Державний вищий навчальний заклад

«Чернівецький політехнічний коледж»

Відділення програмування та радіотехніки

Циклова комісія інженерія програмного забезпечення

Допускається до захисту

завідувач відділення

Андрій ШПАК

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до дипломного проєкту

молодшого спеціаліста

на тему: Вебсайт «Shop Rate»

Виконав: студент IV курсу, групи 541

спеціальність 121

«Інженерія програмного забезпечення»

спеціалізація

«Розробка програмного забезпечення»

Рома ПОПЮК

Керівник: Оксана КРАВЧУК

2021

**зміст**

[ВСТУП 3](#_Toc75176916)

[РОЗДІЛ 1. Передпроектні дослідження 5](#_Toc75176917)

[1.1 Аналіз предметної області 5](#_Toc75176918)

[1.2 Аналіз існуючих аналогів 7](#_Toc75176919)

[1.3 Маркетингове дослідження інновацій 12](#_Toc75176920)

[РОЗДІЛ 2. Технічний проєкт 17](#_Toc75176921)

[2.1 Технічне завдання 17](#_Toc75176922)

[2.2 Моделювання програмного забезпечення 19](#_Toc75176923)

[2.3 Моделювання даних 22](#_Toc75176924)

[2.4 Проєктування інтерфейсу 25](#_Toc75176925)

[РОЗДІЛ 3. робочий проєкт 30](#_Toc75176926)

[3.1 Засоби розробки 30](#_Toc75176927)

[3.2 Документи на супроводження програмного забезпечення 32](#_Toc75176928)

[3.3 Тестування програмного продукту 40](#_Toc75176929)

[3.4 Охорона праці 45](#_Toc75176930)

[ВИСНОВКИ 47](#_Toc75176931)

[перелік джерел 48](#_Toc75176932)

[ДОДАТОК 49](#_Toc75176933)

# ВСТУП

На сьогоднішній день можливості інтернету дозволяють всім користувачам, не покидаючи дому, знаходити необхідну інформацію з будь-якої теми. Тому сьогодні створення вебресурсів досить популярне. Вдало організований інформаційний сайт – це великий плюс як для користувачів (пошукачів інформації), так і для власників сайтів (компаній, фірм).

Інформаційні портали займають почесне місце серед сучасних вебресурсів. Інформаційний портал – це великий інтернет ресурс, який надає детальну інформацію на певну тематику. Інформаційний портал, на відміну від простого тематичного сайту, містить інструменти для взаємодії з користувачами, пошуки, рейтинги, а також дозволяють відвідувачам спілкуватися між собою, наприклад, за допомогою чату або через відгуки.

Одним із самих розповсюджених прикладів інформаційного порталу є сайти для міст, які містять новини, погоду, афіші, розклад транспорту, дошку оголошень, каталоги підприємств, магазинів і іншу корисну інформацію.

Даний дипломний проект присвячений розробці вебресурсу з інформацією про магазини міста. Зокрема, користувачі сайту матимуть змогу обмінюватися відгуками про магазини, своїми оцінками стосовно їх роботи, які в результаті утворюють рейтинг магазинів міста.

Предметом дослідження дипломного проекту є вебресурси з інформацією про магазини, їх товари і рейтинги в межах країни.

Об’єктом дослідження є основні принципи створення макету вебсайту та його розробка.

Пояснювальна записка дипломного проекту складається з переліку умовних позначень, вступу, розділу перед проектних досліджень, розділу технічного проекту, розділу робочого проекту, висновків, посилань на джерела та додатків.

У першому розділі пояснювальної записки до дипломного проекту представлено аналіз предметної області, проаналізовано програмне забезпечення аналогічного призначення та проведено маркетингове дослідження інновацій розробки.

У другому розділі вказано вимоги до програмного продукту в технічному завданні, описано процес моделювання програмного забезпечення та моделювання даних, проектування інтерфейсу.

У третьому розділі представлено опис робочого проекту: засоби розробки, керівництво програмісту та користувачу, тестування програмного продукту та інструкції з охорони праці.

Висновки містять ґрунтовний аналіз результату виконаної роботи.

Додатки пояснювальної записки містять допоміжний матеріал, необхідний для повноти сприйняття дипломного проекту.

# РОЗДІЛ 1. Передпроектні дослідження

## 1.1 Аналіз предметної області

Рейтинг – це числовий або порядковий показник успішності або популярності, який відображає важливість або вплив певного об'єкта або явища.

Існують багато різних рейтингів. Їх можна класифікувати за різними параметрами. Враховуючи область призначення, можна навести таки 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 і в тому, і в іншому випадку людям 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 безліч 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 і 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 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.

