city == 'BY'
        holidays_dates = holidays.BY()
    elif country_of_city == 'BE'
        holidays_dates = holidays.BE()
    elif country_of_city == 'BZ'
        holidays_dates = holidays.BZ()
    elif country_of_city == 'BO'
        holidays_dates = holidays.BO()
    elif country_of_city == 'BA'
        holidays_dates = holidays.BA()
    elif country_of_city == 'BW'
        holidays_dates = holidays.BW()
    elif country_of_city == 'BR'
        holidays_dates = holidays.BR()
    elif country_of_city == 'BN'
        holidays_dates = holidays.BN()
    elif country_of_city == 'BG'
        holidays_dates = holidays.BG()
    elif country_of_city == 'BF'
        holidays_dates = holidays.BF()
    elif country_of_city == 'BI'
        holidays_dates = holidays.BI()

```

```

elif country_of_city == 'KH'
    holidays_dates = holidays.KH()
elif country_of_city == 'CM'
    holidays_dates = holidays.CM()
elif country_of_city == 'CA'
    holidays_dates = holidays.CA()
elif country_of_city == 'TD'
    holidays_dates = holidays.TD()
elif country_of_city == 'CL'
    holidays_dates = holidays.CL()
elif country_of_city == 'CN'
    holidays_dates = holidays.CN()
elif country_of_city == 'CO'
    holidays_dates = holidays.CO()
elif country_of_city == 'CR'
    holidays_dates = holidays.CR()
elif country_of_city == 'HR'
    holidays_dates = holidays.HR()
elif country_of_city == 'CU'
    holidays_dates = holidays.CU()
elif country_of_city == 'CW'
    holidays_dates = holidays.CW()
elif country_of_city == 'CY'
    holidays_dates = holidays.CY()
elif country_of_city == 'CZ'
    holidays_dates = holidays.CZ()
elif country_of_city == 'DK'
    holidays_dates = holidays.DK()
elif country_of_city == 'DJ'
    holidays_dates = holidays.DJ()
elif country_of_city == 'DO'
    holidays_dates = holidays.DO()
elif country_of_city == 'EC'
    holidays_dates = holidays.EC()
elif country_of_city == 'EG'
    holidays_dates = holidays.EG()
elif country_of_city == 'SV'
    holidays_dates = holidays.SV()
elif country_of_city == 'EE'
    holidays_dates = holidays.EE()
elif country_of_city == 'SZ'
    holidays_dates = holidays.SZ()
elif country_of_city == 'ET'
    holidays_dates = holidays.ET()
elif country_of_city == 'FI'
    holidays_dates = holidays.FI()
elif country_of_city == 'FR'
    holidays_dates = holidays.FR()
elif country_of_city == 'GA'
    holidays_dates = holidays.GA()
elif country_of_city == 'GE'
    holidays_dates = holidays.GE()
elif country_of_city == 'DE'
    holidays_dates = holidays.DE()
elif country_of_city == 'GR'
    holidays_dates = holidays.GR()

```

```

elif country_of_city == 'GU'
    holidays_dates = holidays.GU()
elif country_of_city == 'GT'
    holidays_dates = holidays.GT()
elif country_of_city == 'HN'
    holidays_dates = holidays.HN()
elif country_of_city == 'HK'
    holidays_dates = holidays.HK()
elif country_of_city == 'HU'
    holidays_dates = holidays.HU()
elif country_of_city == 'IS'
    holidays_dates = holidays.IS()
elif country_of_city == 'IN'
    holidays_dates = holidays.IN()
elif country_of_city == 'ID'
    holidays_dates = holidays.ID()
elif country_of_city == 'IR'
    holidays_dates = holidays.IR()
elif country_of_city == 'IE'
    holidays_dates = holidays.IE()
elif country_of_city == 'IM'
    holidays_dates = holidays.IM()
elif country_of_city == 'IL'
    holidays_dates = holidays.IL()
elif country_of_city == 'IT'
    holidays_dates = holidays.IT()
elif country_of_city == 'JM'
    holidays_dates = holidays.JM()
elif country_of_city == 'JP'
    holidays_dates = holidays.JP()
elif country_of_city == 'KZ'
    holidays_dates = holidays.KZ()
elif country_of_city == 'KE'
    holidays_dates = holidays.KE()
elif country_of_city == 'KG'
    holidays_dates = holidays.KG()
if country_of_city == 'LA'
    holidays_dates = holidays.LA()
elif country_of_city == 'LV'
    holidays_dates = holidays.LV()
elif country_of_city == 'LS'
    holidays_dates = holidays.LS()
elif country_of_city == 'LI'
    holidays_dates = holidays.LI()
elif country_of_city == 'LT'
    holidays_dates = holidays.LT()
elif country_of_city == 'LU'
    holidays_dates = holidays.LU()
elif country_of_city == 'MG'
    holidays_dates = holidays.MG()
elif country_of_city == 'MW'
    holidays_dates = holidays.MW()
elif country_of_city == 'MY'
    holidays_dates = holidays.MY()
elif country_of_city == 'MV'
    holidays_dates = holidays.MV()

```

```

elif country_of_city == 'MT'
    holidays_dates = holidays.MT()
elif country_of_city == 'MH'
    holidays_dates = holidays.MH()
elif country_of_city == 'MX'
    holidays_dates = holidays.MX()
elif country_of_city == 'MD'
    holidays_dates = holidays.MD()
elif country_of_city == 'MC'
    holidays_dates = holidays.MC()
elif country_of_city == 'ME'
    holidays_dates = holidays.ME()
elif country_of_city == 'MA'
    holidays_dates = holidays.MA()
elif country_of_city == 'MZ'
    holidays_dates = holidays.MZ()
elif country_of_city == 'NA'
    holidays_dates = holidays.NA()
elif country_of_city == 'NL'
    holidays_dates = holidays.NL()
elif country_of_city == 'NZ'
    holidays_dates = holidays.NZ()
elif country_of_city == 'NI'
    holidays_dates = holidays.NI()
elif country_of_city == 'NG'
    holidays_dates = holidays.NG()
elif country_of_city == 'MP'
    holidays_dates = holidays.MP()
elif country_of_city == 'MK'
    holidays_dates = holidays.MK()
elif country_of_city == 'NO'
    holidays_dates = holidays.NO()
elif country_of_city == 'PK'
    holidays_dates = holidays.PK()
elif country_of_city == 'PA'
    holidays_dates = holidays.PA()
elif country_of_city == 'PY'
    holidays_dates = holidays.PY()
elif country_of_city == 'PE'
    holidays_dates = holidays.PE()
elif country_of_city == 'PH'
    holidays_dates = holidays.PH()
elif country_of_city == 'PL'
    holidays_dates = holidays.PL()
elif country_of_city == 'PT'
    holidays_dates = holidays.PT()
elif country_of_city == 'PR'
    holidays_dates = holidays.PR()
elif country_of_city == 'RO'
    holidays_dates = holidays.RO()
elif country_of_city == 'SM'
    holidays_dates = holidays.SM()
elif country_of_city == 'SA'
    holidays_dates = holidays.SA()
elif country_of_city == 'RS'
    holidays_dates = holidays.RS()

```

