Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
(повна назва інституту/факультету)

Кафедра інформаційних технологій та комп'ютерної фізики
(повна назва кафедри)

Інформаційно-аналітичний сайт «Shop Rate» для оцінювання торговельних закладів

Кваліфікаційна робота

Рівень вищої освіти - перший (бакалаврський)

Виконав:
студент *4* курсу, групи *417ск*
спеціальності
126 Інформаційні системи та технології
(назва спеціальності)

Попюк Роман Артурович
(прізвище та ініціали)
Керівник *д.т.н., доц., Баловсяк С. В.*
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:

Протокол засідання кафедри № 20

від „15" червня 2023 р.

зав. кафедри доц. Борча М. Д.

Чернівці – 2023

РЕЦЕНЗІЯ

НА БАКАЛАВРСЬКУ РОБОТУ

Студента Попюка Романа Артуровича
Навчально-науковий інститут фізико-технічних та комп'ютерних наук
Кафедра інформаційних технологій та комп'ютерної фізики
Чернівецького національного університету імені Юрія Федьковича

на тему: *Інформаційно-аналітичний сайт «Shop Rate» для оцінювання торговельних закладів*

Актуальність теми *пояснюється практичною потребою у створенні програми, яка б дозволяла оцінювати торговельні заклади та надавала інформацію про них користувачам у зручній для них формі*

Практичне значення *отриманих у роботі результатів полягає в тому, що завдяки розробленому інформаційно-аналітичному сайту користувачі можуть отримувати важливу інформацію про магазини і робити цілеспрямований вибір*

Структура та зміст роботи *робота складається з 3-х розділів. У першому розділі проведено системний аналіз предметної області – інформаційно-аналітичних сайтів, виконано постановку задачі. У другому розділі подано інформацію про вхідні та вихідні дані програми, її функції та структуру, розроблену базу даних. У третьому розділі описано програмну реалізацію інформаційно-аналітичного сайту, проведено тестування розробленої інформаційної системи*

Зауваження *до недоліків роботи слід віднести окремі граматичні помилки*

Оцінка *у результаті виконання бакалаврської роботи студент Попюк Р.А. засвоїв принципи роботи з базами даних, створив працездатний інформаційно-аналітичний сайт для оцінювання торговельних закладів. Вважаю, що завдання бакалаврської роботи Попюк Р.А. виконав і заслуговує оцінки „відмінно" та присвоєння кваліфікації бакалавра зі спеціальності 126 Інформаційні системи та технології.*

Рецензент
Завідувач кафедри комп'ютерних
систем та мереж
Чернівецького національного університету
кандидат фізико-математичних наук
Воробець Г.І.

"15" 06 2023 р.
(підпис)

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра інформаційних технологій та комп фізики

ID перевірки:
1015632514

Дата перевірки:
17.06.2023 13:21:11 EEST

Тип перевірки:
Doc vs Internet + Library + DB

Дата звіту:
17.06.2023 13:29:28 EEST

ID користувача:
36471

Назва документа: Попюк

Кількість сторінок: 14 Кількість слів: 5409 Кількість символів: 46888 Розмір файлу: 72.82 KB ID файлу: 1015279087

10.5%
Схожість

Найбільша схожість: 4.4% з джерелом з Бібліотеки (ID файлу: 1015258902)

4.36% Джерела з Інтернету	3	Сторінка 16
4.4% Джерела з Бібліотеки	4	Сторінка 16

0.35% Цитат

Цитати	1	Сторінка 17

Вилучення списку бібліографічних посилань вимкнене

10.9%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 12 слів та 2%)

8.84% Вилучення з Інтернету	242	Сторінка 18
7.38% Вилученого тексту з Бібліотеки	421	Сторінка 20

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1

Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
(повна назва інституту/факультету)

Кафедра інформаційних технологій та комп'ютерної фізики
(повна назва кафедри)

Інформаційно-аналітичний сайт «Shop Rate» для оцінювання торговельних закладів

Кваліфікаційна робота

Рівень вищої освіти - перший (бакалаврський)

Виконав:
студент *4* курсу, групи *417ск*
спеціальності
126 Інформаційні системи та технології
(назва спеціальності)

Попюк Роман Артурович
(прізвище та ініціали)
Керівник *д.т.н., доц., Баловсяк С. В.*
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:

Протокол засідання кафедри № ____

від „____" _____ 2023 р.

зав. кафедри _____доц. Борча М. Д.

Чернівці – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
Кафедра інформаційних технологій та комп'ютерної фізики

ЗАТВЕРДЖУЮ
Завідувач кафедри
док. фіз.-мат. наук, доц.
_____М. Д. Борча
" ___" _____ 2023 р.

Інформаційно-аналітичний сайт «Shop Rate» для оцінювання торговельних закладів

ЛИСТ ЗАТВЕРДЖЕННЯ

УЗГОДЖЕНО
Керівник роботи
докт. техн. наук, доцент
_____ С.В. Баловсяк
"___" _____ 2023 р.

Виконавець
студент 4-го курсу
_____ Р. А. Попюк
"___" _____ 2023 р.

2023

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Попюку Роману Артуровичу
(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) Інформаційно-аналітичний сайт «Shop Rate» для оцінювання торговельних закладів
керівник роботи *Баловсяк Сергій Васильович, докт. техн. наук, доцент*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від "__"_____2023 року №_____

2. Строк подання студентом проекту (роботи) *2023 р.*

3. Вихідні дані до проекту (роботи) *Мета роботи – створення інформаційно-аналітичного веб-ресурсу (сайту) з інформацією про магазина міста. Користувачі сайту повинні мати можливість обмінюватися відгуками про магазини, оцінювати якість його роботи. Рейтинг магазину формувати на основі таких відгуків користувачів. Інформацію про магазини зберігати у базі даних MySQL. Створення програмного коду виконати в середовищах розробки як Visual Studio Code і IntelliJ IDEA.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) дослідити існуючі інформаційні системи-аналоги
2) описати принцип роботи рейтингових систем
3) створити інформаційно-аналітичний веб сайт
4) виконати аналіз даних сайту
5) дослідити ефективність роботи розробленої програми

5. Перелік графічного матеріалу
1. Логічна модель бази даних
2. Діаграма прецедентів програми

Студент _____ *Попюк Р.А.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Баловсяк С.В.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота виконана студентом 417ск групи Попюком Романом Артуровичем. Тема «Інформаційно-аналітичний сайт «Shop Rate» для оцінювання торговельних закладів». Робота направлена на здобуття ступеня бакалавр за спеціальністю 126 «Інформаційні системи та технології».

Метою кваліфікаційної роботи є створення веб-ресурсу з інформацією про магазина міста, у якому користувачі матимуть змогу обмінюватися відгуками про магазини, своїми оцінками стосовно їх роботи. Рейтинг магазину формується на основі таких відгуків.

Бакалаврська робота містить: кількість сторінок – 71, таблиць – 3, рисунків – 32, додатків – 1, використаних джерел – 15.

Ключові слова: інформаційна система, рейтинг, токен, Java, Spring Boot.

Робота містить результати власних досліджень. Використання чужих ідей, результатів і текстів мають посилання на відповідне джерело.

ANNOTATION

The qualification work was carried out by Roman Arturovich Popyuk, a student of group 417sk. The topic is Information and Analytical Website for Rating Retail Establishments "Shop Rate". The work is aimed at obtaining a bachelor's degree in the field of specialization 126 "Information Systems and Technologies".

The goal of the qualification work is to create a web resource with information about stores in the city. Specifically, website users will have the opportunity to exchange reviews about stores, provide their ratings regarding their performance, which ultimately form the store's rating.

The bachelor's work consists of 71 pages, 3 tables, 32 figures, 1 appendix, and references to 15 sources.

Keywords: information system, rating, token, Java, Spring Boot.

The work contains the results of original research. The use of other people's ideas, results, and texts is properly referenced to the respective source.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних.

Js – JavaScript.

TS – TypeScript.

DAO – Data Transfer Object.

ВСТУП

На сьогоднішній день можливості інтернету дозволяють всім користувачам дистанційно знаходити необхідну інформацію з будь-якої теми. Тому сьогодні створення веб-ресурсів досить популярне. Вдало організований інформаційний сайт надає переваги як для користувачів (шукачів інформації), так і для власників сайтів (компаній, фірм) [1-2].

Інформаційно-аналітичні сайти (портали) займають важливе місце серед сучасних веб-ресурсів, оскільки не тільки надають детальну інформацію на певну тематику, але й виконують її аналіз. Такі сайти містять інструменти для взаємодії з користувачами та пошуку інформації, рейтинги, а також дозволяють відвідувачам спілкуватися між собою, наприклад, за допомогою чату або через відгуки. Одними з найбільш розповсюджених інформаційно-аналітичних сайтів є сайти для міст, які містять новини, погоду, афіші, розклад транспорту, дошку оголошень, каталоги підприємств, магазинів і іншу корисну інформацію. Тому даний дипломний проект присвячений розробці веб-ресурсу з інформацією про магазини міста.

Метою роботи є створення інформаційно-аналітичного сайту "Shop Rate" для оцінювання торговельних закладів. Сайт надає користувачам можливість дізнатися про різні торговельні заклади, оцінювати їх та залишати відгуки. На основі відгуків користувачів формується рейтинг магазинів. Такий сайт для користувачів є зручним і достовірним джерелом інформації про торговельні заклади.

Розробка інформаційно-аналітичного сайту є актуальним завданням, оскільки у сучасному світі велике значення приділяється якості обслуговування та вибору товарів і послуг. Розроблений інформаційно-аналітичний сайт "Shop Rate" відповідає цій потребі, надаючи користувачам зручну платформу для обміну думками, оцінками та відгуками про різні торговельні заклади. Крім того, сайт допомагатиме покращити якість

обслуговування в торговельних закладах, заохочуючи їх до позитивних змін і відповіді на потреби клієнтів.

Об’єктом дослідження є інформаційна системи для отримання даних про торговельні заклади.

Предметом дослідження є інформаційно-аналітичний сайт з даними про магазини, їх товари та рейтинги в межах країни.

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Аналіз предметної області

Предметною областю інформаційно-аналітичні сайти для надання даних про магазини міста, зокрема, про їх рейтинги.

Рейтинг – це числовий або порядковий показник успішності або популярності, який відображає важливість або вплив певного об'єкта або явища. Існують різні види рейтингів, які класифікуються за певними параметрами. Враховуючи область призначення, можна навести такі приклади існуючих рейтингів:

– ступінь популярності відомих діячів в галузі політики, культури і т.д;

– рівень лояльності споживачів до торгової марки товару чи послуги (наприклад, рейтинг конкурсу народних уподобань «Фаворити Успіху»);

– кредитний рейтинг підприємства, корпорації;

– індивідуальний числовий показник оцінки, наприклад, спортивних досягнень у класифікаційному списку (рейтинг-лист);

– рейтинг змісту фільму, передачі або відеогри;

– рейтинг телевізійних та радіо-програм (наприклад, рейтинг «Україна має талант» становив 30,3 %);

– «зірковий» рейтинг готелів, або рейтинг ресторанів та майстерності шеф-кухарів (наприклад, Мішлен, Го-Мійо, S. Pellegrino World's 50 Best Restaurants від британського журналу Restaurant).

За кількістю характеристик, на підставі яких оцінюють об'єкти, що досліджуються, рейтинги можна розділити на однофакторні і багатофакторні (рейтингів і власне рейтинги).

За джерелами інформації, на підставі якої складають рейтинги, їх також можна віднести до декількох категорій:

– складені за результатами опитування цільової аудиторії;

– побудовані на вторинній інформації;

– засновані на даних, отриманих безпосередньо від об'єктів, що

ранжуються.

Залежно від того, скільки разів був випущений рейтинговий продукт, виділяють періодичні і неперіодичні рейтинги.

Однак яким би різноманітнім не була кількість рейтингів, коло цілей, в яких вони використовуються, обмежене.

Попит на рейтинги створюють кілька класів споживачів: інвестори, комерційні і некомерційні організації різного спрямування, громадяни і т.д.

Рейтинги необхідні для здійснення коректного вибору в умовах як надлишкової, так і неповної інформації, оскільки в усіх випадках користувачам важко обрати правильне рішення: якщо даних багато, то їх складно узагальнити і структурувати, якщо мало – висока ймовірність зробити помилку, не врахувавши важливі параметри.

Для створення рейтингів виділяють такі етапи роботи:

- постановка задачі;
- визначення важливих параметрів, що характеризують об’єкт;
- визначення відносної значимості кожного параметру;
- збір інформації;
- обробка інформації;
- оцінка і представлення результату.

Перш ніж приступати до створення рейтингу, потрібно не лише знати, які об’єкти в нього ввійдуть, але й розуміти, в яких цілях продукт буде використаний. Вибір методики залежить від того, навіщо і кому потрібний конкретний рейтинг.

Можна оцінювати об’єкти тільки по одному параметру, наприклад, університети – за конкурсом на вступних іспитах, футболістів – за кількістю забитих м’ячів, дистриб’юторів – за рівнем цін і т.д. В результаті буде побудований однофакторний рейтинг. Такі рейтинги досить об'єктивні, адже точність оцінки залежить лише від того, наскільки достовірна вихідна інформація. Однак об’єкти рідко оцінюють на підставі одного параметру

Наприклад, аптеки, вибираючи дистриб’юторів, звертають увагу не тільки на рівень цін, а й на сервіс, можливість покупки в кредит, асортимент і т.д. Тому, незважаючи на те, що однофакторний рейтинг має свої переваги, частіше об'єкти ранжують на підставі кількох параметрів (багатофакторний рейтинг). Головне питання для багатофакторних рейтингів полягає в тому, скільки потрібно таких параметрів.

Після визначення характеристик необхідно вияснити значимість. При цьому часто значимість всіх факторів беруть за одиницю, а відносну значимість для кожної характеристики виражають в долях одиниці.

Наступним етапом буде безпосередньо збір інформації і обробка інформації. Після чого останнім етапом є вивід результуючого рейтингу. На останньому етапі слід визначити форму відображення інформації.

Отже, проаналізовано предметну область, а саме введено поняття рейтингу, їх класифікацію і етапи створення.

1.2 Аналіз існуючих аналогів

Існує значна кількість сайтів для перегляду оцінок діяльності різних магазинів. В якості аналогів до програмного продукту, що буде реалізовано в даному дипломному проєкті, розглянемо сайти Price.ua та ek.ua та їх функціонал.

Price.ua – універсальний сайт порівняння цін на ринку України. Він надає зручний механізм пошуку товарів та послуг в інтернет-магазинах в ході порівняння цін, описів і характеристик. Для зручності користувачам надається багаторівневий пошук товарів за категоріями, виробниками і цінами. Можна подивитися фото, ознайомитися з описом і оцінити динаміку цін потрібного товару. Завдяки відгукам покупців користувачі дізнаються про недоліки і переваги, виявлені в ході експлуатації, а рейтинг фірм, що формується користувачами, дозволить оцінити надійність продавця.

На рисунку 1.1 показна стартова сторінка сайту Price.ua, де зображено каталог товарів, пошук і такі розділи як Популярні категорії, Акції та знижки, Популярні товари, Магазини в Україні, Бренди тощо.

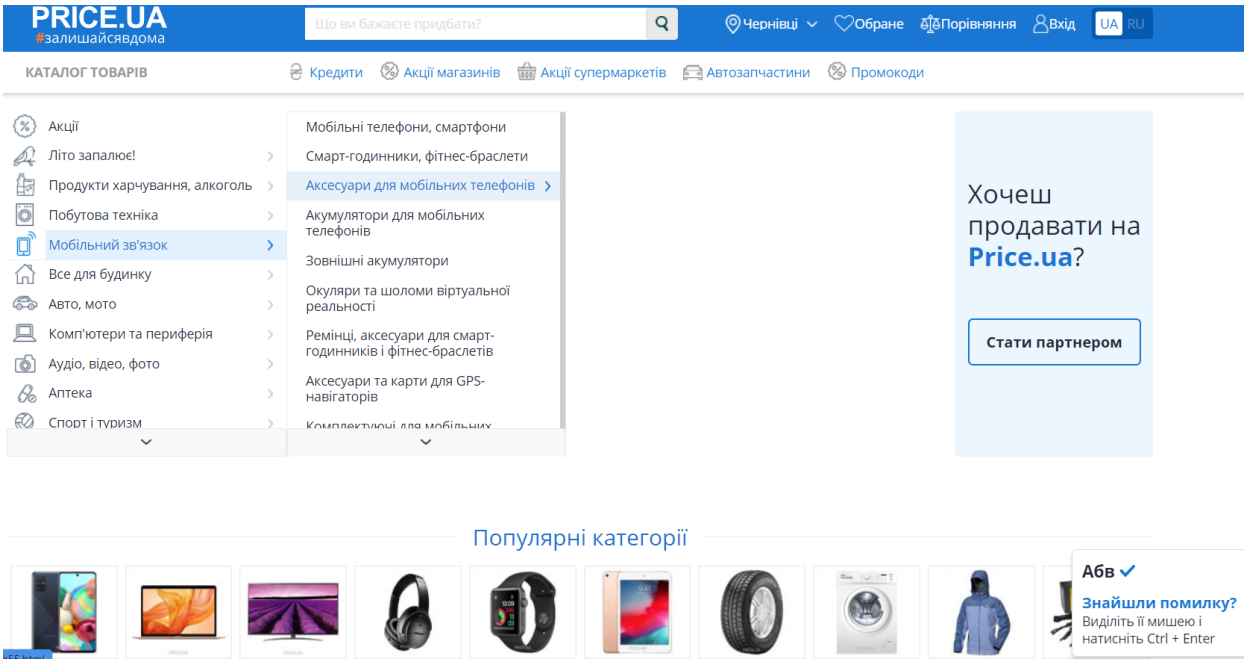

Рисунок 1.1 – Стартова сторінка сайту Price.ua

На рисунку 1.2 зображено розділ «Акції та знижки» сайту Price.ua.

Рисунок 1.2 – Сторінка «Акції та знижки» сайту Price.ua

На даній сторінці розташована інформація про акції в різних магазинах, каталог товарів різних виробників. Аналогічний вигляд мають й інші розділи, що розміщені в нижній частині головної сторінки.

На сайті Price.ua передбачено можливість пошуку інформації про бажаний товар. Для цього призначено поле введення у верхній частині сайту (рисунок 1.3).