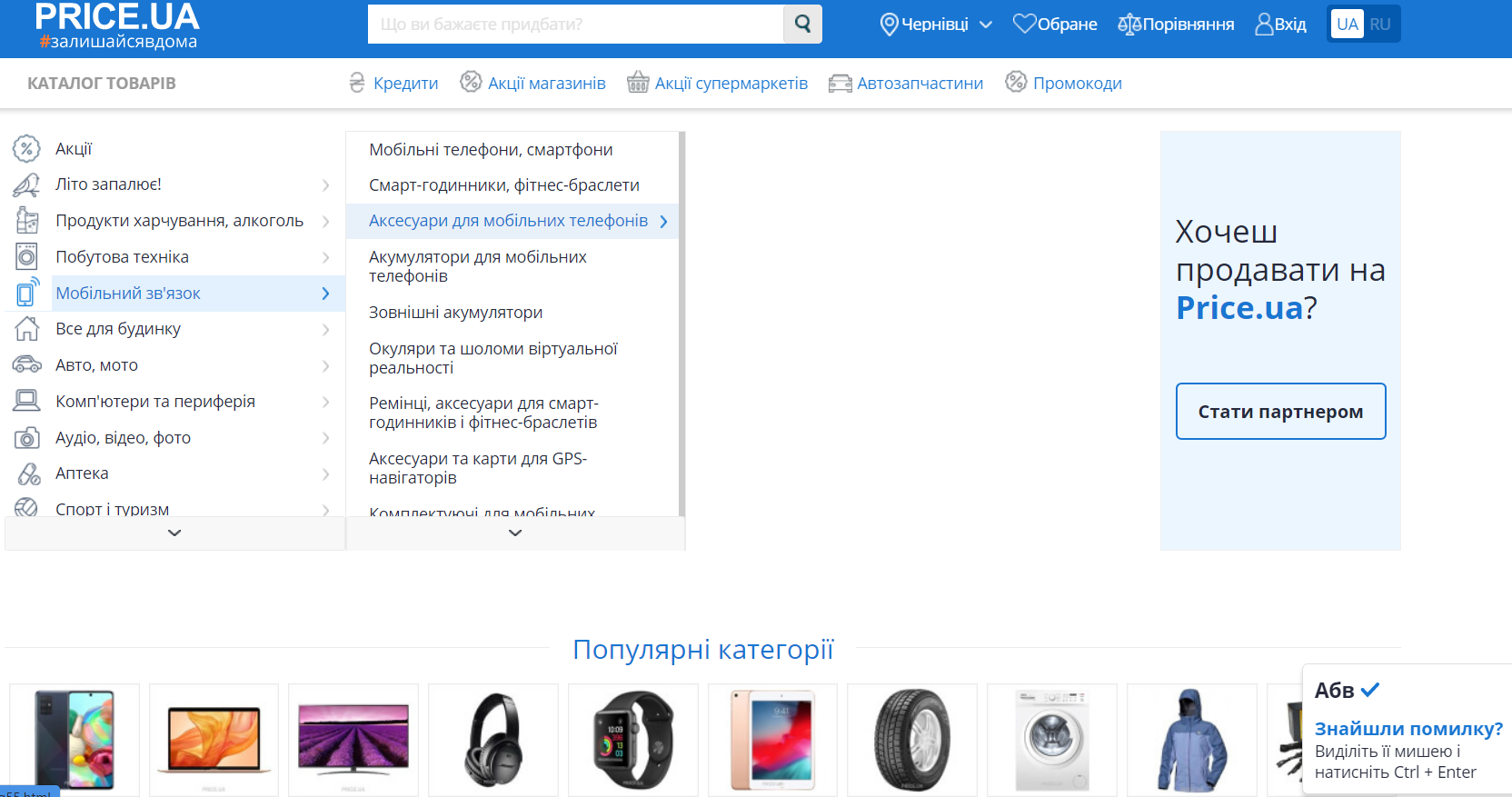


Рисунок 1.1 – Стартова сторінка Price.ua

На рисунку 1.2 зображено розділ «Акції та знижки». На даній сторінці розташована інформація про акції в різних магазинах, каталог товарів різних виробників.

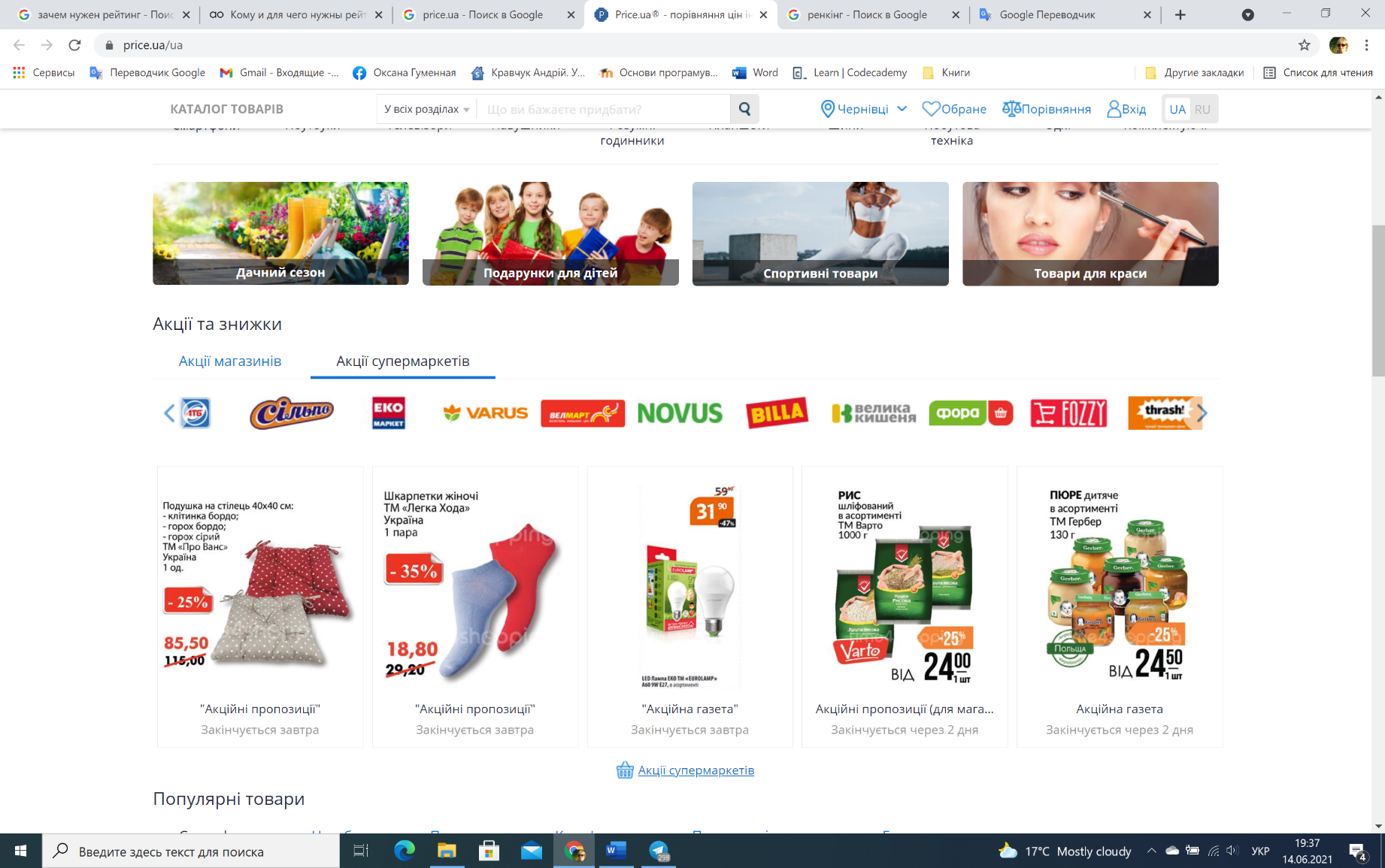


Рисунок 1.2 – Сторінка «Акції та знижки»

Аналогічний вигляд мають й інші розділи, що розміщені в нижній частині головної сторінки.

Передбачено можливість пошуку інформації про бажаний товар. Для цього призначено поле введення у верхній частині сайту (рисунок 1.3).