```

elif country_of_city == 'SG'
    holidays_dates = holidays.SG()
elif country_of_city == 'SK'
    holidays_dates = holidays.SK()
elif country_of_city == 'SI'
    holidays_dates = holidays.SI()
elif country_of_city == 'ZA'
    holidays_dates = holidays.ZA()
elif country_of_city == 'KR'
    holidays_dates = holidays.KR()
elif country_of_city == 'ES'
    holidays_dates = holidays.ES()
elif country_of_city == 'SE'
    holidays_dates = holidays.SE()
elif country_of_city == 'CH'
    holidays_dates = holidays.CH()
elif country_of_city == 'TW'
    holidays_dates = holidays.TW()
elif country_of_city == 'TZ'
    holidays_dates = holidays.TZ()
elif country_of_city == 'TH'
    holidays_dates = holidays.TH()
elif country_of_city == 'TL'
    holidays_dates = holidays.TL()
elif country_of_city == 'TO'
    holidays_dates = holidays.TO()
elif country_of_city == 'TN'
    holidays_dates = holidays.TN()
elif country_of_city == 'TR'
    holidays_dates = holidays.TR()
elif country_of_city == 'UA'
    holidays_dates = holidays.UA()
elif country_of_city == 'AE'
    holidays_dates = holidays.AE()
elif country_of_city == 'GB'
    holidays_dates = holidays.GB()
elif country_of_city == 'UM'
    holidays_dates = holidays.UM()
elif country_of_city == 'US'
    holidays_dates = holidays.US()
elif country_of_city == 'UY'
    holidays_dates = holidays.UY()
elif country_of_city == 'UZ'
    holidays_dates = holidays.UZ()
elif country_of_city == 'VU'
    holidays_dates = holidays.VU()
elif country_of_city == 'VA'
    holidays_dates = holidays.VA()
elif country_of_city == 'VE'
    holidays_dates = holidays.VE()
elif country_of_city == 'VN'
    holidays_dates = holidays.VN()
elif country_of_city == 'VI'
    holidays_dates = holidays.VI()
elif country_of_city == 'ZM'
    holidays_dates = holidays.ZM()

```

```

elif country_of_city == 'ZW'
    holidays_dates = holidays.ZW()
if current_moment and (current_date in holidays_dates):
    holiday_day = True
elif period_date and (period_date in holidays_dates):
    holiday_day = True
delete_coffee_place = False
check_finished = False
if holiday_day:
    try:
        day_timetable = Timetable.objects.get(coffeeplace = coffee_place, day_type = 7)
        if current_moment and (day_timetable.open_time < time_now or
day_timetable.finish_time >= time_now):
            delete_coffee_place = True
            elif period_date and (day_timetable.open_time < start_period_time and
day_timetable.open_time <= finish_period_time) or (day_timetable.finish_time >= start_period_time
and day_timetable.finish_time > finish_period_time):
                delete_coffee_place = True
                check_finished = True
            except Timetable.DoesNotExist:
                coffee_place.timetable = 'no+'
        if not holiday_day or not check_finished:
            try:
                day_timetable = Timetable.objects.get(coffeeplace = coffee_place, day_type =
day_of_the week)
                if current_moment and (day_timetable.open_time < time_now or
day_timetable.finish_time >= time_now):
                    delete_coffee_place = True
                    elif period_date and (day_timetable.open_time < start_period_time and
day_timetable.open_time <= finish_period_time) or (day_timetable.finish_time >= start_period_time
and day_timetable.finish_time > finish_period_time):
                        delete_coffee_place = True
                        except Timetable.DoesNotExist:
                            coffee_place.timetable = 'no'
                    if not delete_coffee_place:
                        coffeeplaces_search_results.append(coffee_place)
            distance_from_address = self.request.GET.get('dist')
            if not distance_from_address or distance_from_address < 0:
                distance_from_address = 1000
            street_of_coffeeplace = self.request.GET.get('address')
            if street_of_coffeeplace:
                address_of_search = get_full_address(street_of_coffeeplace)
                nminatim_connection = Nominatim(user_agent='cofloapp')
                location_of_search = nminatim_connection.geocode(address_of_search)
                location_of_search = (location_of_search.latitude, location_of_search.longitude)
            for coffee_place in coffeeplaces_search_results:
                coffee_place_address = coffee_place.address.split('.')
                coffee_place_address_latitude = float(coffee_place_address[0])
                coffee_place_address_longitude = float(coffee_place_address[1])
                coffee_place_address = (coffee_place_address_latitude,
coffee_place_address_longitude)
                distance_in_meters = haversine.haversine(location_of_search, coffee_place_address,
unit = haversine.Unit.METERS)
                if distance_in_meters > distance_from_address:
                    coffeeplaces_search_results.remove(coffee_place)
            coffeeplaces_search_results_with_user_information = []

```

```

    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
        for coffee_place in coffeeplaces_search_results:
            coffeeloover_ratings = Coffeeplacering.objects.filter(coffeeplace = coffee_place,
creator_coffeeloover = coffeeloover).order_by('-created_time')
            coffee_place.ratings_count = coffeeloover_ratings.count()
            coffee_place.last_rating = coffeeloover_ratings.values()[0]
            coffeeloover_lists = Coffeeplacelistmember.objects.filter(coffeeplace = coffee_place,
placelist_creator_coffeeloover =
coffeeloover).select_related('coffeeplacelist').values_list('coffeeplacelist')
            coffee_place.lists = coffeeloover_lists
            coffeeplaces_search_results_with_user_information.append(coffee_place)
        return render(request, "cofloapp/search_place.html", {"coffeeplaces":
coffeeplaces_search_results_with_user_information})

def search_coffeeplace_ratings(request):
    coffee_place_ratings = Coffeeplacering.objects.all()
    slug_of_coffeeplace = self.request.GET.get('slug')
    if slug_of_coffeeplace:
        coffee_place_ratings = coffee_place_ratings.filter(coffeeplace__place_slug =
slug_of_coffeeplace)
        min_rating_of_coffeeplace = float(self.request.GET.get('minrating'))
        if not min_rating_of_coffeeplace or min_rating_of_coffeeplace < 0 or
min_rating_of_coffeeplace > 5:
            min_rating_of_coffeeplace = 0
        coffee_place_ratings = coffee_place_ratings.filter(total_rating__gte =
min_rating_of_coffeeplace)
        max_rating_of_coffeeplace = float(self.request.GET.get('maxrating'))
        if not max_rating_of_coffeeplace or max_rating_of_coffeeplace > 5 or
max_rating_of_coffeeplace < 0:
            max_rating_of_coffeeplace = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(total_rating__lte =
max_rating_of_coffeeplace)
        min_coffee_rating = float(self.request.GET.get('mincoffee'))
        if not min_coffee_rating or min_coffee_rating < 0 or min_coffee_rating > 5:
            min_coffee_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(coffee_rating__gte = min_coffee_rating)
        max_coffee_rating = float(self.request.GET.get('maxcoffee'))
        if not max_coffee_rating or max_coffee_rating < 0 or max_coffee_rating > 5:
            max_coffee_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(coffee_rating__lte = max_coffee_rating)
        min_vibe_rating = float(self.request.GET.get('minvibe'))
        if not min_vibe_rating or min_vibe_rating < 0 or min_vibe_rating > 5:
            min_vibe_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(vibe_rating__gte = min_vibe_rating)
        max_vibe_rating = float(self.request.GET.get('maxvibe'))
        if not max_vibe_rating or max_vibe_rating < 0 or max_vibe_rating > 5:
            max_vibe_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(vibe_rating__lte = max_vibe_rating)
        min_service_rating = float(self.request.GET.get('minservice'))
        if not min_service_rating or min_service_rating < 0 or min_service_rating > 5:
            min_service_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(service_rating__gte =
min_service_rating)
        max_service_rating = float(self.request.GET.get('maxservice'))
        if not max_service_rating or max_service_rating < 0 or max_service_rating > 5:

```