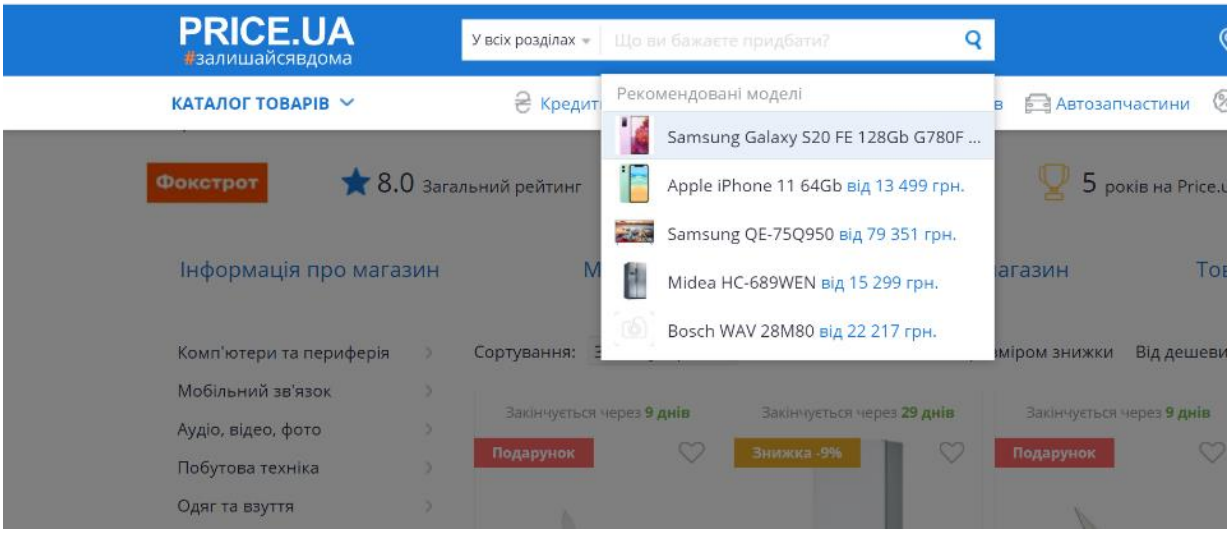

Рисунок 1.3 – Пошук інформації про бажаний товар на сайті Price.ua

Результатом пошуку є сторінка з інформацією про даний товар: в якому магазині та по якій ціні можна придбати даний товар (рисунок 1.4).

Рисунок 1.4 – Результат пошуку

Корисним на сайті Price.ua для даного дипломного проєкту є сторінка про магазини. Вона містить список магазинів, які зареєстровані в базі даних сайту, і для кожного магазину можна переглянути статистичну інформацію з приводу роботи даного магазину. Приклад відображення інформації про магазин зображено на рисунку 1.5.

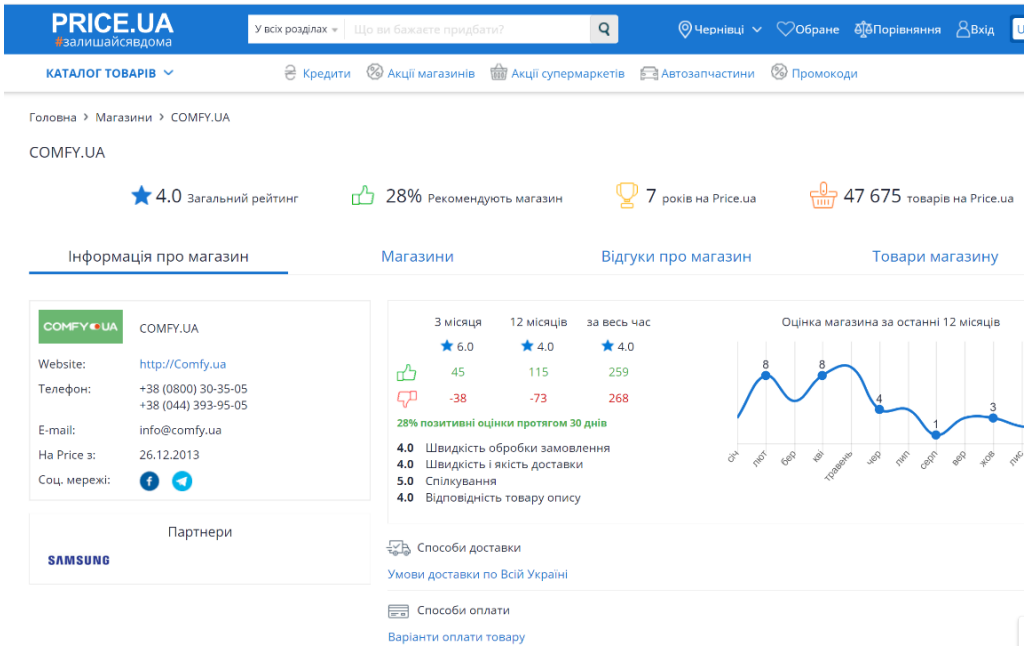

Рисунок 1.5 – Сторінка про магазин Comfy

Наступний веб сайт, який розглядався в якості аналогу, це E-katalog.ua. Головна сторінка сайту містить поле для здійснення пошуку, каталог товарів і категорії. Вигляд її зображено на рисунку 1.6.

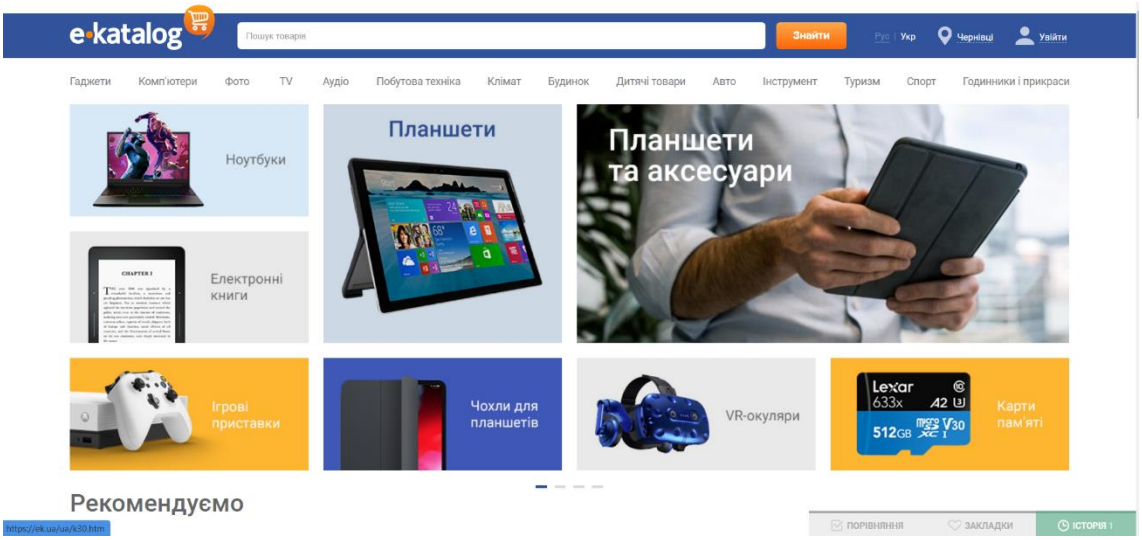

Рисунок 1.6 – Головна сторінка E-katalog.ua

На рисунку 1.7 зображено сторінку «Новини», де відображаються новини, нові матеріали за останній час.

Рисунок 1.7 – Сторінка новин E-katalog.ua

На рисунку 1.8 зображено сторінки «Відгуки про товар». Тут можна переглянути інші відгуки користувачів сайту та додати відгук про товар.

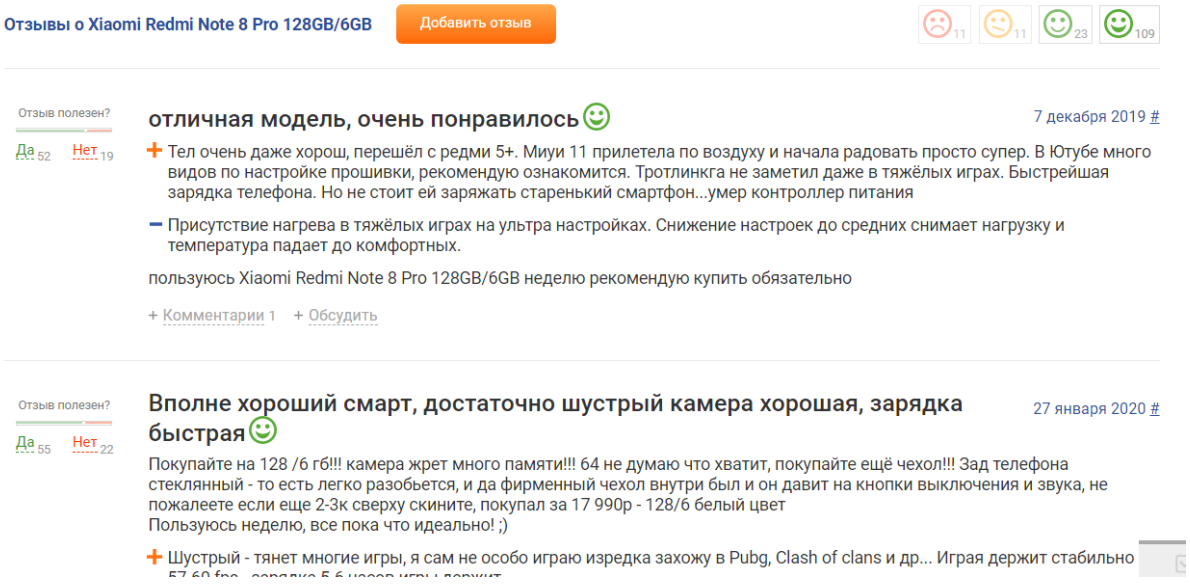

Рисунок 1.8 – Сторінка відгуків E-katalog.ua

На рисунку 1.9 зображено сторінку для створення відгуку.

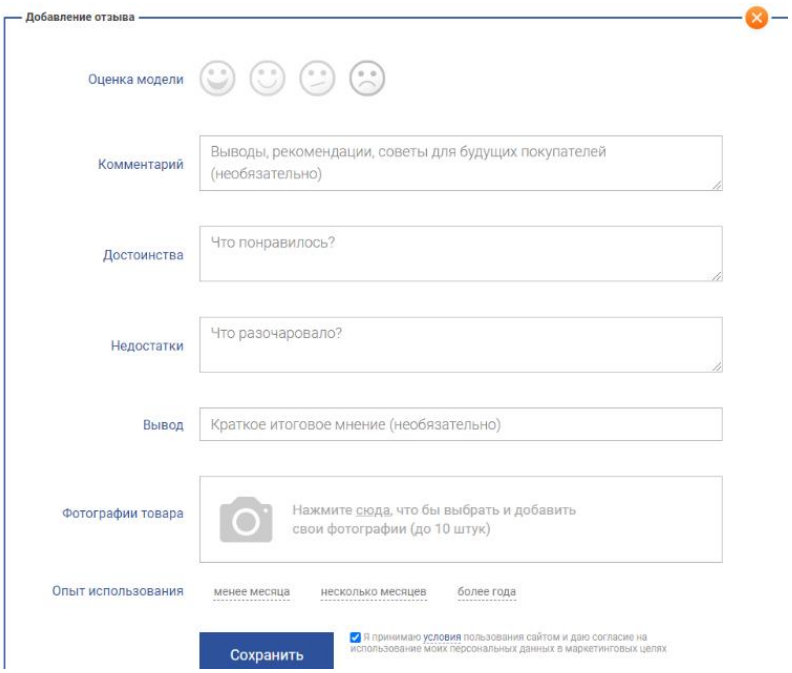

Рисунок 1.9 – Сторінка створення відгука E-katalog.ua

Підсумуємо переваги і недоліки сайтів-аналогів у таблиці 1.1.

Таблиця 1.1 – Переваги та недоліки сайтів-аналогів

Назва веб-сайтів	E-katalog.ua	Price.ua
Відображення рейтингів про магазин	–	+
Повна статистика магазину	–	+
Відгуки про товар	+	+
Порівняння характеристик товару	+	–

Таким чином, розглянуті веб-сайти Price.ua і E-katalog.ua мають ряд недоліків, що є підставою для розробки власного програмного продукту.

Висновки до розділу 1

Проаналізувавши сайти-аналоги Price.ua і E-katalog.ua зроблено висновок про потребу у розробці власного інформаційно-аналітичного сайту "Shop Rate" з інформацією про торговельні заклади (магазини) міста. Сайт повинен виконувати такі функції: надавати користувачам інформацію про різні торговельні заклади, дозволяти оцінювати їх та залишати відгуки. На основі відгуків користувачів формувати рейтинг магазинів.

Результатом роботи сайту є покращення якості обслуговування в торговельних закладах, заохочення їх до позитивних змін у відповідності до потреб клієнтів.

РОЗДІЛ 2. ПОБУДОВА СИСТЕМИ

2.1 Технічне завдання

Розглянемо технічне завдання дипломної роботи, яке полягає у розробці інформаційно-аналітичного сайту про магазини міста з відгуками користувачів про них, а також з рейтингом магазинів по місту.

Вхідні дані:

– інформація про магазин (логотип, вебсайт, пошта, категорія, телефон);

– відгуки про магазин;

– потрапляння чи ні магазину у список улюблених;

– оцінка роботи магазину.

Вихідні дані:

– інформація про магазин;

– кількість відгуків;

– список улюблених магазинів;

– рейтинг магазинів.

Функціональні можливості користувачів при роботі з сайтом залежать від рівня доступу:

– адміністратор;

– користувач.

Функціональна специфікація користувача:

– аутентифікація користувача;

– зміна даних користувача;

– доступ до сайту за доменною адресою;

– перегляд інформації про магазин;

– додавання відгуків про магазин;

– оцінювання діяльності магазину.

Функціональна специфікація адміністратора:

– аутентифікація адміністратора;

– введення інформації;

– редагування інформації;

– видалення інформації;

– збереження даних.

Потрібно забезпечити стабільність роботи із сайтом при наявності зв’язку з сервером.

Програмне забезпечення, що розробляється, повинно вирішувати наступні задачі:

– принципові;

– функціональні;

– сервісні.

До принципових задач відносять:

– забезпечення збереження даних про магазин;

– зручний та зрозумілий інтерфейс;

– мінімізація можливості виникнення помилок у роботі вебпродукту.

До функціональних задач відносять збереження зміни даних.

До сервісних задач відносяться:

– організація головного меню;

– оптимізація для телефонів;

– забезпечення маніпуляції з даними.

Надійність даного проєкту забезпечується використанням методології мов програмування Java та JavaScript.

Мінімальні вимоги до апаратного забезпечення:

– операційна система – Microsoft Windows XP, 7, 8, 10;

– наявність браузера (Google Chrome, Firefox, Opera).

Етапи розробки додатку можуть уточнювати згідно календарного плану робіт по узгодженню між замовником та виконавцем (таблиця 2.1).

Таблиця 2.1 – Етапи роботи над дипломним проєктом

Етапи виконання роботи	Термін виконання та приблизний обсяг робіт	Звітні матеріали
Перед-проєктне дослідження	3.05-10.05 Аналіз предметної області та існуючих аналогів, вивчення технологій, структури даних, методів рішення тощо	Опис предметної області, аналіз аналогів, вибору методів рішення завдання та засобів розробки, оформлення технічного завдання
Моделю-вання проєкту	12.05-16.05 Створення макету вебсайту під ПК та мобільні пристрої. Вибір стилів шрифтів, кольорової гами. Визначення стилів блоків.	Макет майбутнього вебсайту створений у додатку для web-дизайнерів «Figma».
Технічний проєкт	17.05-19.05 Проєктування програмного забезпечення. Розробка алгоритму, визначення форми представлення даних, архітектури програми	Специфікація вимог (опис алгоритму, правил, функціональних вимог, критеріїв якості тощо), опис концептуальної, інформаційної та функціональної моделей
Робочий проєкт	20.05-5.06 Реалізація програмного забезпечення	Опис засобів розробки, розробка документів на супроводження ПЗ (інструкції програмісту та користувачу), опис плану тестування (розробка тестів, аналіз результатів тестування)

Після закінчення відповідного етапу робіт формується відповідний комплект документації.

Завершення етапу розробки фіксується відповідним протоколом захисту дипломних проєктів.

Таким чином, розроблено технічну документацію для поставленої задачі та розглянуто етапи розробки веб сайту.

2.2 Моделювання програмного забезпечення

Тепер розглянемо процес моделювання програмного забезпечення.

Діаграми прецедентів є первісним, концептуальним представленням (концептуальною моделлю) програмної системи у процесі проєктування й розробки системи. Діаграми прецедентів виступають основою подальшої деталізації системи у формі різних логічних і фізичних моделей. Зокрема, прецеденти допомагають перевіряти й контролювати архітектуру програмної системи у процесі її розробки. Для проєктування діаграми прецедентів спочатку необхідно визначити дійових осіб (акторів), а потім визначити, які дії у системі може виконувати кожен з акторів. На рисунку 2.1 зображено концептуальну модель для даного проєкту.

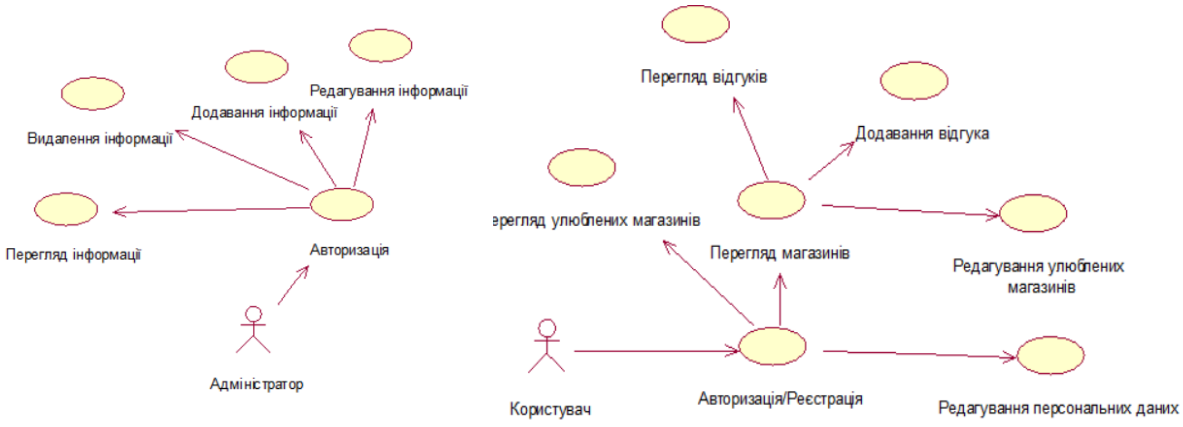

Рисунок 2.1 – Модель прецедентів

На рис. 2.1 зображено взаємодію користувача та адміністратора з вебсайтом. Адміністратор має доступ до адміністративної частини програмної системи. В його обов’язки входить збір та опублікування даних про нові магазини, їх редагування та маніпуляції. В моделі також представлено актора користувача. Користувач може переглядати інформацію про магазини, про улюблені магазини, добавляти та переглядати відгуки, а також редагувати магазин улюблених.