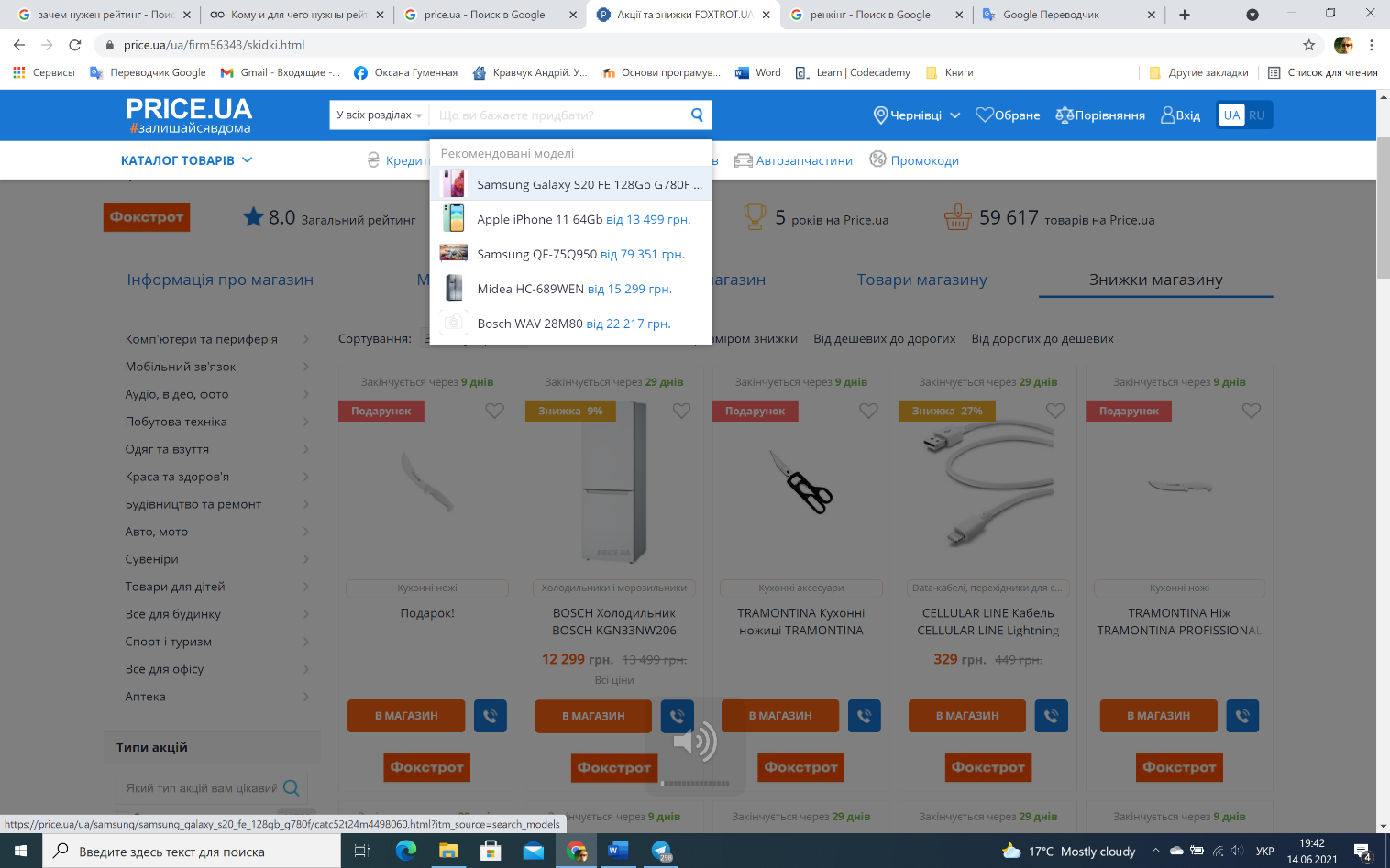


Рисунок 1.3 – Пошук інформації про бажаний товар

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

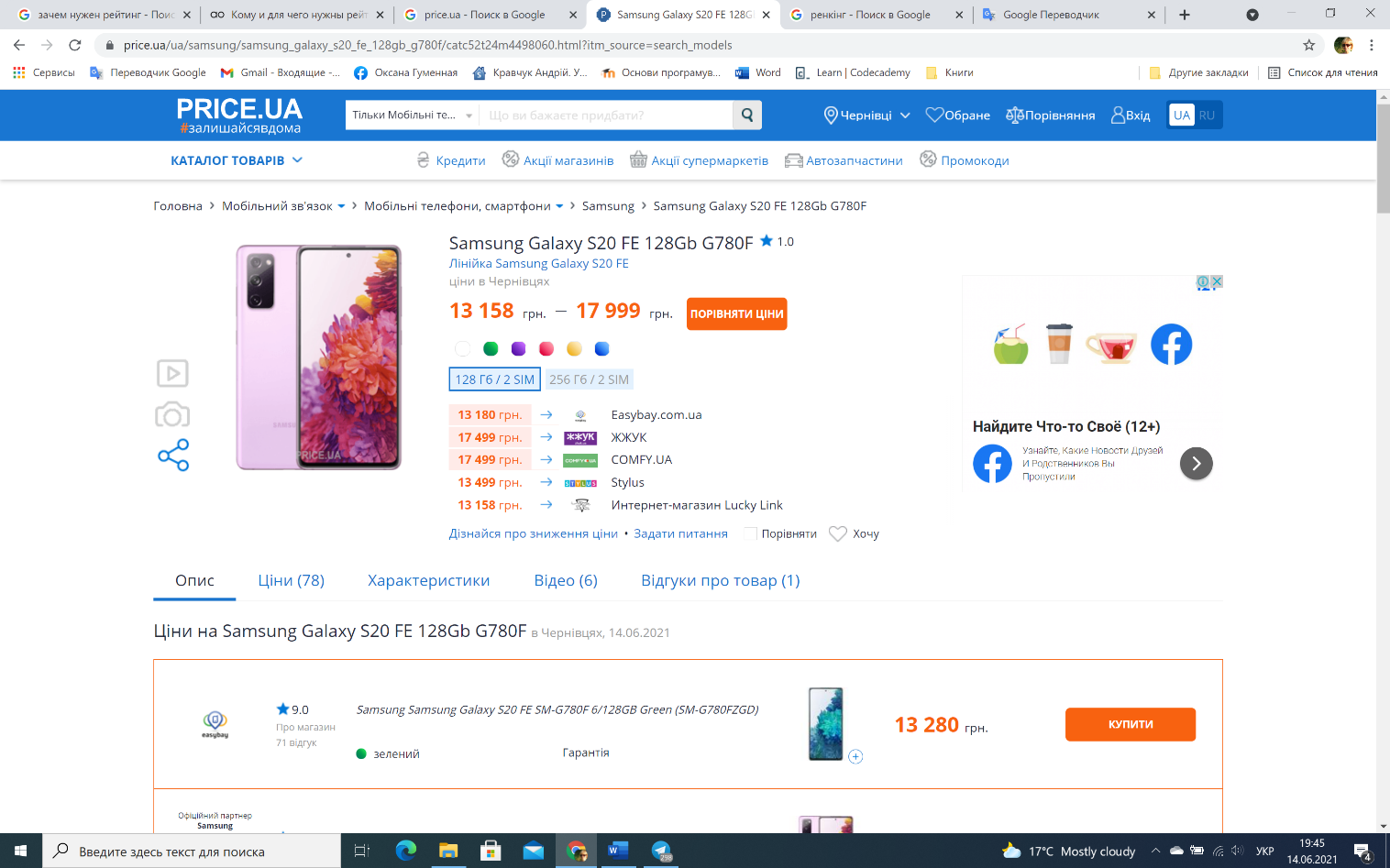


Рисунок 1.4 – Результат пошуку