```

        max_service_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(service_rating__lte =
max_service_rating)
        min_seating_rating = float(self.request.GET.get('minseating'))
        if not min_seating_rating or min_seating_rating < 0 or min_seating_rating > 5:
            min_seating_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(seating_rating__gte =
min_seating_rating)
        max_seating_rating = float(self.request.GET.get('maxseating'))
        if not max_seating_rating or max_seating_rating < 0 or max_seating_rating > 5:
            max_seating_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(seating_rating__lte =
max_seating_rating)
        min_price_rating = float(self.request.GET.get('minprice'))
        if not min_price_rating or min_price_rating < 0 or min_price_rating > 5:
            min_price_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(price_rating__gte = min_price_rating)
        max_price_rating = float(self.request.GET.get('maxprice'))
        if not max_price_rating or max_price_rating < 0 or max_price_rating > 5:
            max_price_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(price_rating__lte = max_price_rating)
        min_parking_rating = float(self.request.GET.get('minparking'))
        if not min_parking_rating or min_parking_rating < 0 or min_parking_rating > 5:
            min_parking_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(parking_rating__gte =
min_parking_rating)
        max_parking_rating = float(self.request.GET.get('maxparking'))
        if not max_parking_rating or max_parking_rating < 0 or max_parking_rating > 5:
            max_parking_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(parking_rating__lte =
max_parking_rating)
        min_location_rating = float(self.request.GET.get('minlocation'))
        if not min_location_rating or min_location_rating < 0 or min_location_rating > 5:
            min_location_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(location_rating__gte =
min_location_rating)
        max_location_rating = float(self.request.GET.get('maxlocation'))
        if not max_location_rating or max_location_rating < 0 or max_location_rating > 5:
            max_location_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(location_rating__lte =
max_location_rating)
        min_internet_rating = float(self.request.GET.get('mininternet'))
        if not min_internet_rating or min_internet_rating < 0 or min_internet_rating > 5:
            min_internet_rating = 0
        coffee_place_ratings = coffee_place_ratings.filter(internet_rating__gte =
min_internet_rating)
        max_internet_rating = float(self.request.GET.get('maxinternet'))
        if not max_internet_rating or max_internet_rating < 0 or max_internet_rating > 5:
            max_internet_rating = 5.0
        coffee_place_ratings = coffee_place_ratings.filter(internet_rating__lte =
max_internet_rating)
        rating_text = self.request.GET.get('text')
        if rating_text:
            coffee_place_ratings = coffee_place_ratings.filter(text_presentation__icontains =
rating_text)
        rating_coffeeover = self.request.GET.get('author')

```

```

if rating_coffeeloover:
    coffee_lover = Coffeeloover.objects.get(user__username = rating_coffeeloover)
    coffee_place_ratings = coffee_place_ratings.filter(creator_coffeeloover =
coffee_lover).prefetch_related('creator_coffeeloover__follows')
    coffee_place_ratings = coffee_place_ratings.order_by('-created_time')
    period_date_str = self.request.GET.get('date')
    period_date = datetime.datetime.strptime(period_date_str, '%m.%d.%Y')
    period_date.hour = 0
    period_date.minute = 0
    more_or_less = int(self.request.GET.get('way'))
    if more_or_less == 0:
        coffee_place_ratings = coffee_place_ratings.filter(created_time__lte = period_date)
    else:
        coffee_place_ratings = coffee_place_ratings.filter(created_time__gte = period_date)
    reactions = self.request.GET.get('reactions').split('.')
    reaction_conditions = []
    for reaction_pair in reactions:
        reaction_pair = reaction_pair.split('-')
        reaction_name = reaction_pair[0]
        reaction_number = int(reaction_pair[1])
        reaction_type = Reactiontype.objects.get(name = reaction_name)
        reaction_condition = []
        reaction_condition.append(reaction_type)
        reaction_condition.append(reaction_number)
        reaction_conditions.append(reaction_condition)
    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
        coffee_place_ratings_search_results = []
        for coffee_place_rating in coffee_place_ratings:
            add_to_results = True
            coffee_place_rating.reactions = {}
            for reaction_condition in reaction_conditions:
                reactions = Coffeeplaceratingreaction.objects.filter(coffeeplacerating =
coffee_place_rating, reactiontype = reaction_condition[0]).count()
                if reactions < reaction_condition[1]:
                    add_to_results = False
                coffee_place_rating.reactions[reaction_condition[0]] = reaction_condition[1]
            coffee_lover = coffee_place_rating.creator_coffeeloover
            coffee_place_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()
            reference_made = False
            coffee_place_rating.follow_number = len(coffee_lover.follows)
            coffee_place_rating.follow = False
            if request.user.is_authenticated:
                if coffeeloover in coffee_lover.follows:
                    coffee_place_rating.follow = True
            try:
                created_reference = Reference.objects.get(creator_coffeeloover = coffeeloover)
                reference_made = True
            except Reference.DoesNotExist:
                reference_made = False
            coffee_place_rating.referenced = reference_made
            if add_to_results:
                coffee_place_ratings_search_results.append(coffee_place_rating)
        all_ratings = Coffeeplaceratingreaction.objects.filter(coffeeplace__place_slug =
slug_of_coffeeplace)

```

```

ratings_5 = all_ratings.filter(total_rating__gt = 4.0).count()
ratings_4 = all_ratings.filter(total_rating__lte = 4.0, total_rating__gt = 3.0).count()
ratings_3 = all_ratings.filter(total_rating__lte = 3.0, total_rating__gt = 2.0).count()
ratings_2 = all_ratings.filter(total_rating__lte = 2.0).count()
ratings_count = [ratings_5, ratings_4, ratings_3, ratings_2]
return render(request, "cofloapp/search_place_ratings.html", {"ratings":
coffee_place_ratings_search_results, "levels": ratings_count })

def search_coffeeplace_drink_ratings(request):
    all_coffee_place_drink_ratings = Coffeeplacedrinkrating.objects.all()
    slug_of_coffeeplace = self.request.GET.get('slug')
    if slug_of_coffeeplace:
        all_coffee_place_drink_ratings =
all_coffee_place_drink_ratings.filter(coffeeplacedrink__coffeeplace__place_slug =
slug_of_coffeeplace)
        name_of_drink = self.request.GET.get('drink')
        if name_of_drink:
            coffee_drink = Drink.objects.get(name = name_of_drink)
            all_coffee_place_drink_ratings =
all_coffee_place_drink_ratings.filter(coffeeplacedrink__drink = coffee_drink)
            coffee_place_drink_ratings = all_coffee_place_drink_ratings
            min_rating_of_placedrink = float(self.request.GET.get('minrating'))
            if not min_rating_of_placedrink or min_rating_of_placedrink < 0 or
min_rating_of_placedrink > 5:
                min_rating_of_placedrink = 0
            coffee_place_drink_ratings = coffee_place_drink_ratings.filter(total_rating__gte =
min_rating_of_placedrink)