Діаграма взаємодії – одна з моделей опису поведінки взаємодіючих груп об’єктів в UML. Діаграма взаємодії для розробленої частини програмного продукту зображена на рисунках 2.2-2.4. На даній діаграмі зображена взаємодія користувача з системою, певна послідовність дій для ефективної роботи програмного продукту та його можливості. Основними діями користувачів в даній системі є зберігання, перегляд та обробка даних.

На рисунку 2.2 зображена модель взаємодії, що описує процес перегляду даних з бази даних.

Рисунок 2.2 – Діаграма взаємодії при перегляді таблиць

На рисунку 2.3 зображена модель взаємодії, що описує процес додавання даних в таблицю.

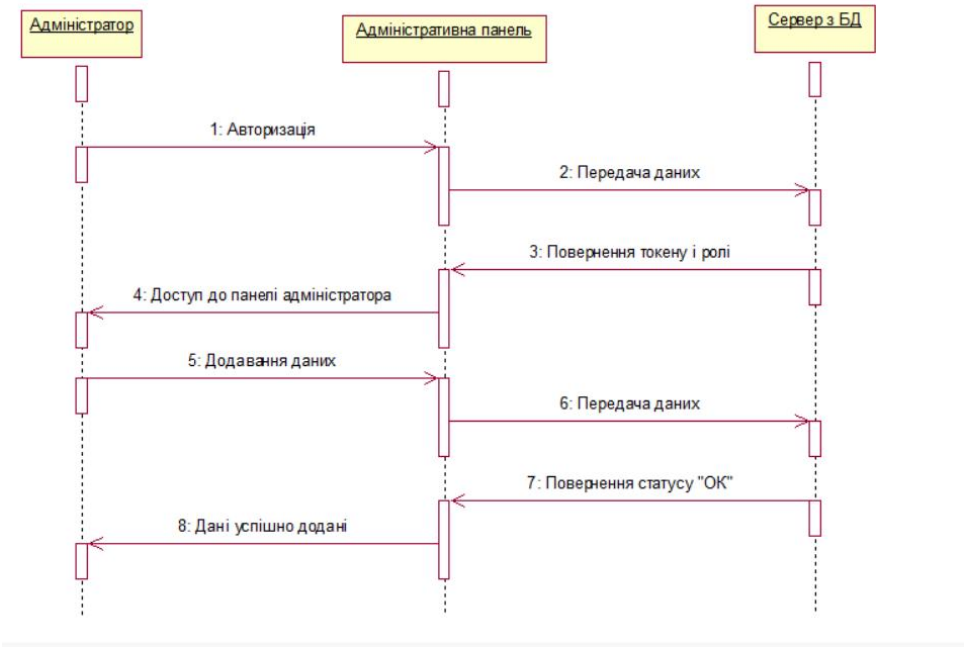

Рисунок 2.3 – Діаграма взаємодії при додаванні даних

На рисунку 2.4 зображена модель взаємодії, що описує процес редагування даних в таблицю.

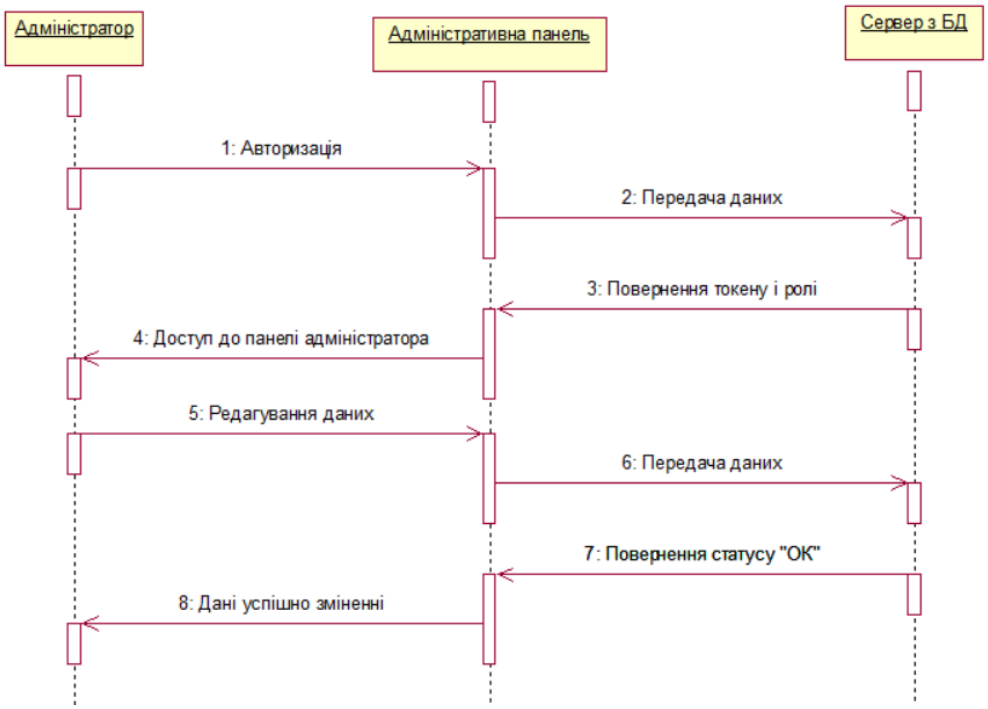

Рисунок 2.4 – Діаграма взаємодії для редагування даних

За допомогою розроблених моделей можна краще зрозуміти предметну область.

2.3 Моделювання даних

Розглянемо фізичну і логічну модель даних база даних.

При роботі з структурованою інформацією, для її збереження, зручно використовувати реляційну модель даних.

Після аналізу предметної області була розроблена фізична модель даних.

Для збереження даних про магазин використовується таблиця «User_Entity», інформація яку містить таблиця відображена в рисунку 2.5.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	id	bigint(20)			No	None		AUTO_INCREMENT
☐	2	email	varchar(255)	utf8_general_ci		Yes	NULL		
☐	3	lastname	varchar(255)	utf8_general_ci		Yes	NULL		
☐	4	logo	mediumblob			Yes	NULL		
☐	5	name	varchar(255)	utf8_general_ci		Yes	NULL		
☐	6	password	varchar(255)	utf8_general_ci		Yes	NULL		
☐	7	phone	varchar(13)	utf8_general_ci		Yes	NULL		
☐	8	surname	varchar(255)	utf8_general_ci		Yes	NULL		
☐	9	user_name	varchar(255)	utf8_general_ci		Yes	NULL		

Рисунок 2.5 – Таблиця «User_Entity»

Опишемо поля таблиці «User_Entity»:

– id – ціле числове поле, унікальне поле;
– email – текстове поле, для збереження поштової адреси користувача;
– lastname – текстове поле, для збереження по батькові користувача;
– logo –для збереження аватарки користувача;
– name – текстове поле, для збереження імені користувача;
– password – текстове поле, для збереження паролю користувача;
– phone – текстове поле, для збереження телефону користувача;
– surname – текстове поле, для збереження прізвища користувача;
– user_name – текстове поле, для збереження нікнейма користувача.

Для збереження даних про магазин створена таблиця «Info_Shop». Структура таблиці зображена на рисунку 2.6.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	id	bigint(20)			No	None		AUTO_INCREMENT
☐	2	contacts_people	varchar(255)	utf8_general_ci		Yes	NULL		
☐	3	email	varchar(255)	utf8_general_ci		Yes	NULL		
☐	4	logo	mediumblob			Yes	NULL		
☐	5	name_shop	varchar(255)	utf8_general_ci		Yes	NULL		
☐	6	phone	varchar(255)	utf8_general_ci		Yes	NULL		
☐	7	website	varchar(255)	utf8_general_ci		Yes	NULL		
☐	8	id_kategory	bigint(20)			No	None		

Рисунок 2.6 – Структура таблиці «Info_Shop»

Опишемо поля таблиці «Info_Shop»:

– id – ціле числове унікальне поле;

– contacts_people – текстове поле, для збереження інформації про контактних осіб магазину;

– email – текстове поле, для збереження поштової адреси магазину;

– logo –для збереження аватарки магазину;

– name_shop – текстове поле, для збереження імені магазину;

– phone – текстове поле, для збереження телефона магазину;

– website – текстове поле, для збереження сайт магазина;

– id_kategory – ціле числове поле, унікальне поле для таблиці «Kategory».

Для збереження даних про улюблені магазини створена таблиця «Favourite». Її структура зображена на рисунку 2.7.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	id	bigint(20)			No	None		AUTO_INCREMENT
☐	2	id_shop	bigint(20)			No	None		
☐	3	id_users	bigint(20)			No	None		

Рисунок 2.7 – Структура таблиці «Favourite»

Опишемо поля таблиці «Favourite»:

– id – ціле числове унікальне поле;

– id_shop – ціле числове поле, унікальне поле для таблиці «Info_Shop»;

– id_user – ціле числове поле, зовнішній ключ для зв’язку з таблицею «User_Entity».

Для збереження даних про відгуки магазинів створена таблиця «Review_Entity». Її структура зображена на рисунку 2.8.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	bigint(20)			No	None		AUTO_INCREMENT
2	ball	int(11)			Yes	NULL		
3	data	date			Yes	NULL		
4	header	varchar(255)	utf8_general_ci		Yes	NULL		
5	text	varchar(414)	utf8_general_ci		Yes	NULL		
6	id_shop	bigint(20)			No	None		
7	id_user	bigint(20)			No	None		

Рисунок 2.8 – Структура таблиці «Review_Entity»

Опишемо поля таблиці «Review_Entity»:

– id – ціле числове унікальне поле;

– ball – ціле числове поле, для збереження балів відгуку;

– data – датове тип поле, для збереження дати відгуку;

– header – текстове поле, для збереження заголовку відгуку;

– text – текстове поле, для збереження тексту відгуку;

– id_shop – ціле числове поле, зовнішній ключ для зв’язку з таблицею «Info_Shop»;

– id_user – ціле числове поле, зовнішній ключ для зв’язку з таблицею «User_Entity».

Для збереження даних про категорії магазинів створена таблиця «Kategory». Структура таблиці зображено на рисунку 2.9.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id_kategory	bigint(20)			No	None		AUTO_INCREMENT
2	name_kategory	varchar(255)	utf8_general_ci		Yes	NULL		

Рисунок 2.9 – Структура таблиці «Kategory»

Опишемо поля таблиці «Kategory»:

– id_kategory – ціле числове унікальне поле;

– name_kategory – текстове поле, для збереження назви категорії.

Логічна модель створеної бази даних зображена на рисунку 2.10.

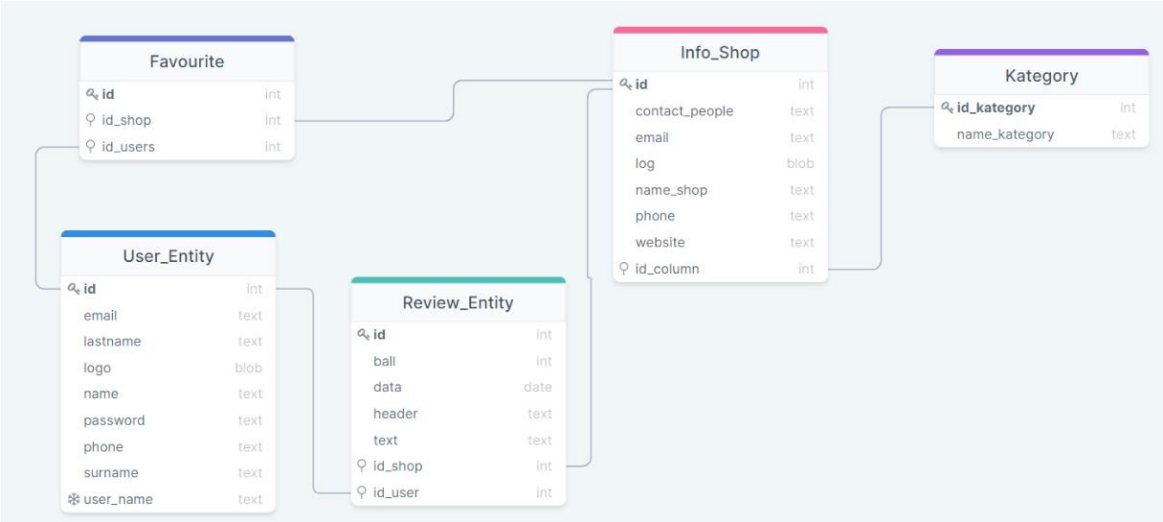

Рисунок 2.10 – Логічна модель бази даних

Таким чином, розроблено логічну і фізичну модель таблиць бази даних.

2.4 Проєктування інтерфейсу

Розглянемо проєктування інтерфейсу веб сайту. Сайт повинен коректно відображатися в сучасних браузерах останніх версій. В таблиці 2.2 наведено і коротко описано призначення сторінок вебсайту.

Таблиця 2.2 – Сторінки інтернет-орієнтованої системи

Назва сторінки	Призначення
Головна	Перегляд каталогу категорій магазинів, список популярних магазинів
Каталог	Перегляд всіх магазинів і пошук
Улюблені	Перегляд улюблених магазинів користувача
Профіль	Перегляд і редагування даних про користувача
Характеристика магазину	Перегляд інформації про магазин і відгуки про нього
Написання відгука	Написання відгука про діяльність даного магазину

Структура головної сторінки зображена на рисунку 2.11.

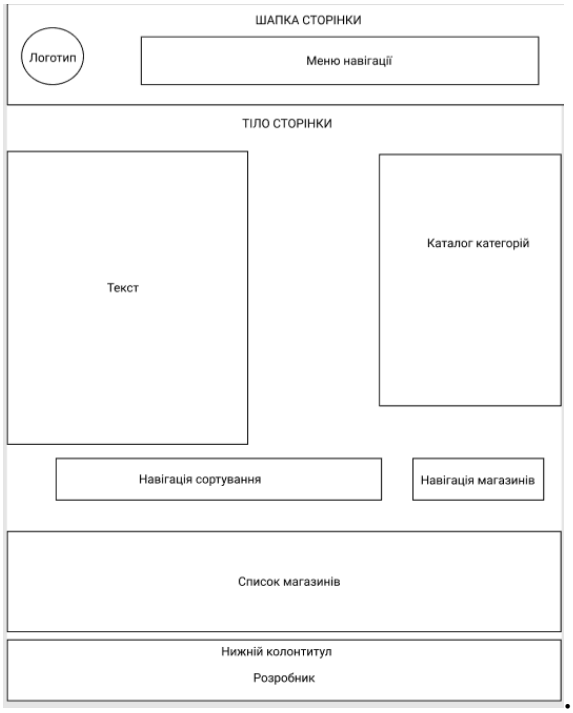

Рисунок 2.11 – Структура головної сторінки

На головній сторінці розташовані такі статичні елементи як нижній колонтитул з назвою розробника. Також тут розміщені унікальні елементи слайдер та хідер. Для переходу між сторінками використовується меню навігації. Структура сторінки «Каталог» зображена на рисунку 2.12.

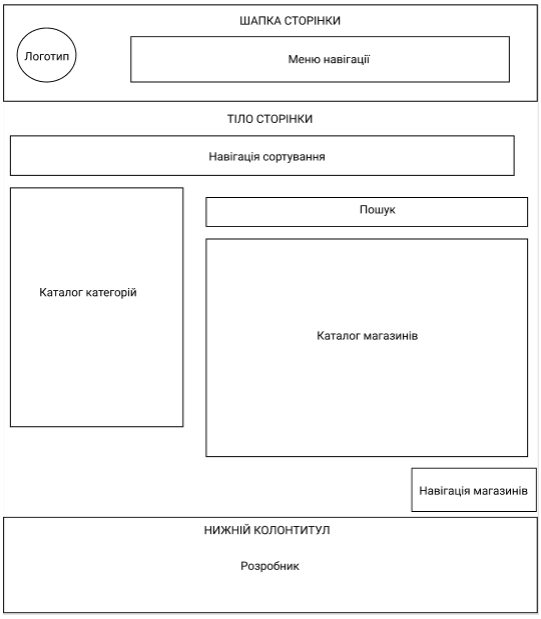

Рисунок 2.12 – Структура сторінки «Каталог»

На сторінці «Каталог» зображені статичні елементи хідер та нижній колонтитул. Тіло сторінки містить інформацію про всі магазини, а також пошук по ним і навігацію по відгукам.

Структура сторінки «Улюблені» зображена на рисунку 2.13.

Рисунок 2.13 – Структура сторінки «Улюблені»

На сторінці «Улюблені» зображені статичні хідер та нижній колонтитул. Тіло сторінки містить меню навігації, каталог улюблених магазинів і навігацію по магазинам. Структура сторінки «Профіль» зображена на рисунку 2.14.

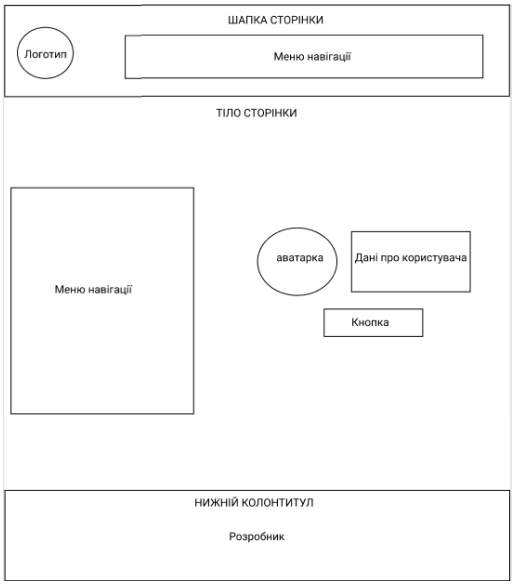

Рисунок 2.14 – Структура сторінки «Профіль»

На сторінці «Профіль» зображені статичні хідер та нижній колонтитул. Тіло сторінки містить меню навігації і інформацію про користувача. Структура сторінки «Характеристика магазину» зображена на рисунку 2.15.