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.

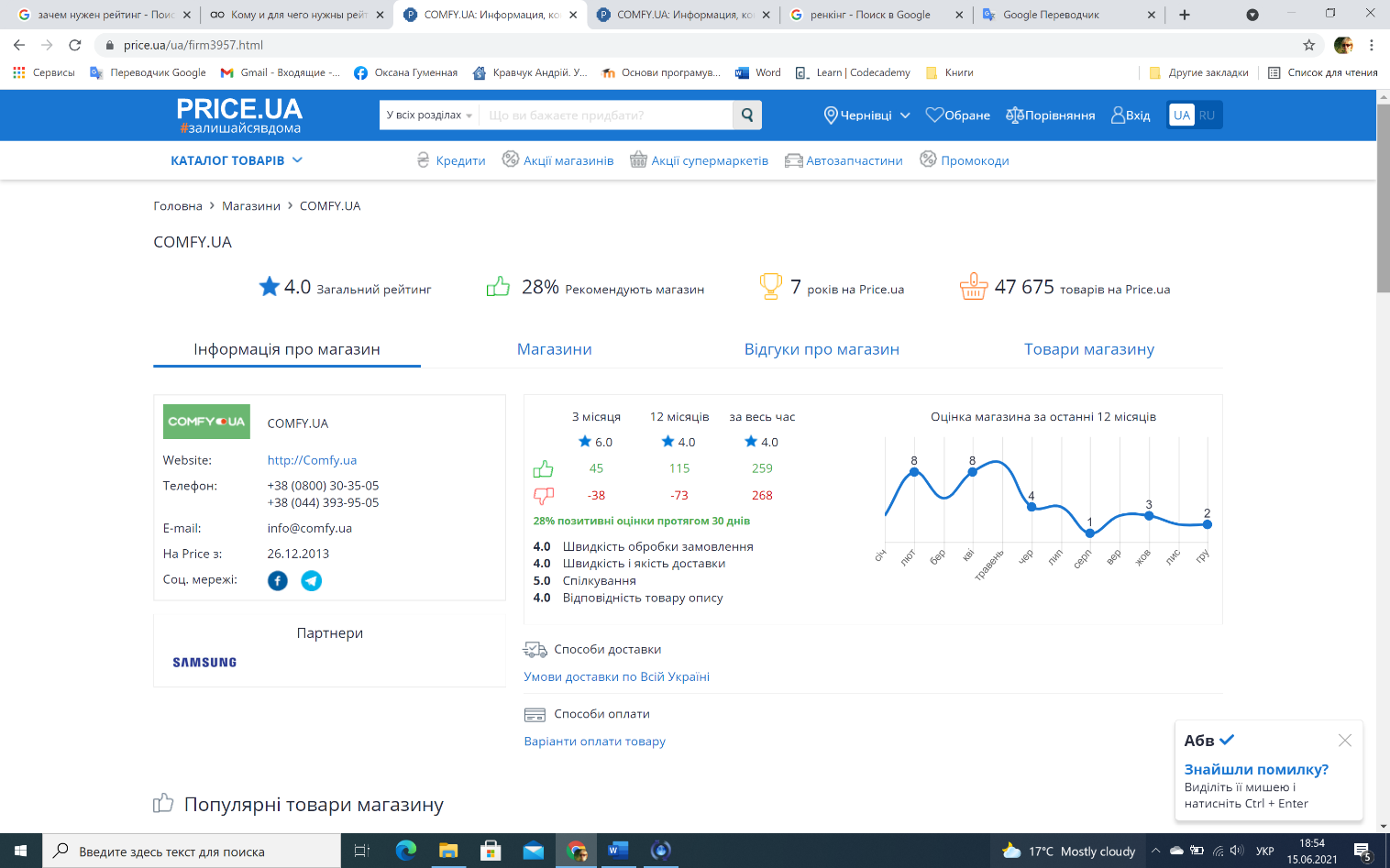


Рисунок 1.5 – Сторінка про магазин Comfy

Наступний вебсайт, 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.