            max_rating_of_placedrink = float(self.request.GET.get('maxrating'))
            if not max_rating_of_placedrink or max_rating_of_placedrink > 5 or
max_rating_of_placedrink < 0:
                max_rating_of_placedrink = 5.0
            coffee_place_drink_ratings = coffee_place_drink_ratings.filter(total_rating__lte =
max_rating_of_placedrink)
            rating_text = self.request.GET.get('text')
            if rating_text:
                coffee_place_drink_ratings =
coffee_place_drink_ratings.filter(text_presentation__icontains = rating_text)
            rating_coffee_lover = self.request.GET.get('author')
            if rating_coffee_lover:
                coffee_lover = Coffee_lover.objects.get(user__username = rating_coffee_lover)
                coffee_place_drink_ratings = coffee_place_drink_ratings.filter(creator_coffee_lover =
coffee_lover)
            coffee_place_drink_ratings =
coffee_place_drink_ratings.select_related('creator_coffee_lover').prefetch_related('creator_coffee_love
r__follows')
            coffee_place_drink_ratings = coffee_place_drink_ratings.order_by('-created_time')
            period_date_str = self.request.GET.get('date')
            period_date = datetime.datetime.strptime(period_date_str, '%m.%d.%Y')
            period_date.hour = 0
            period_date.minute = 0
            more_or_less = int(self.request.GET.get('way'))
            if more_or_less == 0:
                coffee_place_drink_ratings = coffee_place_drink_ratings.filter(created_time__lte =
period_date)
            else:

```

```

        coffee_place_drink_ratings = coffee_place_drink_ratings.filter(created_time__gte =
period_date)
    reactions = self.request.GET.get('reactions').split('.')
    reaction_conditions = []
    for reaction_pair in reactions:
        reaction_pair = reaction_pair.split('-')
        reaction_name = reaction_pair[0]
        reaction_number = int(reaction_pair[1])
        reaction_type = Reactiontype.objects.get(name = reaction_name)
        reaction_condition = []
        reaction_condition.append(reaction_type)
        reaction_condition.append(reaction_number)
        reaction_conditions.append(reaction_condition)
    if request.user.is_authenticated:
        coffeelover = Coffeelover.objects.get(user = request.user)
        coffee_place_drink_ratings_search_results = []
        for coffee_place_drink_rating in coffee_place_drink_ratings:
            add_to_results = True
            coffee_place_drink_rating.reactions = {}
            for reaction_condition in reaction_conditions:
                reactions = Coffeeplacedrinkratingreaction.objects.filter(coffeeplacedrinkrating =
coffee_place_drink_rating, reactiontype = reaction_condition[0]).count()
                if reactions < reaction_condition[1]:
                    add_to_results = False
                    coffee_place_drink_rating.reactions[reaction_condition[0]] = reaction_condition[1]
            coffee_lover = coffee_place_drink_rating.creator_coffeelover
            coffee_place_drink_rating.reference_number =
Reference.objects.filter(followed_coffeelover = coffee_lover).count()
            coffee_place_drink_rating.follow_number = len(coffee_lover.follows)
            reference_made = False
            coffee_place_drink_rating.follow = False
            if request.user.is_authenticated:
                if coffeelover in coffee_lover.follows:
                    coffee_place_drink_rating.follow = True
            try:
                created_reference = Reference.objects.get(creator_coffeelover = coffeelover)
                reference_made = True
            except Reference.DoesNotExist:
                reference_made = False
            coffee_place_drink_rating.referenced = reference_made
            if add_to_results:
                coffee_place_drink_ratings_search_results.append(coffee_place_drink_rating)
        ratings_5 = all_coffee_place_drink_ratings.filter(rating__gt = 4.0).count()
        ratings_4 = all_coffee_place_drink_ratings.filter(rating__lte = 4.0, total_rating__gt =
3.0).count()
        ratings_3 = all_coffee_place_drink_ratings.filter(rating__lte = 3.0, total_rating__gt =
2.0).count()
        ratings_2 = all_coffee_place_drink_ratings.filter(rating__lte = 2.0).count()
        ratings_count = [ratings_5, ratings_4, ratings_3, ratings_2]
        return render(request, "cofloapp/search_place_drink_ratings.html", {"ratings":
coffee_place_drink_ratings_search_results, "levels": ratings_count })

def coffee_lover_place_ratings(request, coffeelover_username):
    coffeeplac ratings = []
    coffeelover_found = True

```

```

if coffeeloader_username:
    coffeeloader = Coffeeloader.objects.get(user = coffeeloader_username)
elif request.user.is_authenticated:
    coffeeloader = Coffeeloader.objects.get(user = request.user)
else:
    coffeeloader_found = False
if coffeeloader_found:
    coffeeplaceratings = Coffeeplacerating.objects.filter(creator_coffeeloader = coffeeloader)
    coffeeplaceratings = coffeeplaceratings.order_by('created_time')
    reaction_types = Reactiontype.objects.all().values()
    for coffee_place_rating in coffeeplaceratings:
        coffee_place_rating.reactions = {}
        for reaction_condition in reaction_types:
            reactions = Coffeeplaceratingreaction.objects.filter(coffeeplacerating =
coffee_place_rating, reactiontype = reaction_condition).count()
            coffee_place_rating.reactions[reaction_condition] = reactions
        ratings_5 = coffeeplaceratings.filter(rating__gt = 4.0).count()
        ratings_4 = coffeeplaceratings.filter(rating__lte = 4.0, total_rating__gt = 3.0).count()
        ratings_3 = coffeeplaceratings.filter(rating__lte = 3.0, total_rating__gt = 2.0).count()
        ratings_2 = coffeeplaceratings.filter(rating__lte = 2.0).count()
        ratings_count = [ratings_5, ratings_4, ratings_3, ratings_2]
        return render(request, "cofloapp/coffeeloader_ratings.html", {"ratings":
coffeeplaceratings, "levels": ratings_count, "user": coffeeloader })

def coffee_lover_drink_ratings(request, coffeeloader_username):
    coffeeplacedrinkratings = []
    coffeeloader_found = True
    if coffeeloader_username:
        coffeeloader = Coffeeloader.objects.get(user = coffeeloader_username)
    elif request.user.is_authenticated:
        coffeeloader = Coffeeloader.objects.get(user = request.user)
    else:
        coffeeloader_found = False
    if coffeeloader_found:
        coffeeplacedrinkratings = Coffeeplacedrinkrating.objects.filter(creator_coffeeloader =
coffeeloader).order_by('-created_time')
        reaction_types = Reactiontype.objects.all().values()
        for coffee_drink_rating in coffeeplacedrinkratings:
            coffee_drink_rating.reactions = {}
            for reaction_condition in reaction_types:
                reactions = Coffeeplacedrinkratingreaction.objects.filter(coffeeplacedrinkrating =
coffee_drink_rating, reactiontype = reaction_condition).count()
                coffee_drink_rating.reactions[reaction_condition] = reactions
            ratings_5 = coffeeplacedrinkratings.filter(rating__gt = 4.0).count()
            ratings_4 = coffeeplacedrinkratings.filter(rating__lte = 4.0, total_rating__gt = 3.0).count()
            ratings_3 = coffeeplacedrinkratings.filter(rating__lte = 3.0, total_rating__gt = 2.0).count()
            ratings_2 = coffeeplacedrinkratings.filter(rating__lte = 2.0).count()
            ratings_count = [ratings_5, ratings_4, ratings_3, ratings_2]
            return render(request, "cofloapp/coffeeloader_drink_ratings.html", {"ratings":
coffeeplacedrinkratings, "levels": ratings_count, "user": coffeeloader })

def coffeeplace_ratings(request, coffeeplace_slug):
    coffee_place_ratings = Coffeeplacerating.objects.all()
    coffee_place_ratings = coffee_place_ratings.filter(coffeeplace_place_slug =
coffeeplace_slug)
    if request.user.is_authenticated:

```