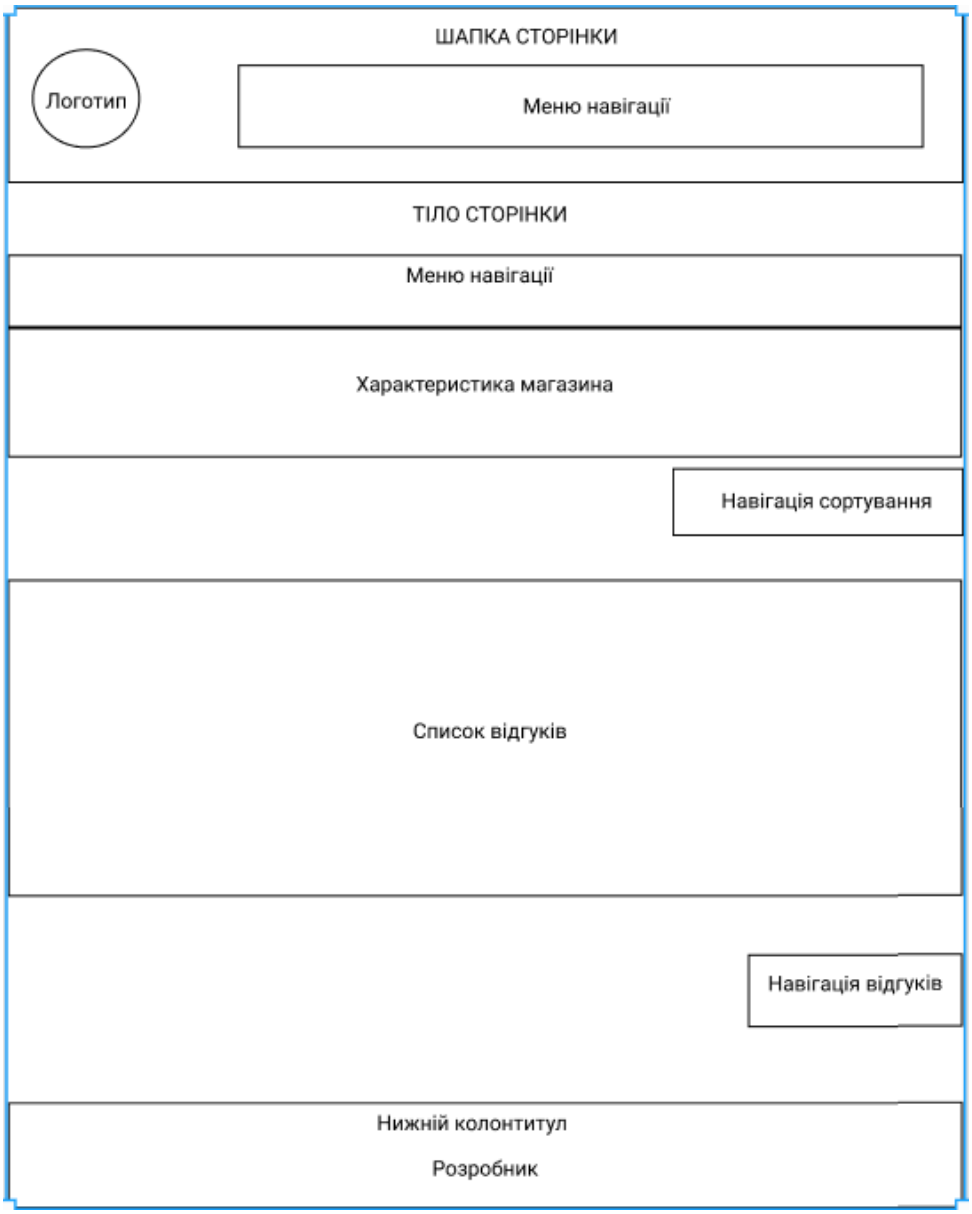

Рисунок 2.15 – Структура сторінки «Характеристика магазину»

На сторінці «Характеристика магазину» зображені статичні елементи хідер та нижній колонтитул. Тіло сторінки містить меню навігації, характеристику магазину, список відгуків і навігацію по відгукам.

Структура сторінки «Написання відгуку» зображена на рисунку 2.16.

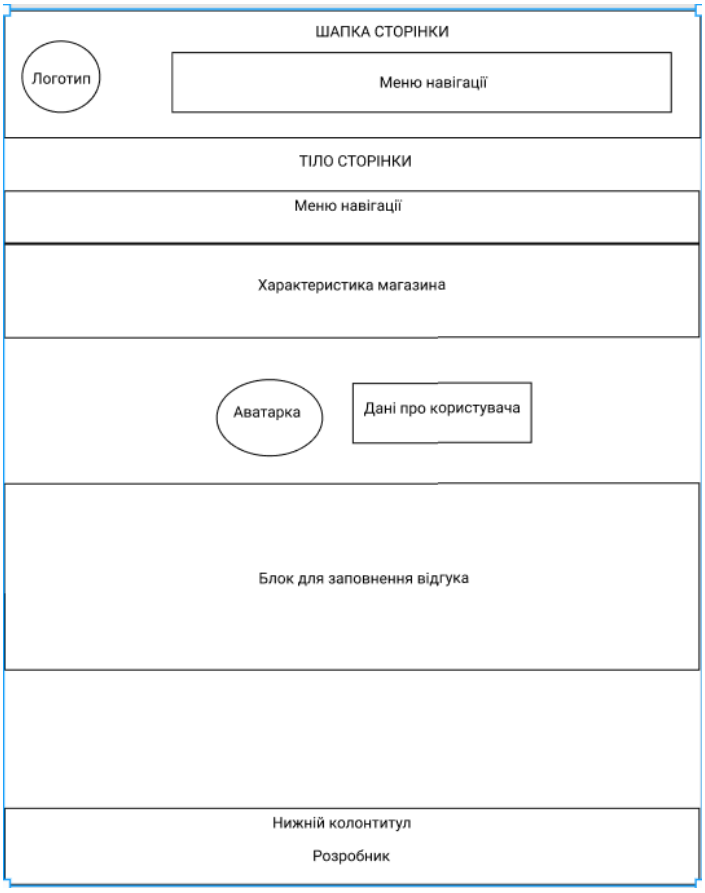

Рисунок 2.16 – Структура сторінки «Написання відгуку»

На сторінці «Написання відгуку» зображені статичні елементи хідер та нижній колонтитул. Тіло сторінки містить аватарку, характеристику користувача і блок для заповнення відгуку.

Таким чином, розглянуто структуру інтерфейсу вебсайту.

Висновок до розділу 2

Розглянемо технічне завдання дипломної роботи, яке полягає у розробці інформаційно-аналітичного сайту про магазини міста. Описано вхідні та вихідні дані системи, а також її функції.

Проведено моделювання програмного забезпечення, розроблено діаграми прецедентів та діаграми взаємодії.

Проведено моделювання даних, розроблено логічну модель бази даних та структуру таблиць бази даних. Виконано проектування інтерфейсу, розроблено структуру сторінок сайту.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Засоби розробки

Розглянемо засоби розробки, які були обрані для створення даного програмного продукту.

Java – мова програмування загального призначення. Відноситься до об’єктно-орієнтованих мов програмування, до мов з сильною типізацією. Творці реалізували принцип WORA: write once, run anywhere або «пиши один раз, запускай скрізь». Це означає, що написаний на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлена середовище виконання Java (JRE, Java Runtime Environment). Це завдання вирішується завдяки компіляції написаного на Java коду в байт-код. Цей формат виконує JVM або віртуальна машина Java. JVM – частина середовища виконання Java (JRE). Віртуальна машина не залежить від платформи. В Java реалізований механізм управління пам'яттю, який називається складальником сміття або garbage collector. Розробник створює об'єкти, а JRE за допомогою «збирача сміття» очищує пам'ять, коли об’єкти перестають використовуватися.

Фреймворк Spring Boot – це середовище на основі Java з відкритим вихідним кодом, яка використовується для створення мікросервіса. Він розроблений Pivotal Team і використовується для створення автономних і готових до використання програм.

Мова програмування JavaScript – скріптова мова, що найчастіше використовується при створенні сценаріїв поведінки браузера, що вбудовуються у вебсторінки. Хоча JavaScript має C-подібний синтаксис, але в порівнянні з мовою C має такі корінні відмінності:

- функції як об’єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне «прибирання сміття»;
- анонімні функції.

Фрейморк Angular – відкрита і вільна платформа для розробки вебдодатків, написана мовою TypeScript, що розробляється командою з компанії Google, а також спільнотою розробників з різних компаній. Angular – повністю переписаний фреймворк від тієї ж команди, яка написала AngularJS і яка є нащадком JavaScript.

Для створення програмного коду обрано такі середовища розробки як Visual Studio Code та IntelliJ IDEA.

Visual Studio Code – редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кроссплатформенної розробки веб і хмарних додатків. Включає в себе налагоджувач, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом.

IntelliJ IDEA – комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Перш за все, це середовище розробки для Java. З цією мовою вона дружить найбільше, відмінно його розуміє і допомагає в написанні розробнику. Однією з сильних сторін в JetBrains вважають підтримку широкого кола технологій. Його алгоритм прогнозування може точно передбачати, що програміст намагається набрати, і завершує його для нього, навіть якщо він не знає точного імені певного класу, члена або будь-якого іншого ресурсу. IntelliJ IDEA розроблений на основі принципу кодування, згідно з яким розробникам повинно бути дозволено писати коди з якомога меншою кількістю відволікаючих чинників.

Для зберігання коду в віддаленому репозиторії та контролю версіями, використовувалася системи контролю версій Git. Система спроєктована як набір програм, спеціально розроблених з урахуванням їх використання в сценаріях. Це дозволяє зручно створювати спеціалізовані системи контролю

версій на базі Git або призначені для користувача інтерфейси. Git підтримує швидкий поділ і злиття версій, включає інструменти для візуалізації та навігації по нелінійній історії розробки. При розробці даного проєкту в якості віддаленого репозиторію використовувався сервіс Github.

Data Transfer Object (DTO) – один з шаблонів проєктування, використовується для передачі даних між підсистемами додатку.

Компоненти DTO:

– сервіс – відповідає за зв'язок між контролером. Код компоненту service визначає, які функції викликати і за яким доменом.

– контролер – відповідає за зв'язок між service. Код компоненту controller визначає, як сайт реагує на дії користувача.

MySQL – це система управління базами даних, яка поширюється як вільне програмне забезпечення (користувачі мають право на необмежену установку, запуск, вільне використання).

Розробка вебсайту передбачає встановлення додаткових бібліотек, які є реалізацією API відповідних сервісів на мові програмування Java і JavaScript, а саме фреймворків Spring boot та Angular.

Отже, розглянуто та описано усі засоби розробки, які використовувались для реалізації вебсайту.

3.2 Документи на супроводження програмного забезпечення

3.2.1 Інструкція програмісту

Даний проєкт присвячений розробці вебсайту «Shop Rate».

Структура проєкту:

Вміст папки «adminka»:

– папка add містить файл стилів сторінки для додавання даних;

– папка admin містить файл стилів сторінки для відображення таблиць;

– папка headerforadmin містить файл стилів сторінки для верхнього колонтитула;

– папка footerforadmin містить файл стилів сторінки для нижнього колонтитула;

– папка update містить файл стилів сторінки для оновлення даних.

Папка «all-responce» містить файл стилів сторінки для виводу всіх відгуків.

Папка «exeption» містить файл стилів сторінки для помилки 404.

Папка «favourite» містить файл стилів сторінки для виводу всіх улюблених магазинів.

Папки «footer» і «header» містять файли стилів сторінки для нижнього і верхнього колонтитулів.

Папка «katalog» містить файл стилів сторінки для виводу всіх магазинів.

Папка «login» містить файл стилів сторінки для авторизації користувача.

Папка «main» містить файл стилів сторінки для головної сторінки.

Папка «profile» містить файл стилів сторінки для профілю користувача.

Папка «registrashion» містить файл стилів сторінки для реєстрації користувача.

Папка «registrashion» містить файл стилів сторінки для реєстрації користувача.

Папка «Services»містить наступні файли:

– папка Services містить сервіси отримання/відправлення на сервер;

– папка Classes містить файли класів для отримання/відправлення на сервер.

Папка «shop-stats» містить файл стилів сторінки для повної характеристики магазину.

Папка «update-account-login-and-password» містить файл стилів сторінки для змінення логіну і паролю користувача.

Папки «update-account-setting» містить файл стилів сторінки для змінення дані про користувача.

Папки «writeresponce» містить файл стилів сторінки для написання відгука.

Інші файли:

– app.component.css містить стилі для головної сторінки;

– app.component.hml містить показ для головної сторінки;

– app.component.ts містить програмний код для головної сторінки;

– app.module.ts містить програмний код для переходу по домену.

Мінімальні вимоги для програміста: розуміння шаблону DTO, змістовні знання мови програмування Java I JavaScript, розуміння, що таке API, та як з ним працюють. Розуміння фреймворку Spring boot і Angular.

Проєкт, який розробляється розміщений, на віддаленому репозиторію Github. Для того, щоб розгорнути його на своїй робочій машині, потрібно виконати наступні дії.

Клонувати проєкт з віддаленого репозиторію. Для цього необхідно в командному рядку ввести наступну команду:

git clone <url адреса репозиторію>

Після клонування потрібно встановити всі додаткові бібліотеки. При розробці будь-яких Angular за допомогою системи керування пакунками npm потрібно встановити ці бібліотеки. Команда для встановлення:

npm install @angular/cli

Також для роботи з Spring boot потрібно встановити Java, яку можна завантажити з офіційного сайту Oracle.