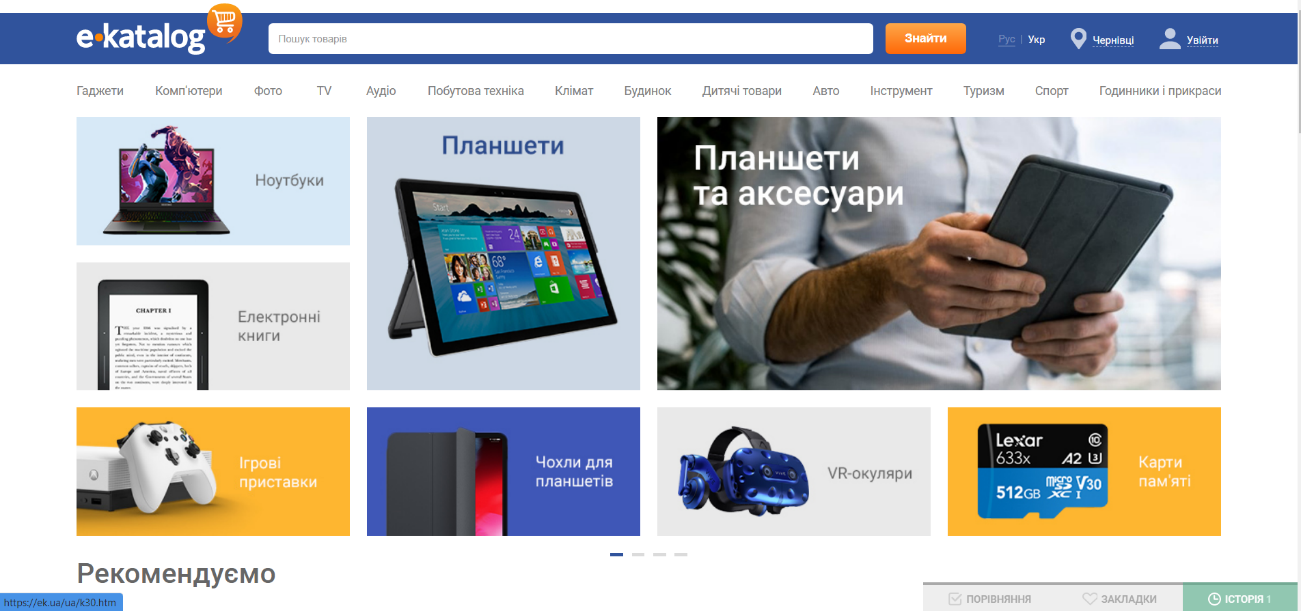


Рисунок 1.6 – Головна сторінка E-katalog.ua

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.

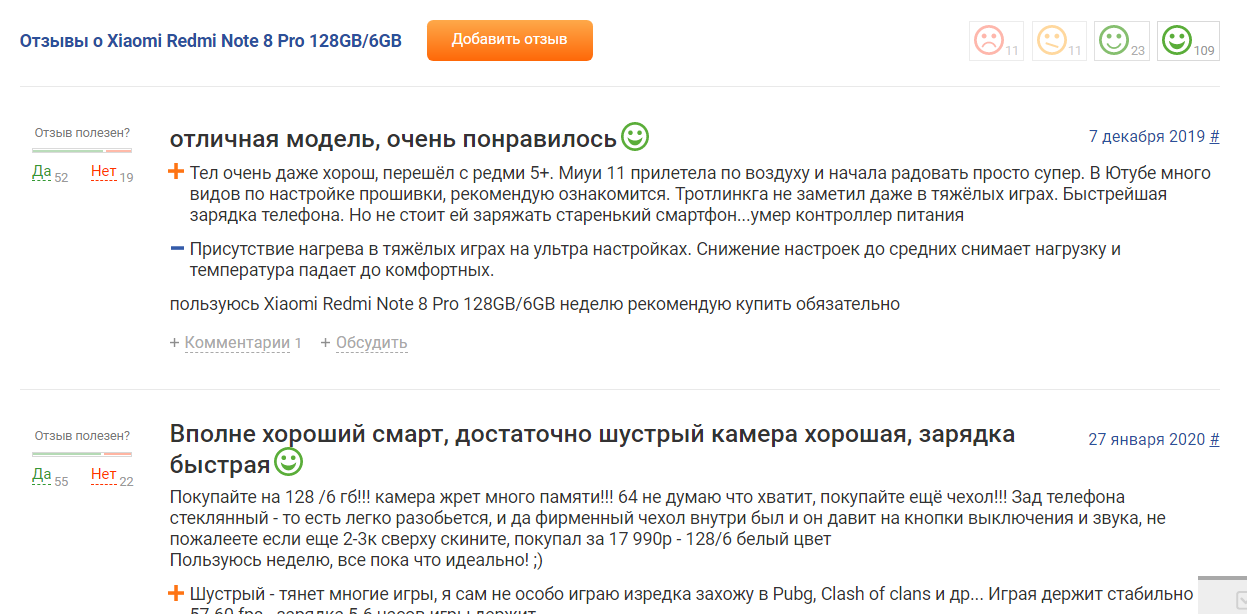


Рисунок 1.8 – Сторінка відгуків E-katalog.ua

На рисунку 1.9 зображено сторінку для створення відгуку.

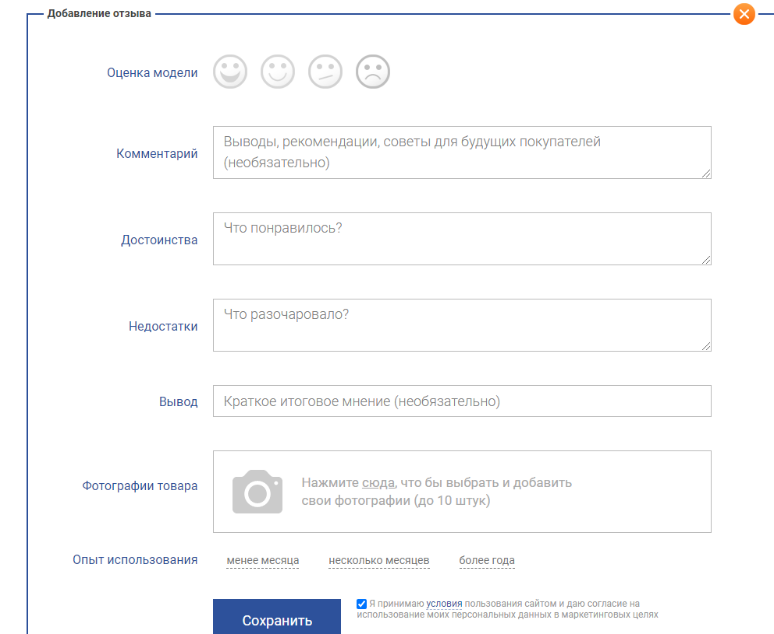


Рисунок 1.9 – Сторінка створення відгука E-katalog.ua

Підсумуємо позитивні сторони вебсайтів і їх недоліки в таблиці 1.1

Таблиця 1.1 – Плюси і мінуси аналогів-вебсайтів

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

Отже, ми розглянули такі вебсайти Price.ua і E-katalog.ua.

## 1.3 Маркетингове дослідження інновацій

Розглянемо маркетингове дослідження вебсайту

Інформаційний сайт що розробляється, є аналогом вже існуючих сайтів каталогів. Потенційним користувачем може бути будь-яка людина, яка цікавиться покупкою товарів, хоче дізнатись інформацію про магазинів. Вебсайт орієнтований на одноосібне використання недосвідченого користувача.

З часом очікується попит на сайт, так як він буде зручним у використанні та мати інтуїтивно-зрозумілий інтерфейс.