```

        coffeeloover = Coffeeloover.objects.get(user = request.user)
        reaction_types = Reactiontype.objects.all().values()
        coffee_place_ratings =
coffee_place_ratings.select_related('creator_coffeeloover').order_by('-created_time')
        for coffee_place_rating in coffee_place_ratings:
            coffee_place_rating.reactions = {}
            for reaction_condition in reaction_types:
                reactions = Coffeeplaceringreaction.objects.filter(coffeeplacering =
coffee_place_rating, reactiontype = reaction_condition).count()
                coffee_place_rating.reactions[reaction_condition] = reactions
                coffee_lover = coffee_place_rating.creator_coffeeloover
                coffee_place_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()
                reference_made = False
                coffee_place_rating.follow_number = len(coffee_lover.follows)
                coffee_place_rating.follow = False
                if request.user.is_authenticated:
                    if coffeeloover in coffee_lover.follows:
                        coffee_place_rating.follow = True
                try:
                    created_reference = Reference.objects.get(creator_coffeeloover = coffeeloover)
                    reference_made = True
                except Reference.DoesNotExist:
                    reference_made = False
                coffee_place_rating.referenced = reference_made
                ratings_5 = coffee_place_ratings.filter(total_rating__gt = 4.0).count()
                ratings_4 = coffee_place_ratings.filter(total_rating__lte = 4.0, total_rating__gt =
3.0).count()
                ratings_3 = coffee_place_ratings.filter(total_rating__lte = 3.0, total_rating__gt =
2.0).count()
                ratings_2 = coffee_place_ratings.filter(total_rating__lte = 2.0).count()
                ratings_count = [ratings_5, ratings_4, ratings_3, ratings_2]
                return render(request, "cofloapp/place_ratings.html", {"ratings": coffee_place_ratings,
"levels": ratings_count })

```

```

def drink_ratings(request, coffeeplace_slug, drink_name = None):
    coffee_drink_ratings = Coffeeplacedrinkrating.objects.all()
    coffee_drink_ratings =
coffee_drink_ratings.filter(coffeeplacedrink__coffeeplace__place_slug = coffeeplace_slug)
    if drink_name:
        coffee_drink_ratings = coffee_drink_ratings.filter(coffeeplacedrink__drink_name =
drink_name)
    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
        reaction_types = Reactiontype.objects.all().values()
        coffee_drink_ratings =
coffee_drink_ratings.select_related('creator_coffeeloover').order_by('-created_time')
        for coffee_drink_rating in coffee_drink_ratings:
            coffee_drink_rating.reactions = {}
            for reaction_condition in reaction_types:
                reactions = Coffeeplacedrinkratingreaction.objects.filter(coffeeplacedrinkrating =
coffee_drink_rating, reactiontype = reaction_condition).count()
                coffee_drink_rating.reactions[reaction_condition] = reactions
                coffee_lover = coffee_drink_rating.creator_coffeeloover
                coffee_drink_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()

```

```

reference_made = False
coffee_drink_rating.follow_number = len(coffee_lover.follows)
coffee_drink_rating.follow = False
if request.user.is_authenticated:
    if coffeelover in coffee_lover.follows:
        coffee_drink_rating.follow = True
    try:
        created_reference = Reference.objects.get(creator_coffeelover = coffeelover)
        reference_made = True
    except Reference.DoesNotExist:
        reference_made = False
    coffee_drink_rating.referenced = reference_made
    ratings_5 = coffee_drink_ratings.filter(total_rating__gt = 4.0).count()
    ratings_4 = coffee_drink_ratings.filter(total_rating__lte = 4.0, total_rating__gt =
3.0).count()
    ratings_3 = coffee_drink_ratings.filter(total_rating__lte = 3.0, total_rating__gt =
2.0).count()
    ratings_2 = coffee_drink_ratings.filter(total_rating__lte = 2.0).count()
    ratings_count = [ratings_5, ratings_4, ratings_3, ratings_2]
    return render(request, "cofloapp/drink_ratings.html", {"ratings": coffee_drink_ratings,
"levels": ratings_count })

def coffee_place_photos(request, coffeeplace_slug):
    coffee_place_photos = Coffeeplacephoto.objects.all()
    coffee_place_photos = coffee_place_photos.filter(coffeeplace__place_slug =
coffeeplace_slug).order_by('-created_date')
    return render(request, "cofloapp/place_photos.html", {"photos": coffee_place_photos })

def coffee_place_drink_photos(request, drink_name, coffeeplace_slug):
    coffee_drink_photos = Coffeeplacedrinkphoto.objects.all()
    coffee_drink_photos =
coffee_drink_photos.filter(coffeeplacedrink__coffeeplace__place_slug = coffeeplace_slug)
    coffee_drink_photos = coffee_drink_photos.filter(coffeeplacedrink__drink_name =
drink_name)
    coffee_drink_photos = coffee_drink_photos.order_by('-created_date')
    return render(request, "cofloapp/drink_photos.html", {"photos": coffee_drink_photos })

def coffee_drink_photos(request, drink_name):
    coffee_drink_photos = Coffeeplacedrinkphoto.objects.all()
    coffee_drink_photos = coffee_drink_photos.filter(coffeeplacedrink__drink_name =
drink_name)
    coffee_drink_photos = coffee_drink_photos.order_by('-created_date')
    return render(request, "cofloapp/all_drink_photos.html", {"photos": coffee_drink_photos
})

def coffee_lover_drink_photos(request, coffeelover_username = None):
    coffee_drink_photos = []
    coffeelover_found = True
    if coffeelover_username:
        coffeelover = Coffeelover.objects.get(user = coffeelover_username)
    elif request.user.is_authenticated:
        coffeelover = Coffeelover.objects.get(user = request.user)
    else:
        coffeelover_found = False
    if coffeelover_found:
        coffee_drink_photos = Coffeeplacedrinkphoto.objects.all()

```