Завершальним етапом є налаштування з’єднання Spring boot з базою даних. У файлі application.properties потрібно заповнити тип з’єднання, address, port, назву бази даних та дані для авторизації. Приклад application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3307/infopage
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
server.port=8080
```

server.address=192.168.1.10

Даний файл не відстежується системою Git, і тому під час отримання актуальної версії проєкту зміни у файлі не відбуваються.

Також потрібно налаштувати сайт, де викликається IP адрес сервера з бази даних. Він знаходиться ../src/app/Services/*.service.ts. В цих сервісах є поле «ipv4», яке відповідає за виклик функціоналу з серверу бази даних. Необхідно встановити той самий адрес, як і у сервера.

Для того, щоб запустити проєкт (сайт), потрібно ввести таку команду:

ng serve --host=0.0.0.0 --disable-host-check

Після чого проєкт запуститься по IP адресу, який було вказано.

Після завершення всіх цих етапів можна приступати безпосередньо до розробки.

Весь програмний код наведений у додатках.

3.2.2 Інструкція користувачу

Для перегляду інформації сайту необхідно наявність програми-браузера.

При завантаженні сайту користувачеві відображається головна сторінка, яка зображена на рисунку 3.1. Вона містить логотип, меню-навігацію та каталог категорій. Для зручного переходу в каталог є кнопка, яка посилається на каталог магазинів.

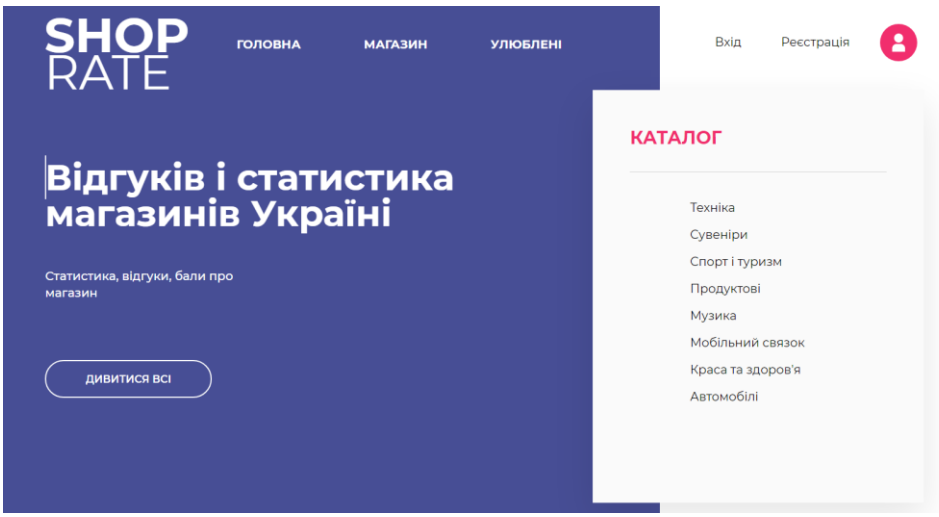

Рисунок 3.1 – Головна сторінка

Також на головній сторінці зображено популярні магазини (рисунок 3.2). Блок з основною інформацією про магазин, кількість балів, відгуки і можливістю одразу перейти на сторінку повною характеристикою магазину. Також є стрілки для гортання магазинів.

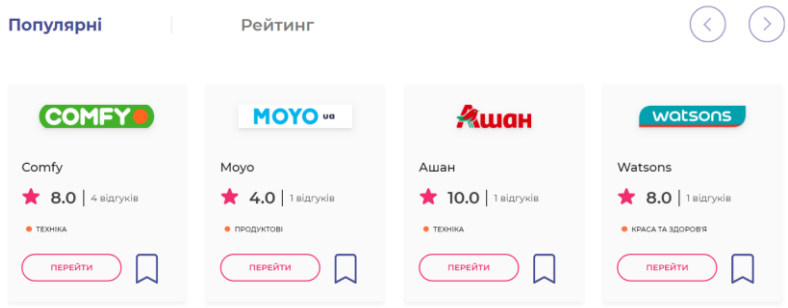

Рисунок 3.2 – Популярні магазини.

Є можливість переглянути рейтинг магазинів інформація про магазини зображено у відсортованому за оцінками порядку (рис. 3.4).

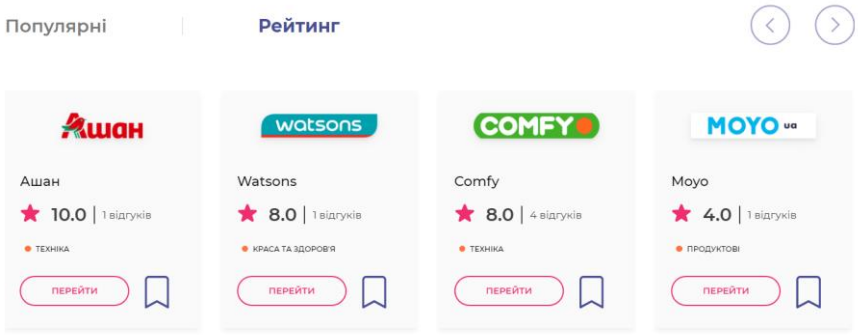

Рисунок 3.2 – Рейтинг магазинів

Користувач сайту може авторизуватися на цьому сайті. Вікно авторизації зображено на рисунку 3.3.

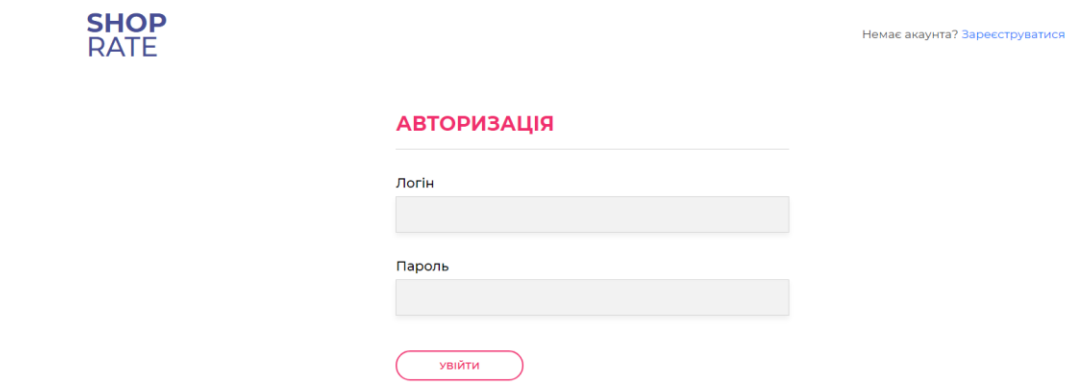

Рисунок 3.3 – Авторизація користувача

У разі відсутності акаунту користувача його потрібно зареєструвати. Для цього необхідно натиснути кнопку реєстрації у верхньому правому куті сторінки. В результаті відкриється сторінка реєстрації, яка зображена на рисунку 3.4.

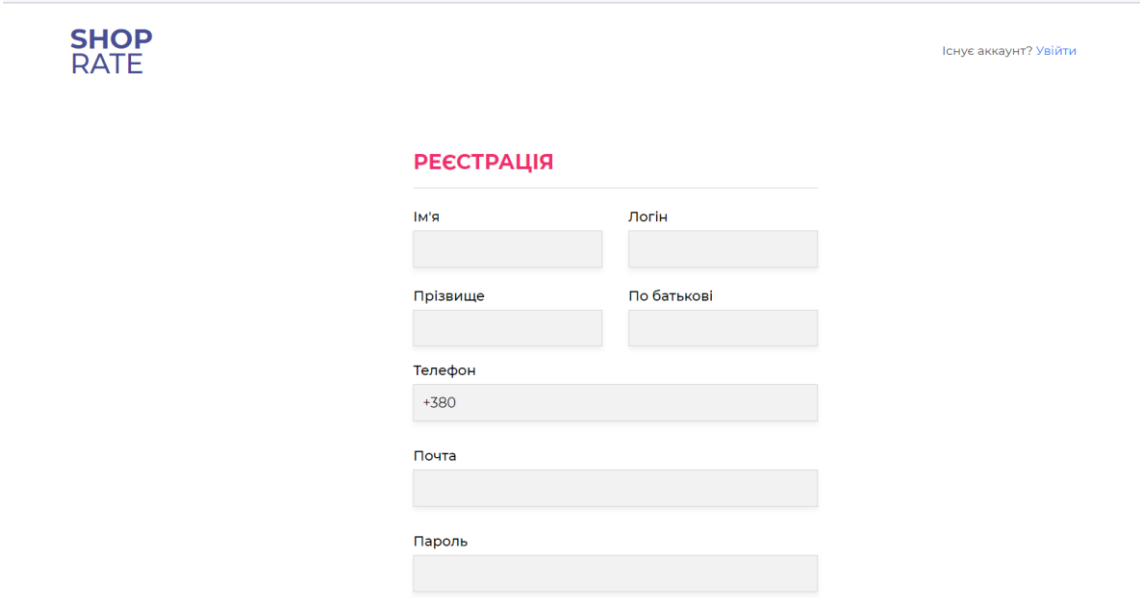

Рисунок 3.4 – Реєстрація користувача

Після успішної авторизації користувач може переглядати сторінку свого профіля. Вигляд сторінки зображено на рисунку 3.5.

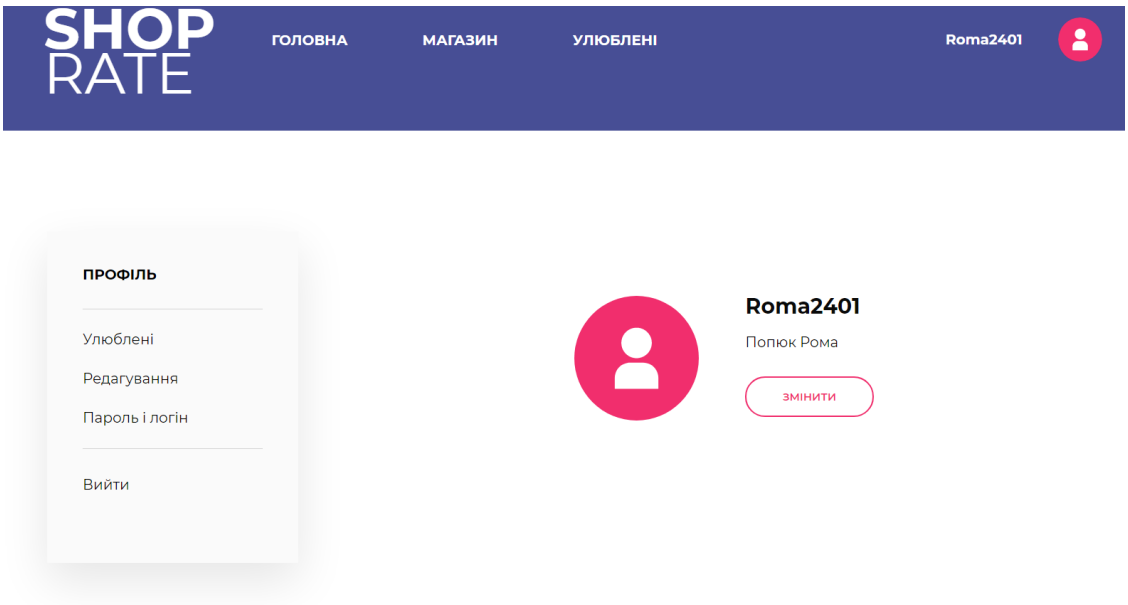

Рисунок 3.5 – Успішна авторизація

Передбачено можливість редагування профілю. Для цього існують відповідні пункти меню «Редагування» та «Пароль і логін». Також редагування профілю можливе при натисканні на аватарку користувача (викликається модальне меню). Сторінка редагування профілю та модальне меню розміщені на рисунку 3.6.

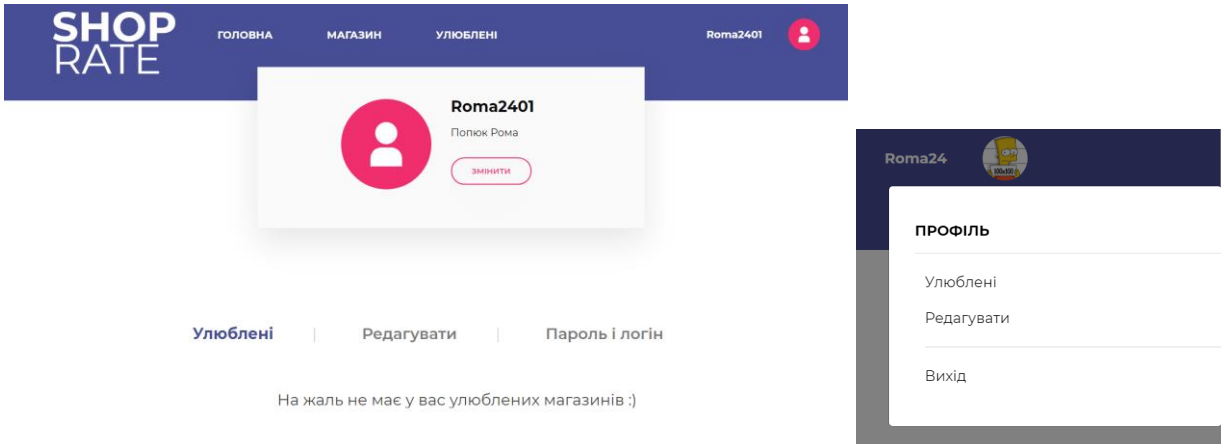

Рисунок 3.6 – Сторінка редагування профілю

Для користувача є можливість переглядати каталог магазинів, інформація про які зберігається у базі, та обирати серед них улюблені. Каталог магазинів, доступний для користувача, розміщено на рисунку 3.7.

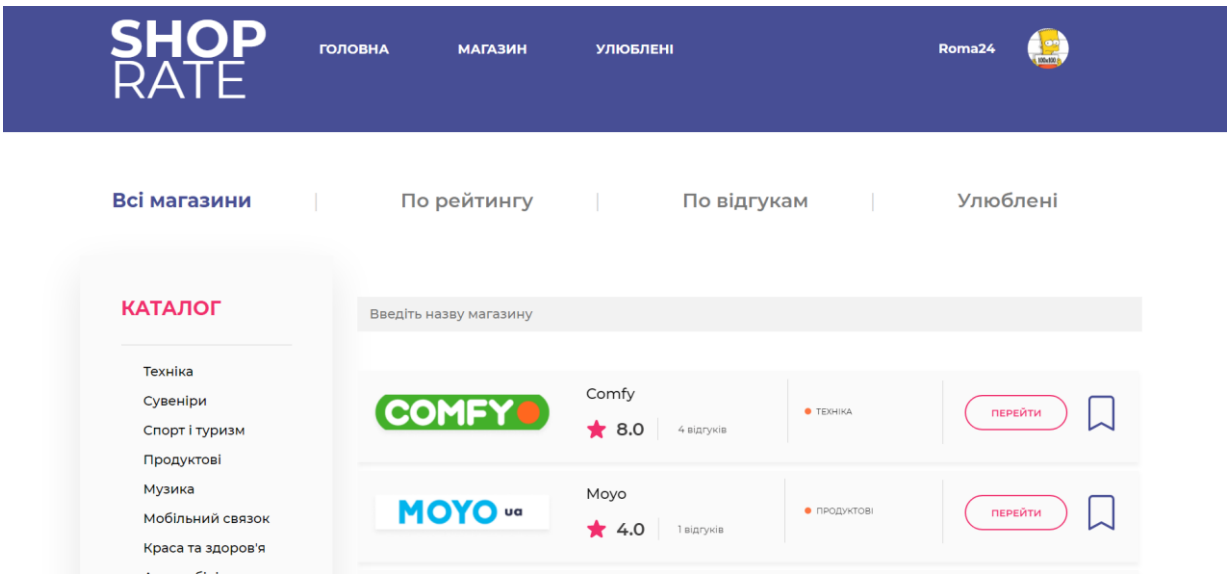

Рисунок 3.7 – Каталог магазинів для користувача

На даній сторінці є пошук магазинів за назвою, розподіл за категоріями, рейтинг і відгуки. Для навігації призначені відповідні кнопки переходу.

Для того щоб додати до улюбленого магазину, потрібно натиснути на вкладку і вибиває наступне повідомлення (рисунок 3.8).

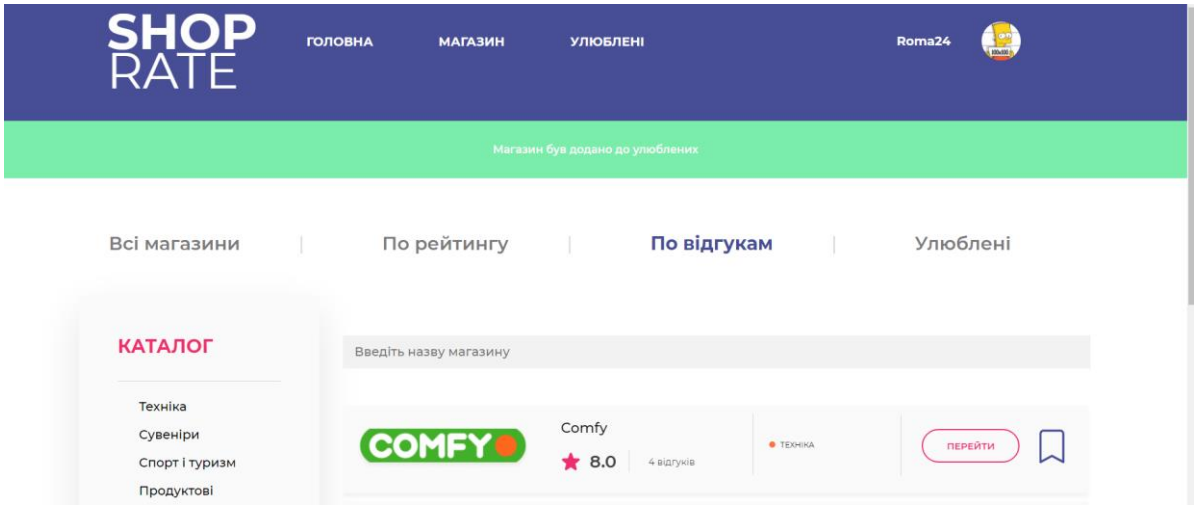

Рисунок 3.8 – Додавання до улюблених магазинів

Після того як додали, улюблений магазин пропадає з каталогу магазинів і з’являється вкладці улюблені (рисунок 3.9).

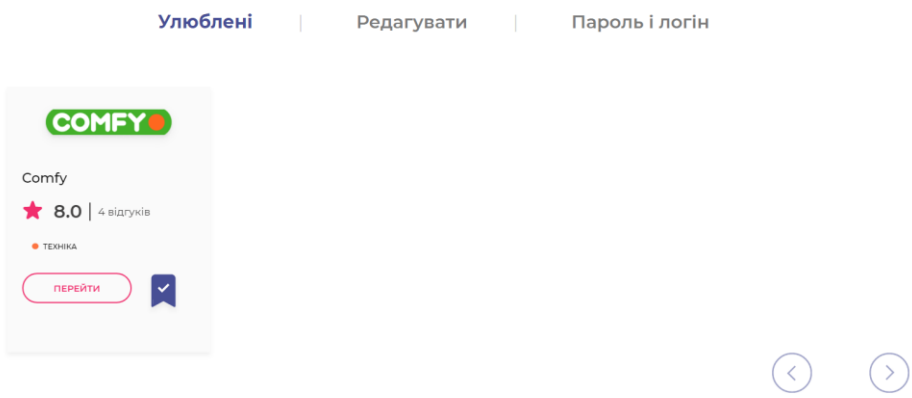

Рисунок 3.9 – Улюблені магазини

Для повної характеристики магазину потрібно натиснути кнопку «Перейти». Сторінка з повною інформацією про магазин зображена на рисунку 3.10.

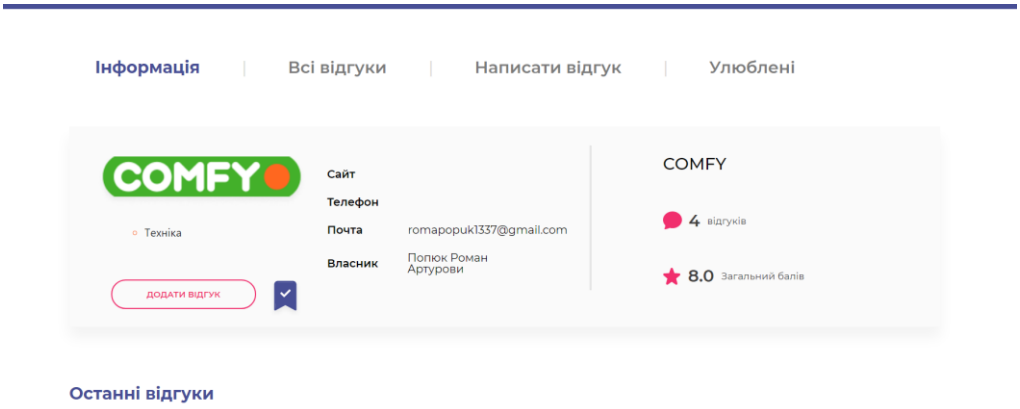

Рисунок 3.10 – Повна інформація про магазин

Сторінка для відображення відгуків зображена на рисунку 3.11.

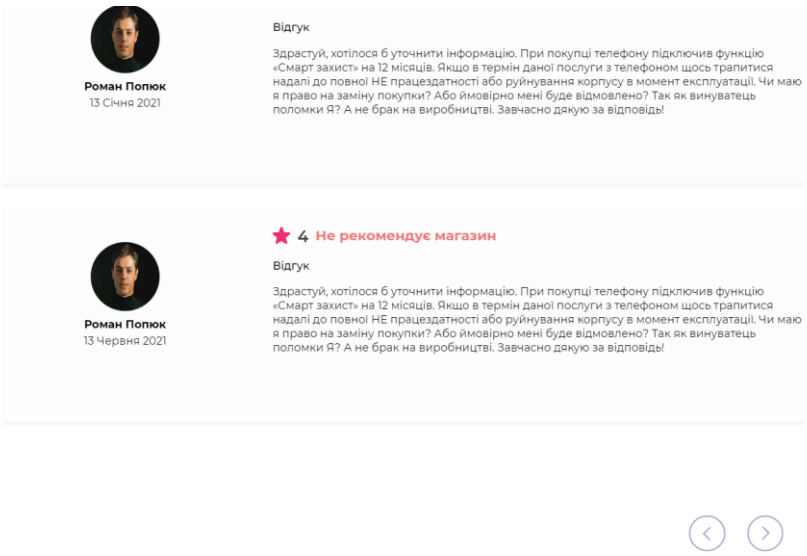

Рисунок 3.11 – Відгуки про магазин

Для того, щоб написати відгук, потрібно натиснути кнопку «Додати відгук». В результаті відкриється сторінка для виставлення оцінки діяльності магазину (рисунок 3.12).

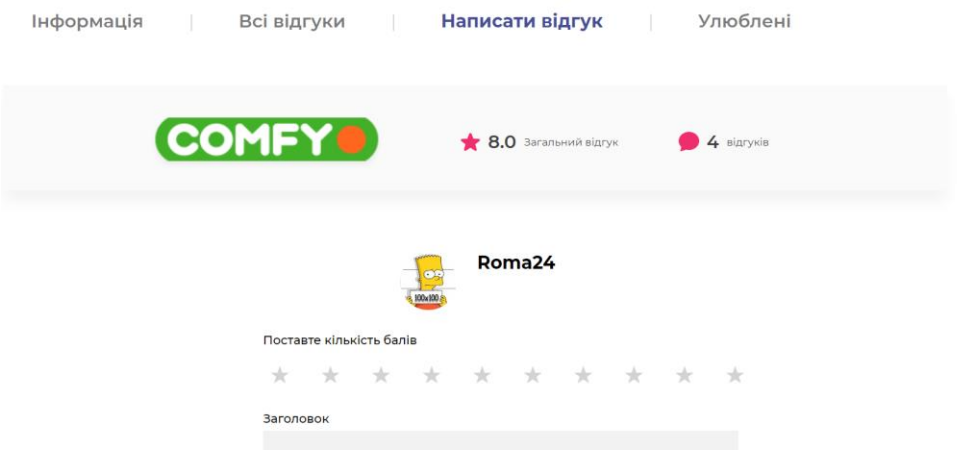

Рисунок 3.12 – Оцінювання діяльності магазину

Таким чином, розглянуто інструкцію користувачеві.

3.3 Опис реалізації системи

Розроблений сайт виконує обробку та аналіз даних про магазини, зокрема, проводить вибірку, сортування та візуалізацію даних, а також обчислює статистичні характеристики даних. Як статистичні характеристики

вибірок даних (значень оцінок магазинів) використано їх математичне сподівання (середнє значення оцінки), дисперсію та середнє квадратичне відхилення.

Обчислення дисперсії вибірки величин (стовпця оцінок магазинів) виконується таким чином. Якщо є N величин x_i, де i = 1, 2,... N, то їх дисперсія дорівнює

$$D = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

де середнє значення сукупності дорівнює:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i .$$

Середнім квадратичним відхиленням σ вибірки називається корінь із дисперсії

$$\sigma = \sqrt{D} .$$

Програмно це реалізовано таким чином:

Спочатку отримуються всі оцінки до кожного магазину:

```
@Query("Select r.ball FROM Review  r")
   List<Integer> getRatings();
```

Далі приводимо дисперсію до коду програмування:

```
List<Integer> ratings = IReviewRepository.getRatings();
        List<StatisticsResult> resultsList = new ArrayList<>();
        for (int rating : ratings) {
           int sum = rating;
           double mean = (double) sum / rating;
           double squaredDifferencesSum = Math.pow(rating - mean, 2);
           double variance = squaredDifferencesSum / rating;
           double standardDeviation = Math.sqrt(variance);
           resultsList.add(new StatisticsResult(variance,standardDeviation));
        }
```

return resultsList;

Після чого результат відправляєтьсяї на сайт, де виводиться на екран для адміністратора.

Отримання на сайті:

public Derivastion():Observable<StatisticResult>{
returnthis.http.get<StatisticResult>('http://'+this.ipv4+'/admin/reviews/derevastion'
)

Виведення:

```
<td>{{data.derevastion?.variance}}</td>
<td>{{data.derevastion?.standardDeviation}}</td>
```

Також візуалізується графік, який показує кількість відгуків та їх дату.

Для початку отримуються дані з БД:

```
@Query("SELECT COUNT(r.data), r.data FROM Review r GROUP BY
r.data")
List<Object[]> getDataWithCount();
```

Після чого ми обробляємо їх:

```
List<Object[]> result = IReviewRepository.getDataWithCount();
    result = result.subList(0, Math.min(result.size(), 1000)); // Limit the number
of results to 1000
    return result;
```

Після цього відпраляються дані на сайт і демонструються:

```
public ChartStats():Observable<any>{
  return this.http.get<any>('http://'+this.ipv4+'/admin/chart')
 }
```

Виводиться графік:

```
const labels = data.map(item=>  item[1]);
    const values = data.map(item => item[0]);
    const documentStyle = getComputedStyle(document.documentElement);
 const textColor = documentStyle.getPropertyValue('--text-color');
```

```
  const textColorSecondary = documentStyle.getPropertyValue('--text-color-secondary');
  const surfaceBorder = documentStyle.getPropertyValue('--surface-border');
  this.basicData = {
      labels: labels,
      datasets: [
          {
              label: 'Stats write review for shop',
              data: values,
              backgroundColor: ['rgba(255, 159, 64, 0.2)', 'rgba(75, 192, 192, 0.2)', 'rgba(54, 162, 235, 0.2)', 'rgba(153, 102, 255, 0.2)'],
              borderColor: ['rgb(255, 159, 64)', 'rgb(75, 192, 192)', 'rgb(54, 162, 235)', 'rgb(153, 102, 255)'],
              borderWidth: 1
          }
      ]
  };
  this.basicOptions = {
      plugins: {
          legend: {
              labels: {
                  color: textColor
              }
          }
      },
      scales: {
          y: {
              beginAtZero: true,
              ticks: {
                  color: textColorSecondary
```

```
            },
            grid: {
                color: surfaceBorder,
                drawBorder: false
            }
        },
        x: {
            ticks: {
                color: textColorSecondary
            },
            grid: {
                color: surfaceBorder,
                drawBorder: false
            }
        }
    }
};
```

Вищеописані коди забезпечують зчитування, аналіз та збереження даних. і весь код є у додатку.

3.4 Тестування програмного продукту

Тестування програмного забезпечення – це процес, що використовується для виміру якості розроблюваного програмного забезпечення. Зазвичай, поняття якості обмежується такими поняттями як коректність, повнота, безпечність, але може містити більше технічних вимог, які описані в стандарті ISO 9126.

Об’єктом випробувань є розроблений вебсайт.

Метою випробувань є виявлення програмних помилок та неточностей в роботі при обробці вхідних даних.

В якості прикладів для тестування можна розглянути процес формування списку популярних магазинів та формування рейтингу, а також процес валідації вхідних даних при авторизації та реєстрації користувача.

Розглянемо спочатку процес формування списку популярних магазинів, які визначаються за кількістю відгуків.

Результат цього процесу зображено на рисунку 3.16.

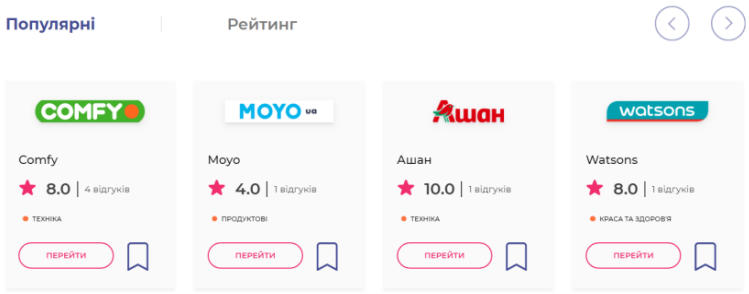

Рисунок 3.16 – Список популярних магазинів

Рейтинг магазинів будується на основі оцінювання діяльності магазинів. Результат побудови рейтингу зображений на рисунку 3.17.

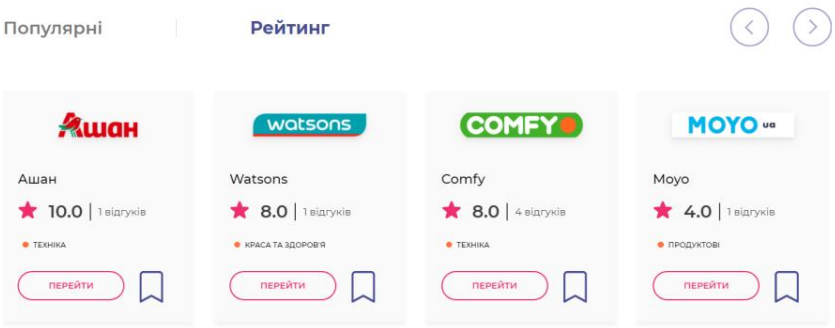

Рисунок 3.17 – Рейтинг магазинів

Аналізуючи отримані результати, можна зробити висновок, що дані процеси працюють коректно.

Протестуємо тепер процес перевірки вхідних даних при авторизації та реєстрації. Для логіну необхідно ввести не менше трьох символів. У противному випадку поле буде виділятися червоним. Це продемонстровано на рисунку 3.18.

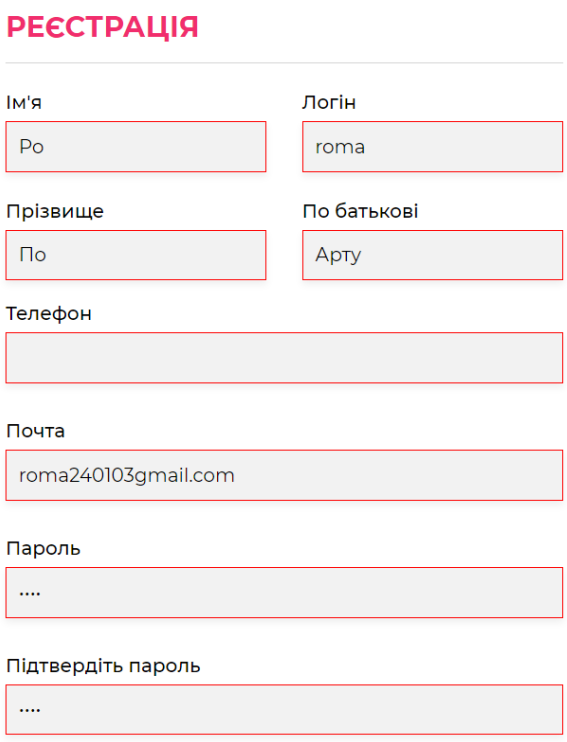

Рисунок 3.18 – Введення некоректних даних

Введені коректно дані зображено на рисунку 3.19.

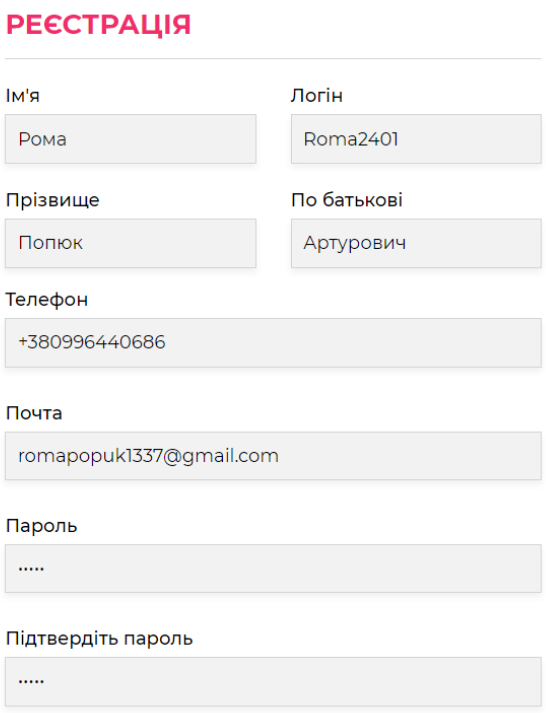

Рисунок 3.19 – Введення коректних даних

У випадку некоректного вводу даних реєстрації система виводить повідомлення про помилку, яке зображено на рисунку 3.20.

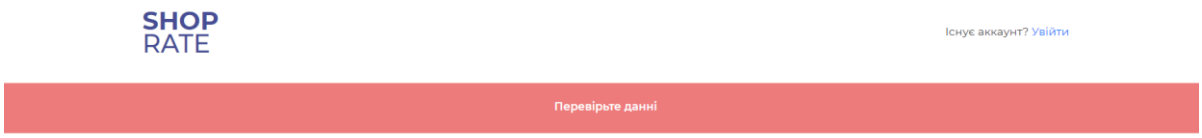

Рисунок 3.20 – Помилка реєстрації

Отже, процес валідації даних при реєстрації працює коректно.

Протестуємо тепер процес авторизації. Після правильних даних при авторизації система відкриває сторінку з профілем, яка зображена на рисунку 3.21.

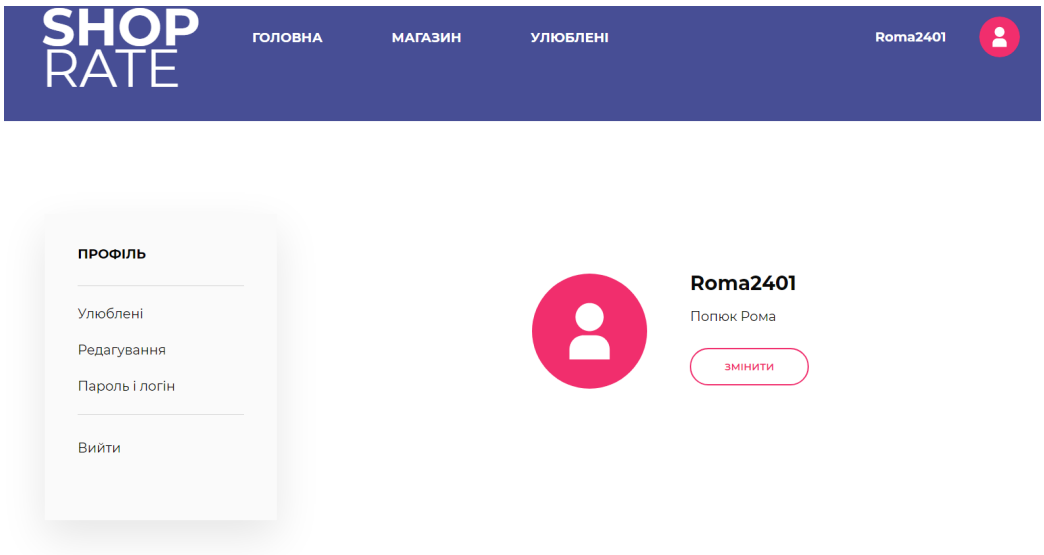

Рисунок 3.21 – Успішна авторизація, перехід на профіль користувача

В противному випадку відображається повідомлення про необхідність перевірити дані (рисунок 3.22).

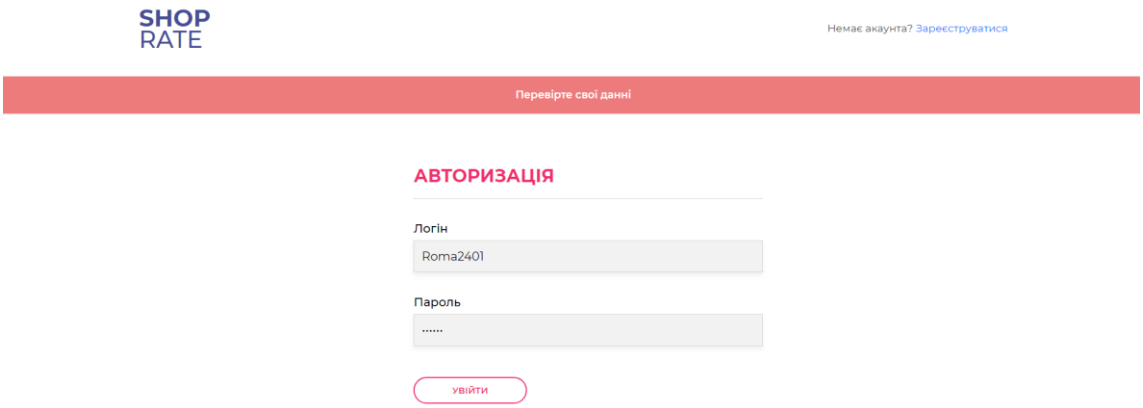

Рисунок 3.22 – Помилка при авторизації

Якщо авторизуватися як адміністратор з правильними даними логіна і пароля, то система завантажує сторінку адміністратора. Якщо ж ввести невірні логін і пароль, система переведе на сторінку «404» (рисунок 3.23).

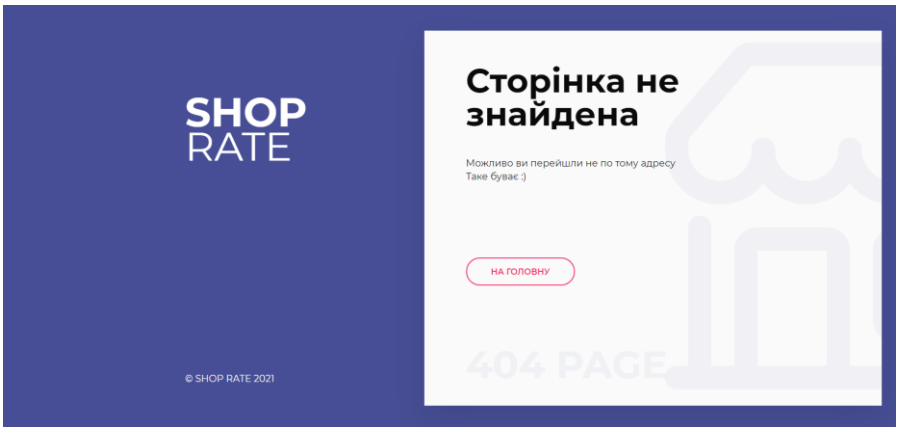

Рисунок 3.23 – Помилка 404

Отриманий результат проходження усіх тестових завдань відповідає очікуваному, з чого можна зробити висновок, що розроблена система працює коректно.

Висновки до розділу 3

Виконано програмну реалізацію інформаційно-аналітичного сайту «Shop Rate» на мові Java. Для створення програмного коду обрано такі середовища розробки як Visual Studio Code та IntelliJ IDEA. Дані сайту

зберігаються в базі даних MySQL. Розробка вебсайту передбачає встановлення додаткових бібліотек, які є реалізацією API відповідних сервісів на мові програмування Java і JavaScript, а саме фреймворків Spring boot та Angular.

Розроблено інструкцію користувача для роботи з сайтом.

Розроблений сайт виконує обробку та аналіз даних про магазини, зокрема, проводить вибірку, сортування та візуалізацію даних, а також обчислює статистичні характеристики даних. Як статистичні характеристики вибірок даних (значень оцінок магазинів) використано їх математичне сподівання (середнє значення оцінки), дисперсію та середнє квадратичне відхилення.

Тестування сайту показало правильне виконання всіх його функцій.

ВИСНОВКИ

1. У результаті аналізу сайти-аналогів Price.ua і E-katalog.ua зроблено висновок про потребу у розробці власного інформаційно-аналітичного сайту з інформацією про торговельні заклади (магазини) міста.
2. Описано вхідні та вихідні дані системи, а також її функції. Проведено моделювання програмного забезпечення, розроблено діаграми прецедентів та діаграми взаємодії. Проведено моделювання даних, розроблено логічну модель бази даних та структуру таблиць бази даних. Виконано проектування інтерфейсу, розроблено структуру сторінок сайту.
3. Виконано програмну реалізацію інформаційно-аналітичного сайту «Shop Rate» на мові Java. Для створення програмного коду обрано такі середовища розробки як Visual Studio Code та IntelliJ IDEA. Дані сайту зберігаються в базі даних MySQL. Розробка вебсайту передбачає встановлення додаткових бібліотек, які є реалізацією API відповідних сервісів на мові програмування Java і JavaScript, а саме фреймворків Spring boot та Angular.
4. Розроблений сайт надає користувачеві інформацію про магазини заданого міста. Зокрема, користувачі сайту можуть обмінюватися відгуками про магазини, своїми оцінками стосовно їх роботи, на основі яких створюється рейтинг магазинів міста. Передбачено створювати списки улюблених магазинів для заданого користувача.
5. Створено макет веб-сайту у додатку для веб-дизайнерів «Figma». Самостійно розглянуто сервіс GitHub, на його платформі створений репозиторій з дипломною роботою. Тестування сайту показало правильне виконання всіх його функцій.
6. У подальшому для даного вебресурсу можна додати новини для магазинів та акції, які в них проводяться. Для користувачів можна додати сповіщення про новини і акції улюблених магазинів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дудзяний І.М. Об’єктно-орієнтоване моделювання програмних систем: Навч. посібник. – Видавничий центр ЛНУ ім. І. Франка, 2007. – 108 с.
2. Бегун А.В., Камінський О.Є. Web-програмування. Навч. посіб. – КНЕУ, 2011. – 436 с.
3. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. – «Магнолія», 2011. – 456 с.
4. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. – Львів: Магнолія 2006, 2016. – 438 с.
5. Глинський Я.М., Ряжська В.А. Інтернет. Сервіси, HTML і web-дизайн. Навчальний посібник. – «Деол», 2003. – 184 с.
6. Грайворонський М.В., Новіков О.М. Безпека інформаційно-комунікаційних систем. – Видавнича група ВНУ, 2009. – 608 с.
7. Форум з зразками коду [Електронний ресурс]. – Режим доступу: https://www.stackoverflow.com/
8. Гончаров С.М., Кушнір Н.Б. Практикум з маркетингу. – «Центр учбової літератури», 2012. – 208 с.
9. Ярцев В.П. Організація баз даних та знань: навч. посібник. – К.: ДУТ 2018. – 214 с.
10. Elmasri R., Navathe S.B. Fundamentals of Database Systems. – Hoboken, USA: Pearson, 2016. – 1273 p.
11. Perkins L., Redmond E., Wilson J.R. Seven Databases in Seven Weeks. A Guide to Modern Databases and the NoSQL Movement. – Raleigh, North Carolina, USA: The Pragmatic Bookshelf, 2018. – 354 p.
12. Лосєв М.Ю., Федько В. В. Бази даних: навчально-практичний посібник для самостійної роботи студентів. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 233 с.
13. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Ужгород, ДВНЗ "УжНУ", 2018. – 118 с.
14. MySQL. [Electronic resource]. - Access mode : https://www.mysql.com

15. Microsoft SQL documentation. [Electronic resource]. - Access mode : https://docs.microsoft.com/uk-ua/sql/t-sql/functions/functions?view=sql-server-ver15

ДОДАТКИ

Додаток А. Лістинги програми

A.1. Spring Boot(BackEnd)

A.1.1 UserService