Для оцінки конкуренто спроможності додатку потрібно визначити основні параметри базового та нового варіантів програмного забезпечення, виконати конструювання еталону конкурентоспроможності, розрахувати інтегральний показник конкурентоспроможності, визначити ефективність нового програмного забезпечення порівняно з базовим.

Визначення основних параметрів базового і нового варіантів програмного забезпечення.

У процесі дослідження виділяються основні технічні та економічні параметри базового та нового варіантів програмного забезпечення (таблиця 1.2).

Таблиця 1.2 – Основні техніко-економічні параметри додатку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва параметру | Одиниця виміру | Варіант | | Характеристика параметра нового варіанта відносно базового (↑, ↓ чи =) |
| базовий  (аналог) | новий |
| Вартість шаблону | грн | 4650 | 2850 | ↑ |
| Результати дослідження інтерфейсу методом GOMS | с | 4,9 | 4,9 | = |
| Кількість функцій шаблону | шт. | 5 | 5 | = |
| Вага шаблону | Мб | 2,5 | 2,6 | ↓ |
| Кількість інформації, що передається | біт | 3,7 | 3,9 | ↑ |
| Можливість нарощування функціональних характеристик | бали | 3 | 3 | = |
| Кількість людей, необхідних для обслуговування | чол. | 1 | 1 | = |

Виходячи з отриманих результатів, розробка є кращою відносно базового варіанта за чотирма наступними параметрами: вартість шаблону, кількість функцій шаблону, кількість інформації, що передається. Нова розробка отримала такий самий результат в трьох наступних критеріях: результати дослідження інтерфейсу методом GOMS, кількість людей необхідних для обслуговування, можливість нарощування функціональних характеристик.

Для визначення конкурентоспроможності необхідно сконструювати еталон конкурентоспроможності додатку.

Еталон – це міра для відтворення, зберігання і передачі одиниць будь-якої величини, що утворена за таким правилом: серед показників-стимуляторів (здійснюється позитивний вплив на конкурентоспроможність) відбираємо дані з максимальним значенням, а серед показників дестимуляторів (чинить негативний плив на конкурентоспроможність) відбираємо дані з мінімальним значенням. Визначення еталонного значення наведено у таблиці1.3 (колонки 2, 3).

Інтегральний показник конкурентоспроможності відносно еталона () визначається за формулою (1.1):

, (1.1)

де ,  – величини за *і*-тим параметром відповідно базового, нового варіанта ПЗ та еталона.

Результати розрахунків заносимо до таблиці 1.3 (колонки 4, 5).

Проведені розрахунки будуть свідчити про те, що базове та нове ПЗ може переважати еталон за параметрами. Результатом буде одна із наступних умов:

* якщо , то базове програмного забезпечення перевищує новий зразок за конкурентоспроможністю;
* якщо , то поступається йому;
* при , нове програмного забезпечення знаходиться на одному рівні з базовим.

Таким чином, базовий додаток переважає за еталон параметрами 6 та 8, а новий – 7 та 4. Тобто новий мобільний додаток є безперечно конкуренто-спроможним порівняно з базовим.

Загальна ефективність розробки нового програмного забезпечення порівняно з базовим визначається за формулою (1.2):

, (1.2)

Якщо серед параметрів є такі, для яких мінімальне значення є найбільш ефективним, то має місце рівність (1.3):

, (1.3)

де ,  – величини за *і*-тим параметром відповідно нового та базового варіанту ПЗ.

Проте, якщо серед параметрів є такі, для яких максимальне значення є найбільш ефективним, то має місце рівність (1.4):

, (1.4)

Для розрахунку загальної ефективності визначимо відносне значення параметрів нового програмного забезпечення відносно базового (див. табл. 1.3, колонка 6).

Розрахунок ефективності окремого параметру нового програмного забезпечення здійснюється за формулами (1.3) та (1.4), результати розрахунків заносимо до таблиці 1.3 (колонки 7, 8).

Таблиця 1.3 – Параметри конкурентоспроможності та показники ефективності

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва параметра | Еталонне значення | | Варіант | |  |  |  |
|  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Вартість додатку | min | 2850 | 1.66 | 1 | 0,6 | 0,4 |  |
| Результати дослідження інтерфейсу методом GOMS | min | 4,9 | 1 | 1 | 1 | 0 |  |
| Кількість функцій додатку | max | 5 | 0,9 | 1 | 1,11 |  | 0,11 |
| Вага додатку | min | 2,5 | 0.85 | 1 | 1 | 0,15 |  |
| Кількість інформації, що передається | max | 3,9 | 0,95 | 1 | 1,05 |  | 0,05 |
| Можливість нарощування функціональних характеристик | max | 3 | 1 | 0,5 | 1 |  | 0 |
| Кількість людей, необхідних для обслуговування | min | 1 | 1 | 1 | 1 | 0 |  |
| Разом: |  |  | 7,21 | 7,02 |  | 0,71 | |

Оскільки загальна вартість шаблону, що розробляється, є нижчою за вартість аналогу, то виграш від цього буде складати: 1 – 0,6 = 0,4. При розробці шаблону не вдалося покращити інтерфейс відносно базового, дані показники є однаковими. При запланованій кількість функцій шаблону виграш буде складати: 1,11 – 1 =0,11. Вага шаблону, що розробляється, є меншою за вагу аналога, тому отримуємо виграш у розмірі: 1– 0,85 = 0,15. Передбачена кількість інформації, що передається, дозволить отримати виграш: 1,05 – 1 = 0,05, запланована можливість нарощення функціональних характеристик є рівною для обох шаблонів.

Відповідно до цього, загальна економічна ефективність складе:

*Ефзаг* = 0,4 + 0,11 + 0,15 + 0,05 = 0,71.

Таким чином, ефективність виробництва нового вебпродукту відносно аналогу складає 0,71.

При виконанні маркетингового дослідження інновації розробки шаблону, виконавши усі необхідні розрахунки, встановлено, що технологія нової розробки відповідає оптимальному рівню витрат, і, у підсумку, розроблений шаблон та його заповнення є економічно доцільним та конкурентоспроможним для вільного розповсюдження на ринку.

Отже, ми розглянули і обчислили суму ефективності нової розробки вебсайту.

В даному розділі представлено аналіз предметної області, проаналізовано сайти аналогічного призначення, виконано постановку задачі, визначено основні функціональні вимоги до розробки та проведено маркетингове дослідження інноваційної розробки.

# РОЗДІЛ 2. Технічний проєкт

## 2.1 Технічне завдання

Розглянемо технічне завдання дипломного проєкту.