```

        coffee_drink_photos = coffee_drink_photos.filter(creator_coffeeloover__username =
coffeeloover_username)
        coffee_drink_photos = coffee_drink_photos.order_by('-created_date')
        return render(request, "cofloapp/cofee_lover_drink_photos.html", {"photos":
coffee_drink_photos, "user": coffeeloover })

def coffee_lover_place_photos(request, coffeeloover_username = None):
    coffee_place_photos = []
    coffeeloover_found = True
    if coffeeloover_username:
        coffeeloover = Coffeeloover.objects.get(user = coffeeloover_username)
    elif request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
    else:
        coffeeloover_found = False
    if coffeeloover_found:
        coffee_place_photos = Coffeeplacephoto.objects.all()
        coffee_place_photos = coffee_place_photos.filter(creator_coffeeloover__username =
coffeeloover_username)
        coffee_place_photos = coffee_place_photos.order_by('-created_date')
        return render(request, "cofloapp/cofee_lover_place_photos.html", {"photos":
coffee_place_photos, "user": coffeeloover })

def coffee_lover_follows(request, coffeeloover_username = None):
    coffee_lover_followers = []
    coffeeloover_found = True
    if coffeeloover_username:
        coffeeloover = Coffeeloover.objects.get(user =
coffeeloover_username).prefetch_related('follows')
    elif request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user).prefetch_related('follows')
    else:
        coffeeloover_found = False
    followed = False
    if request.user.is_authenticated and Coffeeloover.objects.get(user = request.user) in
coffeeloover.follows:
        followed = True
    return render(request, "cofloapp/follows.html", {"user": coffeeloover, "to_follow":
followed })

def new_ratings(request):
    coffee_place_ratings = Coffeeplacering.objects.filter(creator_coffeeloover__public =
True).select_related('creator_coffeeloover').order_by('-created_time').values()[:100]
    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
    reaction_types = Reactiontype.objects.all().values()
    for coffee_place_rating in coffee_place_ratings:
        coffee_place_rating.reactions = {}
        for reaction_condition in reaction_types:
            reactions = Coffeeplaceringreaction.objects.filter(coffeeplacering =
coffee_place_rating, reactiontype = reaction_condition).count()
            coffee_place_rating.reactions[reaction_condition] = reactions
        coffee_lover = coffee_place_rating.creator_coffeeloover
        coffee_place_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()
        reference_made = False

```

```

coffee_place_rating.follow_number = len(coffee_lover.follows)
coffee_place_rating.follow = False
if request.user.is_authenticated:
    if coffeeloader in coffee_lover.follows:
        coffee_place_rating.follow = True
    try:
        created_reference = Reference.objects.get(creator_coffeeloader = coffeeloader)
        reference_made = True
    except Reference.DoesNotExist:
        reference_made = False
    coffee_place_rating.referenced = reference_made
    coffee_drink_ratings = Coffeeplacedrinkrating.objects.filter(creator_coffeeloader__public
= True).select_related('creator_coffeeloader').order_by('-created_time').values()[:100]
    for coffee_drink_rating in coffee_drink_ratings:
        coffee_drink_rating.reactions = {}
        for reaction_condition in reaction_types:
            reactions = Coffeeplacedrinkratingreaction.objects.filter(coffeeplacedrinkrating =
coffee_drink_rating, reactiontype = reaction_condition).count()
            coffee_drink_rating.reactions[reaction_condition] = reactions
        coffee_lover = coffee_drink_rating.creator_coffeeloader
        coffee_drink_rating.reference_number = Reference.objects.filter(followed_coffeeloader
= coffee_lover).count()
        reference_made = False
        coffee_drink_rating.follow_number = len(coffee_lover.follows)
        coffee_drink_rating.follow = False
        if request.user.is_authenticated:
            if coffeeloader in coffee_lover.follows:
                coffee_drink_rating.follow = True
            try:
                created_reference = Reference.objects.get(creator_coffeeloader = coffeeloader)
                reference_made = True
            except Reference.DoesNotExist:
                reference_made = False
            coffee_drink_rating.referenced = reference_made
        return render(request, "cofloapp/new_ratings.html", {"place_ratings":
coffee_place_ratings, "drink_ratings": coffee_drink_ratings })

def place_drinks(request, coffeeplace_slug):
    place_drinks = Coffeeplacedrink.filter(coffeeplacedrink__coffeeplace__place_slug =
coffeeplace_slug).order_by('-average_rating')
    return render(request, "cofloapp/place_drinks.html", {"drinks": place_drinks })

def place_lists(request):
    if request.user.is_authenticated:
        coffeeloader = Coffeeloader.objects.get(user = request.user)
        coffee_place_lists = Coffeeplacelist.filter(creator_coffeeloader = coffeeloader).order_by('-
created_time')
    return render(request, "cofloapp/place_lists.html", {"lists": coffee_place_lists })

def drink_lists(request):
    if request.user.is_authenticated:
        coffeeloader = Coffeeloader.objects.get(user = request.user)
        coffee_drink_lists = Coffeeplacedrinklist.filter(creator_coffeeloader =
coffeeloader).order_by('-created_time')
    return render(request, "cofloapp/drink_lists.html", {"lists": coffee_drink_lists })

```

```

def coffee_lover_references(request):
    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
        references = Reference.filter(creator_coffeeloover =
coffeeloover).select_related('followed_coffeeloover').order_by('-created_time')
        return render(request, "cofloapp/references.html", {"references": references })

def coffee_lover_businessowners(request):
    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
        businesses = Businessowner.filter(coffeeloover =
coffeeloover).select_related('coffeeplace').order_by('-id')
        return render(request, "cofloapp/businesses.html", {"businesses": businesses })

def new_follow_ratings(request):
    if request.user.is_authenticated:
        follows = Coffeeloover.objects.get(user =
request.user).prefetch_related('follows').values_list('follows')
        coffee_place_ratings = Coffeeplacerating.objects.filter(creator_coffeeloover__in =
follows).select_related('creator_coffeeloover').order_by('-created_time')
        reaction_types = Reactiontype.objects.all().values()
        for coffee_place_rating in coffee_place_ratings:
            coffee_place_rating.reactions = {}
            for reaction_condition in reaction_types:
                reactions = Coffeeplaceratingreaction.objects.filter(coffeeplacerating =
coffee_place_rating, reactiontype = reaction_condition).count()
                coffee_place_rating.reactions[reaction_condition] = reactions
            coffee_lover = coffee_place_rating.creator_coffeeloover
            coffee_place_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()
            reference_made = False
            coffee_place_rating.follow_number = len(coffee_lover.follows)
            coffee_place_rating.follow = False
            if request.user.is_authenticated:
                if coffeeloover in coffee_lover.follows:
                    coffee_place_rating.follow = True
            try:
                created_reference = Reference.objects.get(creator_coffeeloover = coffeeloover)
                reference_made = True
            except Reference.DoesNotExist:
                reference_made = False
            coffee_place_rating.referenced = reference_made
            coffee_drink_ratings = Coffeeplacedrinkrating.objects..filter(creator_coffeeloover__in =
follows).select_related('creator_coffeeloover').order_by('-created_time').values()
            for coffee_drink_rating in coffee_drink_ratings:
                coffee_drink_rating.reactions = {}
                for reaction_condition in reaction_types:
                    reactions = Coffeeplacedrinkratingreaction.objects.filter(coffeeplacedrinkrating =
coffee_drink_rating, reactiontype = reaction_condition).count()
                    coffee_drink_rating.reactions[reaction_condition] = reactions
                coffee_lover = coffee_drink_rating.creator_coffeeloover
                coffee_drink_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()
                reference_made = False
                coffee_drink_rating.follow_number = len(coffee_lover.follows)
                coffee_drink_rating.follow = False

```