```
@Autowired
IUserRepository IUserRepository;
@Autowired
AdminService adminService;
@Autowired
IInfoShopRepository IInfoShopRepository;
@Autowired
ICategoryRepository ICategoryRepository;
@Autowired
IFavouriteRepository IFavouriteRepository;
@Autowired
IReviewRepository IReviewRepository;
public void saveImageShop(MultipartFile file, Long id) {
   InfoShop infoShop = IInfoShopRepository.findById(id).orElseThrow(() -> new
RuntimeException("Магази не знайдено"));
   String filename = StringUtils.cleanPath(file.getOriginalFilename());
   if (filename.contains("..")) {
      new Exception("Error");
   }
   try {
      infoShop.setLogo(Base64.getEncoder().encodeToString(file.getBytes()));
   } catch (IOException e) {
      e.printStackTrace();
   }
   IInfoShopRepository.save(infoShop);
}
public void saveImage(MultipartFile file, String username) {
   User user = IUserRepository.findByUserName(username).orElseThrow(() ->
         new UsernameNotFoundException("Користувача не знайдено"));
   String filename = StringUtils.cleanPath(file.getOriginalFilename());
   if (filename.contains("..")) {
      new Exception("Error");
   }
   try {
      user.setLogo(Base64.getEncoder().encodeToString(file.getBytes()));
   } catch (IOException e) {
      e.printStackTrace();
   }
   IUserRepository.save(user);
}
public void setInfo(String username, UserRequestUpdateData userRequestUpdateData) {
   User user = IUserRepository.findByUserName(username).orElseThrow(() ->
         new UsernameNotFoundException("Користувача не знайдено"));
   user.setName(userRequestUpdateData.getName());
   user.setLastname(userRequestUpdateData.getLastname());
   user.setSurname(userRequestUpdateData.getSurname());
   user.setLogo(userRequestUpdateData.getLogo());
   user.setPhone(userRequestUpdateData.getPhone());
   IUserRepository.save(user);
}
public void setInfoLoginAndPassword(String username, UserRequestLoginAndPassword
```

```
userRequestLoginAndPassword) {
    User user = IUserRepository.findByUserName(username).orElseThrow(() ->
            new UsernameNotFoundException("Користувача не знайдено"));
    user.setUserName(userRequestLoginAndPassword.getLogin());
    user.setPassword(userRequestLoginAndPassword.getPassword());
    IUserRepository.save(user);
}
public void deleteImg(String username) {
    User user = IUserRepository.findByUserName(username).orElseThrow(() ->
            new UsernameNotFoundException("Користувача не знайдено"));
    user.setLogo(null);
    IUserRepository.save(user);
}
public UserRequestUpdateData getUser(String username) {
    User user = IUserRepository.findByUserName(username).orElseThrow(() ->
            new UsernameNotFoundException("Користувача не знайдено"));
    UserRequestUpdateData userRequestUpdateData = new UserRequestUpdateData();
    userRequestUpdateData.setName(user.getName());
    userRequestUpdateData.setLogo(user.getLogo());
    userRequestUpdateData.setSurname(user.getSurname());
    userRequestUpdateData.setLastname(user.getLastname());
    userRequestUpdateData.setPhone(user.getPhone());
    return userRequestUpdateData;
}
public InfoShopRequest infoShopRequest(InfoShop infoShop) {
    InfoShopRequest infoShopRequest = new InfoShopRequest();
    infoShopRequest.setId(infoShop.getId());
    infoShopRequest.setName_shop(infoShop.getName_shop());
    infoShopRequest.setId_Kategory(infoShop.getId_Kategory());
    infoShopRequest.setWebsite(infoShop.getWebsite());
    infoShopRequest.setEmail(infoShop.getEmail());
    infoShopRequest.setContacts_people(infoShop.getContacts_people());
    infoShopRequest.setPhone(infoShop.getPhone());
    infoShopRequest.setLogo(infoShop.getLogo());
    return infoShopRequest;
}
public List<InfoShopRequest> getFavourite(String username) {
    User user = IUserRepository.findByUserName(username).orElseThrow(() ->
            new UsernameNotFoundException("Користувача не знайдено"));
    Favourite[] favourites = IFavouriteRepository.findAllByUserId(user).toArray(new Favourite[0]);

    List<InfoShopRequest> infoShopRequests = new ArrayList<>();
    for (Favourite favourite : favourites) {
        InfoShop infoShop = favourite.getId_shop();
        Review[] reviewShop = IReviewRepository.findAllById(infoShop).toArray(new Review[0]);
        InfoShopRequest infoShopRequest = infoShopRequest(infoShop);
        if (reviewShop.length > 0) {
            int sum = 0, count = 0;
            for (Review entity : reviewShop) {
                count++;
                sum = sum + entity.getBall();
            }
            infoShopRequest.setBall(sum / count);
            infoShopRequest.setResponse(count);
        } else {
            infoShopRequest.setBall(0);
            infoShopRequest.setResponse(0);
```

```
        }
        infoShopRequest.setSubscribe(true);
        infoShopRequest.setSubscribeID(favourite.getId());
        infoShopRequests.add(infoShopRequest);
    }
    return infoShopRequests;
}
public void deleteFavourite(Long id) {
    IFavouriteRepository.delete(IFavouriteRepository.findById(id).orElseThrow(() ->
            new UsernameNotFoundException("Улюблені не знайдено")));
}
public List<InfoShopRequest> selectAllShop() {
    List<InfoShop> infoShops = IInfoShopRepository.findAll();
    return infoShops.stream().map(infoShop -> {
        InfoShopRequest infoShopRequest = new InfoShopRequest();
        Review[] reviewShop = IReviewRepository.findAllById(infoShop).toArray(new Review[0]);
        int count = reviewShop.length;
        int sum = Arrays.stream(reviewShop).mapToInt(Review::getBall).sum();
        double ball = count > 0 ? (double) sum / count : 0;
        infoShopRequest.setId(infoShop.getId());
        infoShopRequest.setId_Kategory(infoShop.getId_Kategory());
        infoShopRequest.setWebsite(infoShop.getWebsite());
        infoShopRequest.setName_shop(infoShop.getName_shop());
        infoShopRequest.setContacts_people(infoShop.getContacts_people());
        infoShopRequest.setEmail(infoShop.getEmail());
        infoShopRequest.setPhone(infoShop.getPhone());
        infoShopRequest.setLogo(infoShop.getLogo());
        infoShopRequest.setBall((int)ball);
        infoShopRequest.setResponse(count);
        return infoShopRequest;
    }).collect(Collectors.toList());
}
public List<Kategory> selectallKategory() {
   return ICategoryRepository.findAll();
}
public InfoShopRequest findShop(Long id) {
    InfoShop infoShop = IInfoShopRepository.findById(id)
            .orElseThrow(() -> new UsernameNotFoundException("Магазин не знайдено"));
    Review[] reviewShop = IReviewRepository.findAllById(infoShop).toArray(new Review[0]);
    int count = reviewShop.length;
    int sum = Arrays.stream(reviewShop).mapToInt(Review::getBall).sum();
    InfoShopRequest infoShopRequest = new InfoShopRequest();
    infoShopRequest.setBall(count > 0 ? sum / count : 0);
    infoShopRequest.setResponse(count);
    infoShopRequest.setId(infoShop.getId());
    infoShopRequest.setName_shop(infoShop.getName_shop());
    infoShopRequest.setLogo(infoShop.getLogo());
    infoShopRequest.setPhone(infoShop.getPhone());
    infoShopRequest.setEmail(infoShop.getEmail());
    infoShopRequest.setContacts_people(infoShop.getContacts_people());
    infoShopRequest.setWebsite(infoShop.getWebsite());
    infoShopRequest.setId_Kategory(infoShop.getId_Kategory());
    return infoShopRequest;
}
public InfoShopRequest findShopForUser(Long id, String username) {
    InfoShop infoShop = IInfoShopRepository.findById(id)
            .orElseThrow(() -> new UsernameNotFoundException("Магазин не знайдено"));
```

```
    User user = IUserRepository.findByUserName(username)
            .orElseThrow(() -> new UsernameNotFoundException("Користувача не знайдено"));
    Review[] reviewShop = IReviewRepository.findAllById(infoShop).toArray(new Review[0]);
    Favourite[] favourites = IFavouriteRepository.findAllByUserIdAndIdShop(user,
infoShop).toArray(new Favourite[0]);
    InfoShopRequest infoShopRequest = new InfoShopRequest();
    infoShopRequest.setId(infoShop.getId());
    infoShopRequest.setName_shop(infoShop.getName_shop());
    infoShopRequest.setLogo(infoShop.getLogo());
    infoShopRequest.setPhone(infoShop.getPhone());
    infoShopRequest.setEmail(infoShop.getEmail());
    infoShopRequest.setContacts_people(infoShop.getContacts_people());
    infoShopRequest.setWebsite(infoShop.getWebsite());
    infoShopRequest.setId_Kategory(infoShop.getId_Kategory());
    int sum = 0, count = 0;
    for (Review entity : reviewShop) {
        count++;
        sum += entity.getBall();
    }
    infoShopRequest.setBall(count == 0 ? 0 : sum / count);
    infoShopRequest.setResponse(count);
    for (Favourite favourite : favourites) {
        if (favourite.getId_shop().getId().equals(id)) {
            infoShopRequest.setSubscribe(true);
            infoShopRequest.setSubscribeID(id);
            break;
        }
    }
    return infoShopRequest;
}
public List<InfoShopRequest> findByKategory(Long id){
    Kategory kategory = ICategoryRepository.findById(id)
            .orElseThrow(() -> new UsernameNotFoundException("Магазин не знайдено"));
    InfoShop[] infoShops = IInfoShopRepository.findAllByKategoryName(kategory)
            .toArray(new InfoShop[0]);
    return Arrays.stream(infoShops)
            .map(infoShop -> {
                InfoShopRequest infoShopRequest = new InfoShopRequest();
                Review[] reviewShop = IReviewRepository.findAllById(infoShop).toArray(new Review[0]);
                int count = reviewShop.length;
                int sum = Arrays.stream(reviewShop).mapToInt(Review::getBall).sum();
                int avgRating = count > 0 ? sum / count : 0;
                infoShopRequest.setId(infoShop.getId());
                infoShopRequest.setId_Kategory(infoShop.getId_Kategory());
                infoShopRequest.setContacts_people(infoShop.getContacts_people());
                infoShopRequest.setEmail(infoShop.getEmail());
                infoShopRequest.setLogo(infoShop.getLogo());
                infoShopRequest.setName_shop(infoShop.getName_shop());
                infoShopRequest.setPhone(infoShop.getPhone());
                infoShopRequest.setWebsite(infoShop.getWebsite());
                infoShopRequest.setBall(avgRating);
                infoShopRequest.setResponse(count);
                return infoShopRequest;
            })
            .collect(Collectors.toList());
}
```

```
public List<InfoShopRequest> findByKategoryForUser(Long id, String username) {
   Kategory kategory = ICategoryRepository.findById(id)
         .orElseThrow(() -> new UsernameNotFoundException("Магазин не знайдено"));
   User user = IUserRepository.findByUserName(username)
         .orElseThrow(() -> new UsernameNotFoundException("Користувача не знайдено"));
   List<Favourite> favourites = IFavouriteRepository.findAllByUserId(user);
   List<InfoShop> infoShops = IInfoShopRepository.findAllByKategoryName(kategory);
   List<InfoShopRequest> infoShopRequests = new ArrayList<>();
   for (InfoShop infoShop : infoShops) {
      InfoShopRequest infoShopRequest = new InfoShopRequest();
      int sum = 0, count = 0;
      for (Favourite favourite : favourites) {
         if (favourite.getId_shop().getId().equals(infoShop.getId())) {
            infoShopRequest.setSubscribeID(infoShop.getId());
            infoShopRequest.setSubscribe(true);
            break;
         }
      }
      Review[] reviewShop = IReviewRepository.findAllById(infoShop).toArray(new Review[0]);
      if (reviewShop.length > 0) {
         for (Review entity : reviewShop) {
            count++;
            sum = sum + entity.getBall();
         }
         infoShopRequest.setBall(sum / count);
         infoShopRequest.setResponse(count);
      } else {
         infoShopRequest.setBall(0);
         infoShopRequest.setResponse(0);
      }
      infoShopRequest.setId(infoShop.getId());
      infoShopRequest.setId_Kategory(infoShop.getId_Kategory());
      infoShopRequest.setContacts_people(infoShop.getContacts_people());
      infoShopRequest.setEmail(infoShop.getEmail());
      infoShopRequest.setLogo(infoShop.getLogo());
      infoShopRequest.setName_shop(infoShop.getName_shop());
      infoShopRequest.setPhone(infoShop.getPhone());
      infoShopRequest.setWebsite(infoShop.getWebsite());
      infoShopRequests.add(infoShopRequest);
   }
   return infoShopRequests;
}
public void favourite(String username, Long id) {
   User user = IUserRepository.findByUserName(username).orElseThrow(() ->
         new UsernameNotFoundException("Користувача не знайдено"));
   InfoShop infoShop = IInfoShopRepository.findById(id).orElseThrow(() ->
         new UsernameNotFoundException("Магазин не знайдено"));
   Favourite favourite = new Favourite();
   favourite.setId_shop(infoShop);
   favourite.setId_users(user);
   IFavouriteRepository.save(favourite);
}

public InfoShopRequest[] getAllShops(String username) {
   List<InfoShopRequest> subscribesList = getFavourite(username);
   Set<Long> subscribedIds =
subscribesList.stream().map(InfoShopRequest::getId).collect(Collectors.toSet());
```

```
    List<InfoShopRequest> allshops = selectAllShop();
    List<InfoShopRequest> result = allshops.stream()
            .filter(s -> !subscribedIds.contains(s.getId()))
            .collect(Collectors.toList());
    return result.toArray(new InfoShopRequest[0]);
}
```

A.1.2 ReviewService