Розробити інформаційний сайт про магазини міста з відгуками користувачів про них, а також з рейтингом магазинів по місту.

Вхідна специфікація:

* інформація про магазин (логотип, вебсайт, пошта, категорія, телефон);
* відгуки про магазин;
* потрапляння чи ні магазину у список улюблених;
* оцінка роботи магазину.

Вихідна специфікація:

* 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.

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 |
| --- | --- | --- |
| Передпроєктне дослідження | 1.05-10.05  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 |
| Моделювання проєкту | 12.05-18.05  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 | 19.05-21.05  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 | 26.05-18.06  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.

Отже, ми створили технічну документацію для поставленої задачі та розглянули етапи розробки вебсайту.

## 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.

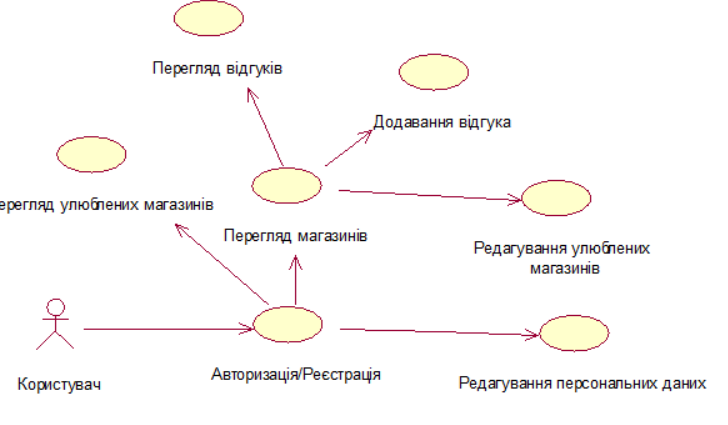
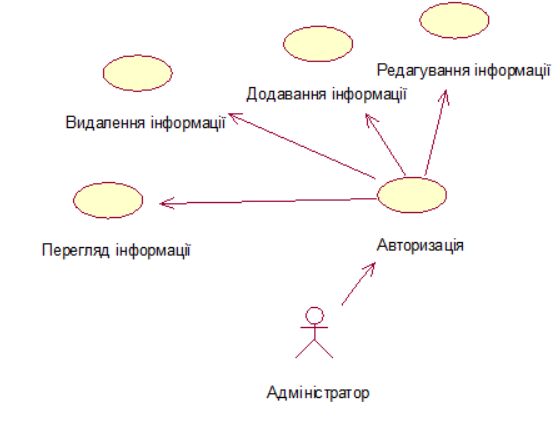


Рисунок 2.1 – Модель прецедентів

На ній 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.

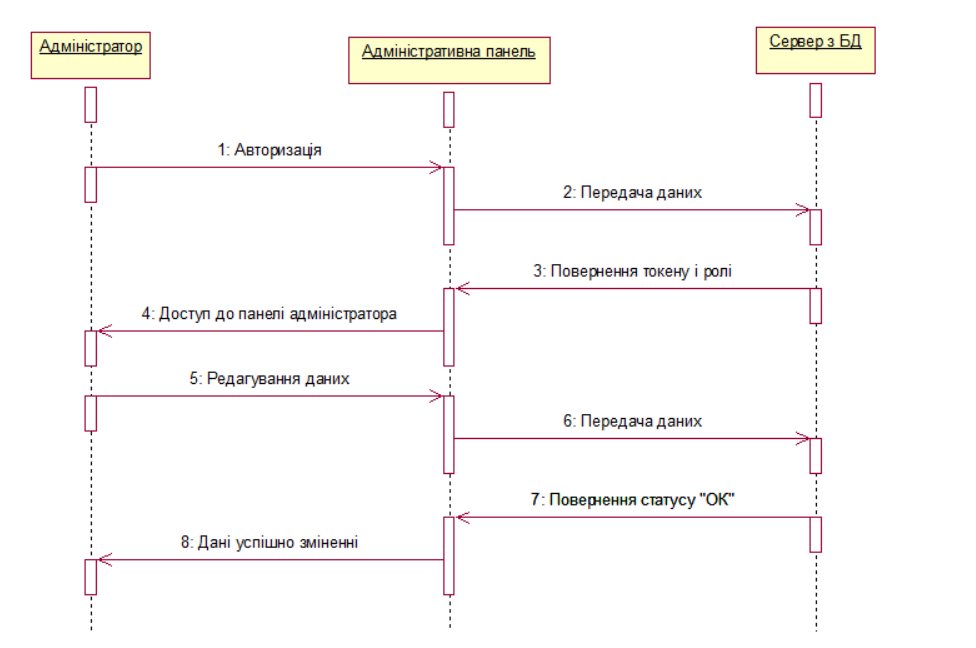


Рисунок 2.4 – Діаграма взаємодії для редагування даних

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.

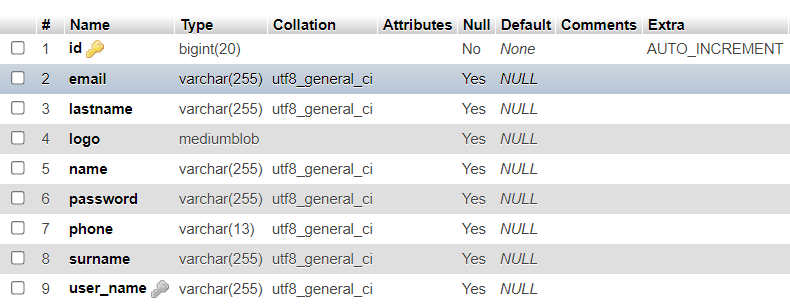


Рисунок 2.5 – Таблиця «User\_Entity»

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.

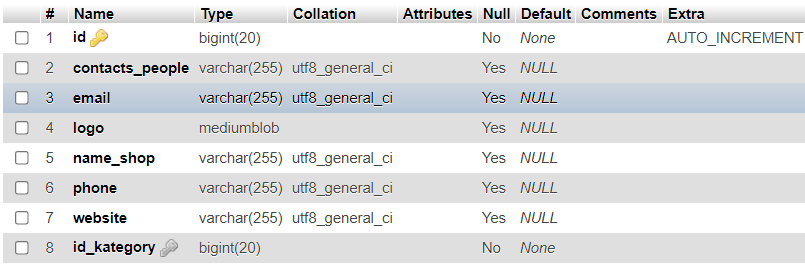


Рисунок 2.6 – Структура таблиці «Info\_Shop»

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.