```

return render(request, "cofloapp/drink_list.html", {"list": coffee_drink_list })

def coffee_place_page(request, coffee_place_slug):
    coffee_place = Coffeeplace.objects.get(place_slug =
coffee_place_slug).select_related('coffeeplacetype').prefetch_related('timetables').prefetch_related('
coffeeplacephotos')
    coffee_place_ratings = Coffeeplacering.objects.filter(coffeeplace =
coffee_place).select_related('creator_coffeeloover').order_by('-created_time')[:5]
    recommendations = Recommendation.objects.filter(coffeeplace_source =
coffee_place).select_related("coffeeplace_reference").values_list('coffeeplace_reference')
    if request.user.is_authenticated:
        coffeeloover = Coffeeloover.objects.get(user = request.user)
        recommendations_with_user_information = []
        for coffee_place in recommendations:
            coffeeloover_ratings = Coffeeplacering.objects.filter(coffeeplace = coffee_place,
creator_coffeeloover = coffeeloover).order_by('-created_time')
            coffee_place.ratings_count = coffeeloover_ratings.count()
            coffee_place.last_rating = coffeeloover_ratings.values()[0]
            coffeeloover_lists = Coffeeplacelistmember.objects.filter(coffeeplace = coffee_place,
placelist_creator_coffeeloover =
coffeeloover).select_related('coffeeplacelist').values_list('coffeeplacelist')
            coffee_place.lists = coffeeloover_lists
            recommendations_with_user_information.append(coffee_place)
        reaction_types = Reactiontype.objects.all().values()
        for coffee_place_rating in coffee_place_ratings:
            coffee_place_rating.reactions = {}
            for reaction_condition in reaction_types:
                reactions = Coffeeplaceringreaction.objects.filter(coffeeplacering =
coffee_place_rating, reactiontype = reaction_condition).count()
                coffee_place_rating.reactions[reaction_condition] = reactions
            coffee_lover = coffee_place_rating.creator_coffeeloover
            coffee_place_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()
            reference_made = False
            coffee_place_rating.follow_number = len(coffee_lover.follows)
            coffee_place_rating.follow = False
            if request.user.is_authenticated:
                if coffeeloover in coffee_lover.follows:
                    coffee_place_rating.follow = True
            try:
                created_reference = Reference.objects.get(creator_coffeeloover = coffeeloover)
                reference_made = True
            except Reference.DoesNotExist:
                reference_made = False
            coffee_place_rating.referenced = reference_made
            coffee_drink_ratings = Coffeeplacedrinkrating.objects.filter(coffeeplace =
coffee_place).select_related('creator_coffeeloover').order_by('-created_time')[:5]
            for coffee_drink_rating in coffee_drink_ratings:
                coffee_drink_rating.reactions = {}
                for reaction_condition in reaction_types:
                    reactions = Coffeeplacedrinkratingreaction.objects.filter(coffeeplacedrinkrating =
coffee_drink_rating, reactiontype = reaction_condition).count()
                    coffee_drink_rating.reactions[reaction_condition] = reactions
                    coffee_lover = coffee_drink_rating.creator_coffeeloover
                    coffee_drink_rating.reference_number = Reference.objects.filter(followed_coffeeloover
= coffee_lover).count()

```

```

reference_made = False
coffee_drink_rating.follow_number = len(coffee_lover.follows)
coffee_drink_rating.follow = False
if request.user.is_authenticated:
    if coffeelover in coffee_lover.follows:
        coffee_drink_rating.follow = True
    try:
        created_reference = Reference.objects.get(creator_coffeelover = coffeelover)
        reference_made = True
    except Reference.DoesNotExist:
        reference_made = False
    coffee_drink_rating.referenced = reference_made
    place_drinks = Coffeepplacedrink.filter(coffeeplacedrink__coffeeplace__place_slug =
coffee_place_slug).order_by('-average_rating')[:5]
    return render(request, "cofloapp/coffee_place.html", {"place": coffee_place,
"plac ratings": coffee_place_ratings, "drink ratings": coffee_drink_ratings, "drinks": place_drinks,
"recommendations": recommendations_with_user_information })

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
кафедра програмних засобів

Тема дипломної кваліфікаційної роботи магістра
«Дослідження та програмна реалізація методів
рекомендування закладів для любителів кави.
Research and Software Implementation of Methods for
Recommending Places to Coffee Lovers»

Виконавець:

Федір ШМАЛЬКО

Керівник: к.т.н., доцент

Валерій ЛЬОВКІН

2023

Рисунок В.1 – Слайд № 1 (Титульний слайд)

Об'єкт, предмет, мета, завдання роботи

Об'єкт роботи – процес пошуку і рекомендування закладів для любителів кави.

Предмет роботи – методи рекомендування закладів для любителів кави.

Мета роботи – розробити програмне забезпечення пошуку і рекомендування закладів для любителів кави для інформування користувачів про релевантні заклади.

Завдання роботи:

- проаналізувати програмне забезпечення, яке використовується для пошуку закладів для любителів кави;
- розробити метод рекомендування закладів для любителів кави і виконати експериментальне дослідження;
- виконати проектування програми пошуку закладів для любителів кави;
- реалізувати програму пошуку закладів для любителів кави.

Рисунок В.2 – Слайд № 2 (Об'єкт, предмет, мета, завдання роботи)

Аналіз програмного забезпечення, яке використовується для пошуку закладів для любителів кави

Критерій	Yelp	FourSquare	Tripadvisor	Представлена програма
Інформація про заклади з продажу кави	Так	Так	Так	Так
Інформація про інші місця, де продаються напої з кави	Ні	Ні	Ні	Так
Інформація про напої з кави	Так	Ні	Ні	Так
Пошук закладів	Так	Так	Так	Розширений (наприклад, за напоями)
Переліки закладів	Так	Так	Ні	Так
Переліки напоїв з кави	Ні	Ні	Ні	Так
Підписки на користувачів	Так	Так	Ні	Так
Рейтинги закладів	Так	Так	Так	Так
Рейтинги напоїв у закладах	Так	Ні	Ні	Так
Реакції на рейтинги	Так	Так	Тільки вподобання	Так, розширені
Рекомендування закладів з продажу кави	Тільки як реклама закладів загалом	Під заклади	Відсутні, тільки найкращі	Під заклади

3

Рисунок В.3 – Слайд № 3 (Аналіз програмного забезпечення, яке використовується для пошуку закладів для любителів кави)

Метод рекомендування закладів для любителів кави

- визначення множини доступних закладів;
- виконання автоматизованого оцінювання рейтингів у випадку відсутності явних оцінок з тексту на основі сентимент-аналізу:

$$r_j^i = \begin{cases} \frac{(s_j^i - o^i)(sc^{\max} - r_{tot}^i)}{\max_i s_j^i - o^i} + r_{tot}^i, & \text{якщо } s_j^i \geq o^i; \\ r_{tot}^i - \frac{r_{tot}^i(o^i - s_j^i)}{o^i - \min_i s_j^i}, & \text{якщо } s_j^i < o^i, \end{cases}$$

- визначення для кожного закладу середніх значень за кожним критерієм;
- виконання спектрального кластерного аналізу.