```
@Service
public class ReviewService {
@Autowired
ReviewRepository reviewRepository;
@Autowired
InfoShopRepository infoShopRepository;
@Autowired
UserRepository userRepository;
public void addReview(Long id, String username, ReviewDto reviewDto){
UserEntity userEntity =userRepository.findByUserName(username).orElseThrow(() ->
new UsernameNotFoundException("Користувача не знайдено"));
InfoShop infoShop = infoShopRepository.findById(id).orElseThrow(() -> new
RuntimeException("Магази не знайдено"));
ReviewEntity reviewEntity = new ReviewEntity();
reviewEntity.setId_shop(infoShop);
reviewEntity.setId_user(userEntity);
reviewEntity.setText(reviewDto.getText());
reviewEntity.setHeader(reviewDto.getHeader());
reviewEntity.setBall(reviewDto.getBall());
reviewEntity.setData(reviewDto.getData());
reviewRepository.save(reviewEntity);
}
public List<ReviewRequest>  findReviewById(Long id){
InfoShop infoShop = infoShopRepository.findById(id).orElseThrow(() -> new
RuntimeException("Магази не знайдено"));
ReviewEntity[] reviewEntities = reviewRepository.findAllByShopId(infoShop).toArray(new
ReviewEntity[0]);
ReviewRequest[] reviewRequests = new ReviewRequest[reviewEntities.length];
for(int i=0;i<reviewEntities.length;i++){
ReviewRequest reviewRequest= new ReviewRequest();
reviewRequest.setId(reviewEntities[i].getId());
reviewRequest.setId_shop(reviewEntities[i].getId_shop().getId());
reviewRequest.setText(reviewEntities[i].getText());
reviewRequest.setBall(reviewEntities[i].getBall());
reviewRequest.setUsername(reviewEntities[i].getId_user().getUserName());
reviewRequest.setLogo(reviewEntities[i].getId_user().getLogo());
reviewRequest.setName(reviewEntities[i].getId_user().getName());
reviewRequest.setHeader(reviewEntities[i].getHeader());
reviewRequest.setSurname(reviewEntities[i].getId_user().getSurname());
reviewRequest.setData(reviewEntities[i].getData());
reviewRequests[i]=reviewRequest;
}
return Arrays.asList(reviewRequests);
}
}
```

A.1.3 AuthService

```
@Service
public class AuthService {
@Autowired
private UserRepository userRepository;
```

```
@Autowired
private PasswordEncoder passwordEncoder;
@Autowired
private AuthenticationManager authenticationManager;
@Autowired
private JwtProvider jwtProvider;
@Autowired
private RoleRepository roleRepository;
@Autowired
private AdminService adminService;
public void signup(RegisterRequest registerRequest) {
UserEntity user = new UserEntity();
user.setUserName(registerRequest.getUsername());
user.setEmail(registerRequest.getEmail());
user.setPassword(encodePassword(registerRequest.getPassword()));
user.setName(registerRequest.getName());
user.setLastname(registerRequest.getLastname());
user.setSurname(registerRequest.getSurname());
user.setPhone(registerRequest.getPhone());
user.setRoles(Arrays.asList(roleRepository.findByName("ROLE_USER")));
userRepository.save(user);
}
public void UpdateLoginAndPassword(String username, UserRequestLoginAndPassword
userRequestLoginAndPassword){
UserEntity userEntity = userRepository.findByUserName(username).orElseThrow(() ->
new UsernameNotFoundException("Користувача не знайдено"));
PasswordEncoder passencoder = new BCryptPasswordEncoder();
if(passencoder.matches(userRequestLoginAndPassword.getOldpassword(),userEntity.getPassword())){
userEntity.setPassword(encodePassword(userRequestLoginAndPassword.getPassword()));
userEntity.setUserName(userRequestLoginAndPassword.getLogin());
}
userRepository.save(userEntity);
}
private String encodePassword(String password) {
return passwordEncoder.encode(password);
}
public AuthenticationResponse login(LoginRequest loginRequest) {
Authentication authenticate = authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(loginRequest.getUsername(),
loginRequest.getPassword()));
SecurityContextHolder.getContext().setAuthentication(authenticate);
String authenticationToken = jwtProvider.generateToken(authenticate);
UserEntity user = userRepository.findByUserName(loginRequest.getUsername()).orElseThrow(()->
new UsernameNotFoundException("Користувача не знайдено"));
Collection<RoleEntity> role = user.getRoles();
RoleEntity[] roles = user.getRoles().toArray(new RoleEntity[0]);
RoleDto[] roleDto = new RoleDto[roles.length];
for(int i=0;i<roles.length;i++){
roleDto[i] = adminService.roleEntityToDto(roles[i]);
}
return new AuthenticationResponse(authenticationToken,
loginRequest.getUsername(),roleDto[0].getName());
}
}
```

A.1.4 AdminService

```
   @Autowired
   IUserRepository IUserRepository;
```

```
@Autowired
IRoleRepository IRoleRepository;
@Autowired
private PasswordEncoder passwordEncoder;
@Autowired
ICategoryRepository ICategoryRepository;
@Autowired
IInfoShopRepository IInfoShopRepository;
@Autowired
IReviewRepository IReviewRepository;
public List<Kategory> selectallKategory() {
   return ICategoryRepository.findAll();
}
public List<InfoShopRequest> selectAllShop() {
   int sum = 0, count = 0;
   InfoShop[] infoShops = IInfoShopRepository.findAll().toArray(new InfoShop[0]);
   InfoShopRequest[] infoShopRequests = new InfoShopRequest[infoShops.length];
   for (int i = 0; i < infoShops.length; i++) {
      InfoShopRequest infoShopRequest = new InfoShopRequest();
      Review[] reviewShop = IReviewRepository.findAllById(infoShops[i]).toArray(new Review[0]);
      if(reviewShop.length>0){
         for (Review entity : reviewShop) {
            count++;
            sum = sum + entity.getBall();
         }
         infoShopRequest.setBall(sum/count);
         infoShopRequest.setResponse(count);
         infoShopRequest.setId(infoShops[i].getId());
         infoShopRequest.setId_Kategory(infoShops[i].getId_Kategory());
         infoShopRequest.setWebsite(infoShops[i].getWebsite());
         infoShopRequest.setName_shop(infoShops[i].getName_shop());
         infoShopRequest.setContacts_people(infoShops[i].getContacts_people());
         infoShopRequest.setEmail(infoShops[i].getEmail());
         infoShopRequest.setPhone(infoShops[i].getPhone());
         infoShopRequest.setLogo(infoShops[i].getLogo());
         infoShopRequests[i] = infoShopRequest;
         count = 0;
         sum = 0;
      }
      else{
         infoShopRequest.setBall(0);
         infoShopRequest.setResponse(0);
         infoShopRequest.setId(infoShops[i].getId());
         infoShopRequest.setId_Kategory(infoShops[i].getId_Kategory());
         infoShopRequest.setWebsite(infoShops[i].getWebsite());
         infoShopRequest.setName_shop(infoShops[i].getName_shop());
         infoShopRequest.setContacts_people(infoShops[i].getContacts_people());
         infoShopRequest.setEmail(infoShops[i].getEmail());
         infoShopRequest.setPhone(infoShops[i].getPhone());
         infoShopRequest.setLogo(infoShops[i].getLogo());
         infoShopRequests[i] = infoShopRequest;
      }
   }
   return Arrays.asList(infoShopRequests);
}
public List<StatisticsResult> derevastion() {
   List<Integer> ratings = IReviewRepository.getRatings();
```

```
        List<StatisticsResult> resultsList = new ArrayList<>();
        for (int rating : ratings) {
            int sum = rating;
            double mean = (double) sum / rating;
            double squaredDifferencesSum = Math.pow(rating - mean, 2);
            double variance = squaredDifferencesSum / rating;
            double standardDeviation = Math.sqrt(variance);
            resultsList.add(new StatisticsResult(variance,standardDeviation));
        }
        return resultsList;
    }
    public List<Review> selectallreview(){
        return IReviewRepository.findAll();
    }
    public List<UserFullInfoDto> selectalluser() {
        return IUserRepository.findAll().stream().map(user -> {
            UserFullInfoDto dto = new UserFullInfoDto();
            dto.setId(user.getId());
            dto.setLogin(user.getUserName());
            dto.setEmail(user.getEmail());
            dto.setLastname(user.getLastname());
            dto.setName(user.getName());
dto.setPassword(Base64.getEncoder().encodeToString(user.getPassword().getBytes(StandardCharsets.UTF_8)));
            dto.setSurname(user.getSurname());
            dto.setPhone(user.getPhone());
            dto.setLogo(user.getLogo());
            return dto;
        }).collect(Collectors.toList());
    }
    public RoleDto roleEntityToDto(Role roleEntity){
        RoleDto role = new RoleDto();
//        role.setName(roleEntity.getName());
        role.setId(roleEntity.getId());
        return role;
    }
    public void AddShop( ShopDto shopDto){
        Kategory kategory = ICategoryRepository.findById(shopDto.getKategory()).orElseThrow(() ->
                new RuntimeException("Категорія не знайдено"));
        InfoShop infoShop = new InfoShop();
        infoShop.setName_shop(shopDto.getName());
        infoShop.setEmail(shopDto.getEmail());
        infoShop.setContacts_people(shopDto.getContact_people());
        infoShop.setPhone(shopDto.getPhone());
        infoShop.setWebsite(shopDto.getWebsite());
        infoShop.setId_Kategory(kategory);
        IInfoShopRepository.save(infoShop);
    }
    public void AddKategory(String kategory){
        Kategory kategory1 =  new Kategory();
        kategory1.setName_kategory(kategory);
        ICategoryRepository.save(kategory1);
    }
    public void UpdateShop(Long id,ShopDto shopDto){
        InfoShop infoShop = IInfoShopRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Магазин не знайдено"));
        Kategory kategory = ICategoryRepository.findById(shopDto.getKategory()).orElseThrow(() ->
```

```
                new RuntimeException("Категорія не знайдено"));
        infoShop.setName_shop(shopDto.getName());
        infoShop.setEmail(shopDto.getEmail());
        infoShop.setContacts_people(shopDto.getContact_people());
        infoShop.setPhone(shopDto.getPhone());
        infoShop.setWebsite(shopDto.getWebsite());
        infoShop.setId_Kategory(kategory);
        IInfoShopRepository.save(infoShop);
    }
    public KategoryDto Findkategory(Long id){
        Kategory kategory = ICategoryRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Магазин не знайдено"));
        KategoryDto kategoryDto = new KategoryDto();
        kategoryDto.setId_kategory(kategory.getId_kategory());
        kategoryDto.setName_kategory(kategory.getName_kategory());
        return kategoryDto;
    }
    public void UpdateKategory(Long id, String kategory){
        Kategory kategory1 = ICategoryRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Магазин не знайдено"));
        kategory1.setName_kategory(kategory);
        ICategoryRepository.save(kategory1);
    }
    public void DeleteShop(Long id ){
        IInfoShopRepository.delete(IInfoShopRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Магазин не знайдено")));
    }
    public void DeleteReview(Long id){
        IReviewRepository.delete(IReviewRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Відгук не знайдено")));
    }
    public void DeleteKategory(Long id){
        ICategoryRepository.delete(ICategoryRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Категорія не знайдено")));
    }
    public void DeleteUser(Long id){
        IUserRepository.delete(IUserRepository.findById(id).orElseThrow(() ->
                new RuntimeException("Категорія не знайдено")));
    }
    public StatsDTO getCountStats(){
     return  new
StatsDTO((int)IUserRepository.count(),(int)IReviewRepository.count(),(int)IInfoShopRepository.count(),
(int) ICategoryRepository.count());
    }
    public List<Object[]> getCountAndDate() {
        List<Object[]> result = IReviewRepository.getDataWithCount();
        result = result.subList(0, Math.min(result.size(), 1000)); // Limit the number of results to 1000
        for (Object[] row : result) {
            Long count = (Long) row[0];
            String data = (String) row[1];
            // Do something with the count and data
        }
        return result;
    }
}
```

A.2 Angular(FrontEnd)

A.2.1 Main

A.2.1.3 Main.ts

```
import { Route } from '@angular/compiler/src/core';
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { AuthService } from '../Services/auth.service';
import { Kategory } from '../Services/Classes/Kategory';
import { Shop } from '../Services/Classes/Shop';
import { UserDataInfo } from '../Services/Classes/UserInfo';
import { UserService } from '../Services/user.service';
@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.css']
})
export class MainComponent implements OnInit {
  isLogin:boolean;
  constructor(private authService :AuthService,private userService:UserService,private
_router:Router,private router:ActivatedRoute) { }
  user:UserDataInfo;
  kategory:Kategory[];
  result:any;
  result1:any;
  username:any;
  shop:Shop[];
  b:Array<any>;
  arr:Array<any>;
  selectedfile:File;
  isRating:boolean;
  isImg:boolean;
  img:any;
  ngOnInit(): void {
this.username = this.authService.isLogin('username');
if(this.username){
 this.UserInfo();
 this.GetAllForUser();
}else{
  this.userService.GetAllShop().subscribe(data=>{
this.shop=data;
let first_li = document.querySelectorAll(".content_link>ul>li:first-child");
let second_li = document.querySelectorAll(".content_link>ul>li:last-child");
this.router.params.subscribe(params=>{
  if(params['name']=='rating'){
this.isRating=true;
first_li[0].classList.remove("selectedLink");
second_li[0].classList.add("selectedLink");
  }
  else{
this.isRating=false;
  }
})
if(this.isRating){
  this.shop.sort(function(c,b){
if(c.ball<b.ball){
  return -1;
}
if(c.ball>b.ball){
  return 1;
```

```
}
return 0;
  })
  this.shop.reverse();
}
let a = Array<Shop[]>();
for(var i=0;i<this.shop.length;i+=4){
  a.push(this.shop.slice(i,i+4))
}
this.arr = a;
for(var j=0;j<this.shop.length;j++){
  this.shop[j].logo=''+this.shop[j].logo;
}
  })
}
  this.GetAllKategory()
}
  UserInfo(){
this.userService.UserInfo().subscribe(data=>{
  this.user = data;
  if( this.user.logo ){
this.isImg = true;
this.user.logo =''+this.user.logo;
  }
  if( !this.user.logo ){
this.isImg = false;
  }
  if(this.username != null){
this.isLogin = true;
  }
   });
  }
  AddFavourite(id:number){
if(this.username){
  this.userService.SetFavourite(id).subscribe(data=>{
   location.reload();
  })
}else{
  this._router.navigate(['login']);
}
  }
  GetAllKategory(){
this.userService.GetAllKategory().subscribe(data=>{
  this.kategory=data;
  this.kategory.sort(function(a,b){
if(a.name_kategory<b.name_kategory){
  return -1;
}
if(a.name_kategory>b.name_kategory){
  return 1;
}
return 0;
  })
  this.kategory.reverse();
})
  }
  GetAllForUser(){
```

```
this.userService.GetAllForUser().subscribe(data=>{
  this.shop=data;
  let first_li = document.querySelectorAll(".content_link>ul>li:first-child");
let second_li = document.querySelectorAll(".content_link>ul>li:last-child");
this.router.params.subscribe(params=>{
  if(params['name']=='rating'){
this.isRating=true;
first_li[0].classList.remove("selectedLink");
second_li[0].classList.add("selectedLink");
  }
  else{
this.isRating=false;
  }
})
if(this.isRating){
  this.shop.sort(function(c,b){
if(c.ball<b.ball){
  return -1;
}
if(c.ball>b.ball){
  return 1;
}
return 0;
  })
  this.shop.reverse();
}
  let a = Array<Shop[]>();
  for(var i=0;i<this.shop.length;i+=4){
a.push(this.shop.slice(i,i+4))
  }
  this.arr = a;
  for(var j=0;j<this.shop.length;j++){
this.shop[j].logo=''+this.shop[j].logo;
  }
})
  }
  LogOut(){
this.authService.Logout()
  }
  HrefToKatalog(){
this._router.navigate(['katalog'])
  }
}
```

A.2.2 Login

A.2.2.3 Login.ts

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from '../Services/auth.service';
import { LoginRequest } from '../Services/Classes/LoginRequest';
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
```

```
  FailToLogin:boolean;
  loginForm:FormGroup;
  loginrequest:LoginRequest
  constructor(private formBuilder: FormBuilder,private authService:AuthService,private _router:Router) {
this.loginForm = this.formBuilder.group({
  username:["",[Validators.required,Validators.minLength(5),Validators.maxLength(15)]],
  password:["",[Validators.required,Validators.minLength(5),Validators.maxLength(15)]],
});
this.loginrequest={
  username:'',
  password:''
  }
   }
  ngOnInit(): void {
  }
  OnLogin(){
this.loginrequest.username = this.loginForm.get('username').value;
this.loginrequest.password = this.loginForm.get('password').value;
this.authService.login(this.loginrequest).subscribe(data=>{
  if(data){
this.FailToLogin=false;
this._router.navigate(['profile'])
  }
},
(err:HttpErrorResponse)=>{
 this.FailToLogin=true;

setTimeout(() => this.FailToLogin=false, 1500)
   }
   )
  }
}
```

A.2.3 Favourite

A.2.3.3 Favourite.ts

```
import{Component,OnInit}from'@angular/core';
import{Router}from'@angular/router';
import{AuthService}from'../Services/auth.service';
import{Shop}from'../Services/Classes/Shop';
import{UserDataInfo}from'../Services/Classes/UserInfo';
import{UserService}from'../Services/user.service';
@Component({
selector:'app-favourite',
templateUrl:'./favourite.component.html',
styleUrls:['./favourite.component.css']
})
exportclassFavouriteComponentimplementsOnInit{
constructor(privateauthService:AuthService,privateuserService:UserService,privaterouter:Router){}
username:any;
user:UserDataInfo;
img:any;
shop:Shop[];
arr:Array<any>;
isImg:boolean;
ngOnInit():void{
this.username=this.authService.isLogin('username');
if(this.username){
this.UserInfo();
```

```
}
this.GetAllShop()
}
GetAllShop(){
this.userService.Favourite(this.username).subscribe(data=>{
this.shop=data;
console.log(this.shop)
leta=Array<Shop[]>();
for(vari=0;i<this.shop.length;i+=8){
a.push(this.shop.slice(i,i+8))
}
this.arr=a;
for(varj=0;j<this.shop.length;j++){
this.shop[j].logo=''+this.shop[j].logo;
}
})
}
DeleteFavourite(id:number){
this.userService.DeleteFavourite(id).subscribe(data=>{
location.reload();
})
}
UserInfo(){
this.userService.UserInfo().subscribe(data=>{
this.user=data;
this.img=''+this.user.logo;
if(this.user.logo){
this.isImg=true;
}
if(!this.user.logo){
this.isImg=false;
}
});
}
HrefToProfile(){
this.router.navigate(['accountSetting']);
}
}
```