4

Рисунок В.4 – Слайд № 4 (Метод рекомендування закладів для любителів кави)

Аналіз фреймворків веброзробки для створення програмного забезпечення рекомендаування закладів для любителів кави

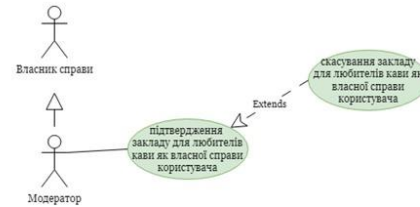
Критерій	Django	Node.js
Використовувана мова програмування	Python	JavaScript
Рівень безпеки даних і засобів	Високий рівень вбудованого захисту	Потребує додаткової роботи, ускладняється при збільшенні джерел даних
Час розробки співставних рішень	Менший	Вищий
Масштабованість	Висока	Висока
Стабільність фреймворків машинного навчання	Вища	Нижча

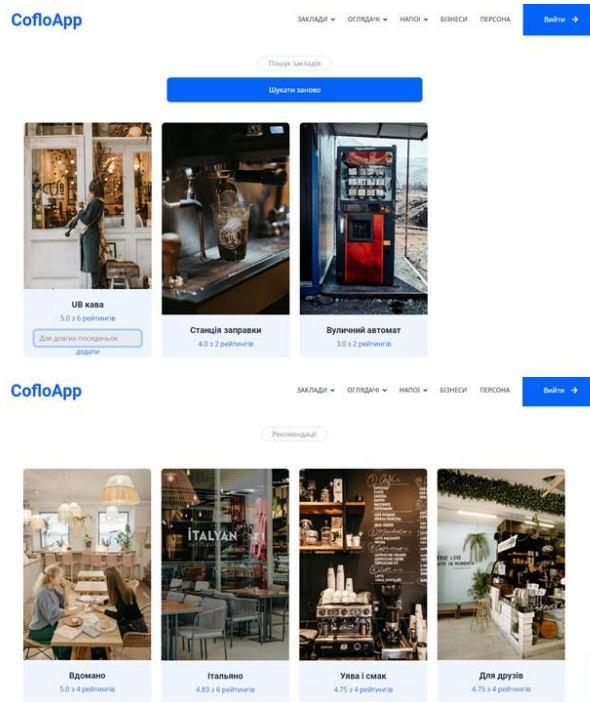
5

Рисунок В.5 – Слайд № 5 (Аналіз фреймворків веброзробки для створення програмного забезпечення рекомендаування закладів для любителів кави)

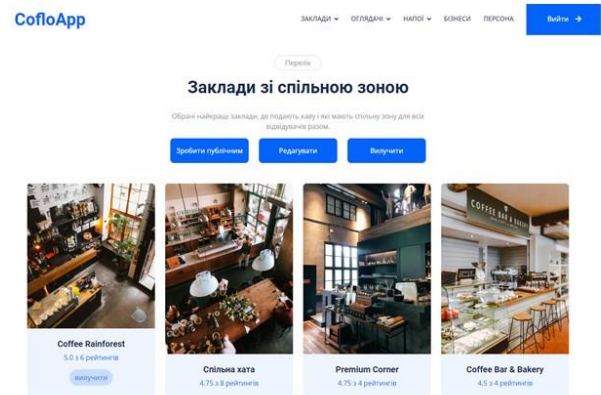


Рисунок В.6 – Слайд № 6 (Діаграми прецедентів)

[illegible]



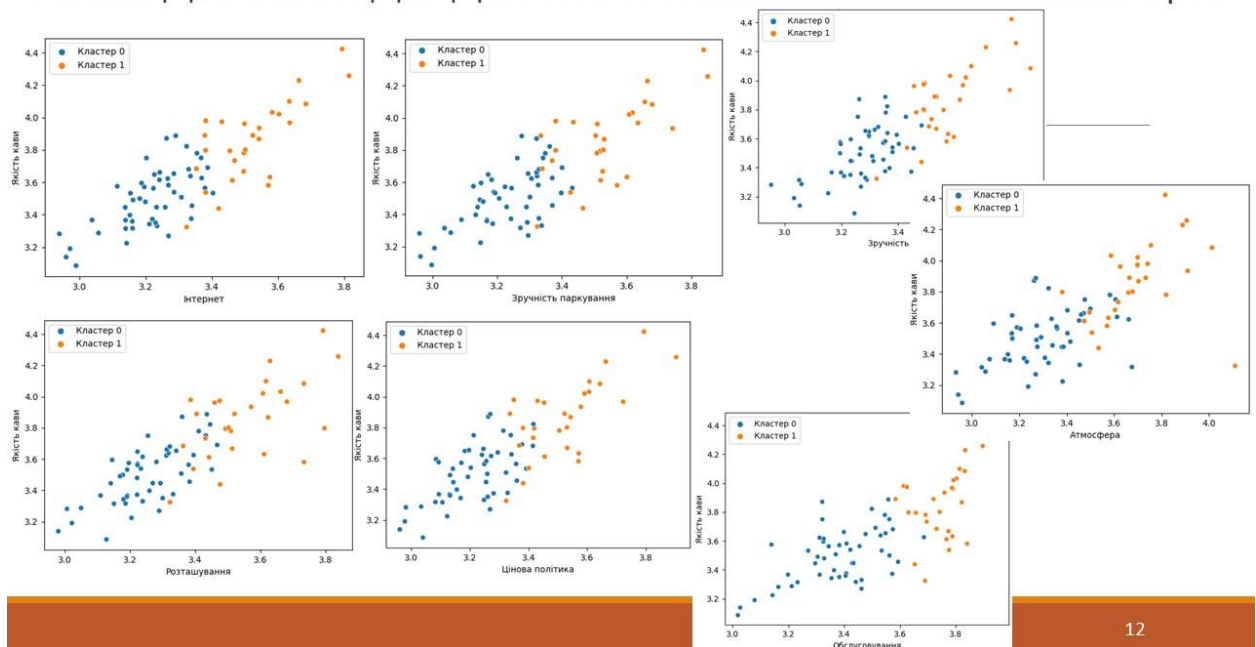
Сторінки сценаріїв роботи користувачів з програмою



11

Рисунок В.11 – Слайд № 11 (Сторінки сценаріїв роботи користувачів з програмою)

Розподіл закладів для любителів кави на 2 кластери



12

Рисунок В.12 – Слайд № 12 (Розподіл закладів для любителів кави на 2 кластери)

Результати експериментального дослідження

Кількість кластерів	Метод рекомендування закладів для любителів кави	Сентимент-аналіз та спектральна кластеризація
2	0,04138394	0,074747796
3	0,032516202	0,084735928
4	0,039730374	0,080925324
5	0,041605689	0,073584029
6	0,037999292	0,07410221
7	0,033188433	0,07155749
8	0,037385413	0,071407412
9	0,026301567	0,070537536
10	0,024168007	0,070184982

13

Рисунок В.13 – Слайд № 13 (Результати експериментального дослідження)

Висновки

За результатами виконання роботи досліджено проблему рекомендування закладів для любителів кави та розроблено програму для інформування любителів кави про заклади, де подається кава або напої з кави, про напої з кави, які подаються у цих закладах, використовуючи для цього пошук цих закладів і рекомендування закладів, подібних тому, який розглядає користувач. Для цього надається можливість створення рейтингів через оцінювання закладів та напоїв, які в них подаються.

Наукова новизна полягає в методі рекомендування закладів для любителів кави, який розділяє заклади у певному місці на кластери, відносячи їх до подібних, використовуючи для цього виділення з рейтингів інформації за окремими критеріями, сентимент-аналіз і визначення оцінок на його основі за кожним з критеріїв у разі відсутності таких оцінок у рейтингу та виконуючи спектральний кластерний аналіз за визначеними оцінками.

14

Рисунок В.14 – Слайд № 14 (Висновки)