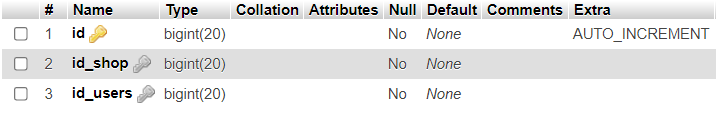


Рисунок 2.7 – Структура таблиці «Favourite»

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.

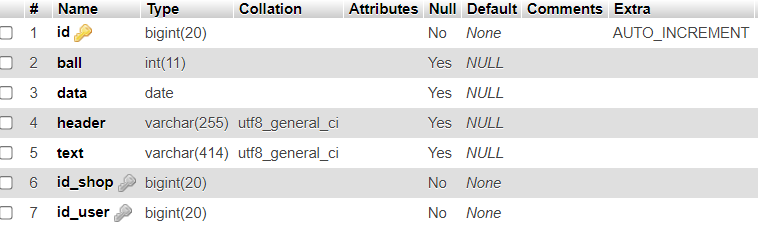


Рисунок 2.8 – Структура таблиці «Review\_Entity»

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.

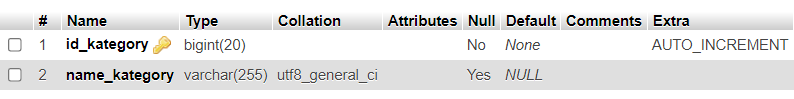


Рисунок 2.9 – Структура таблиці «Kategory»

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.

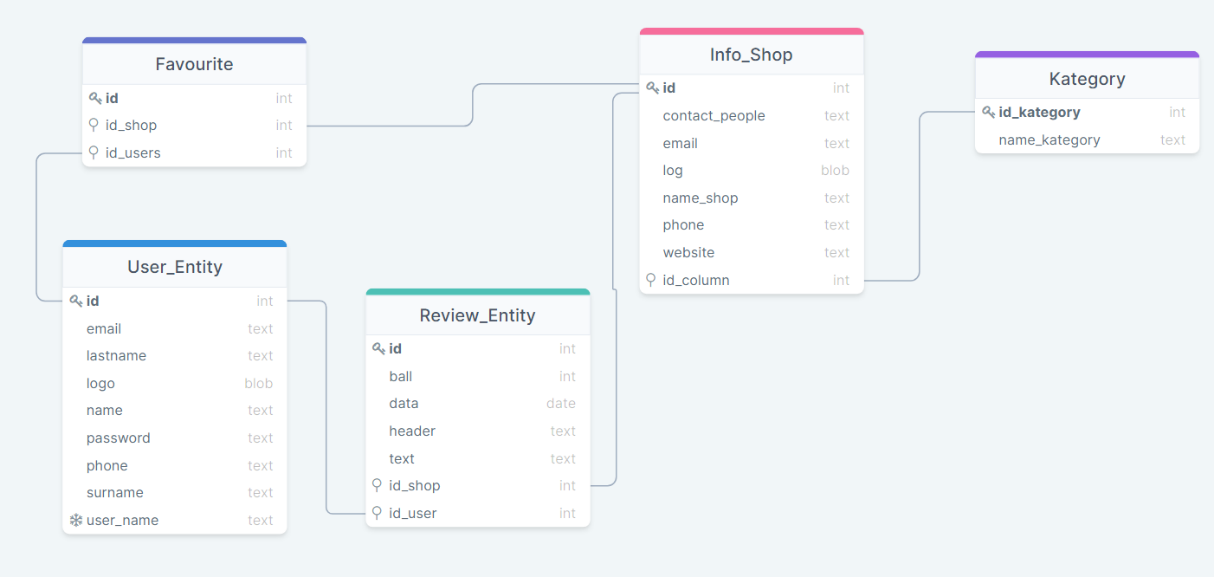


Рисунок 2.10 – Логічна модель бази даних

Отже, розглянули логічну і фізичну модель таблиць бази даних.

## 2.4 Проєктування інтерфейсу

Розглянемо проєктування інтерфейсу вебсайту.

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

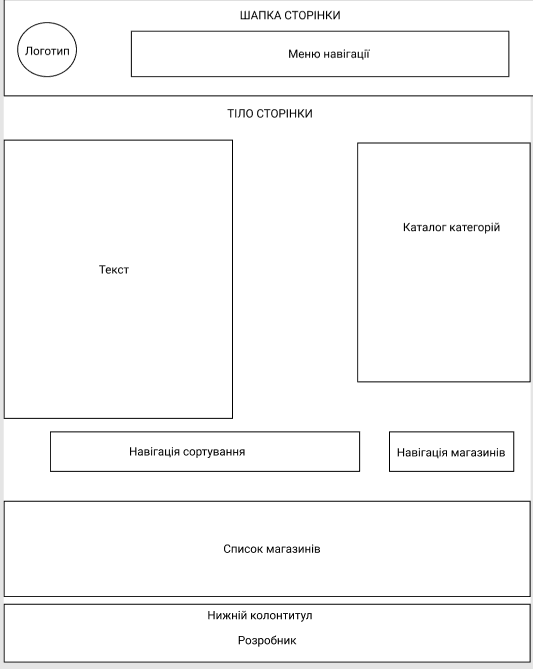
.

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.

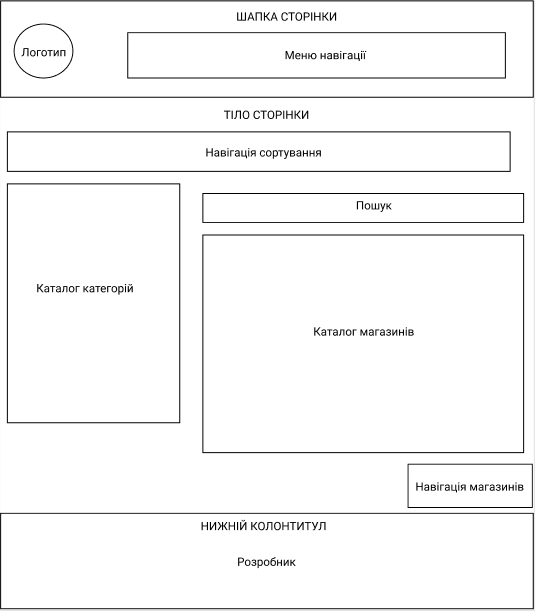


Рисунок 2.12 – Структура сторінки «Каталог»

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

Структура сторінки «Улюблені» зображена на рисунку 2.13.

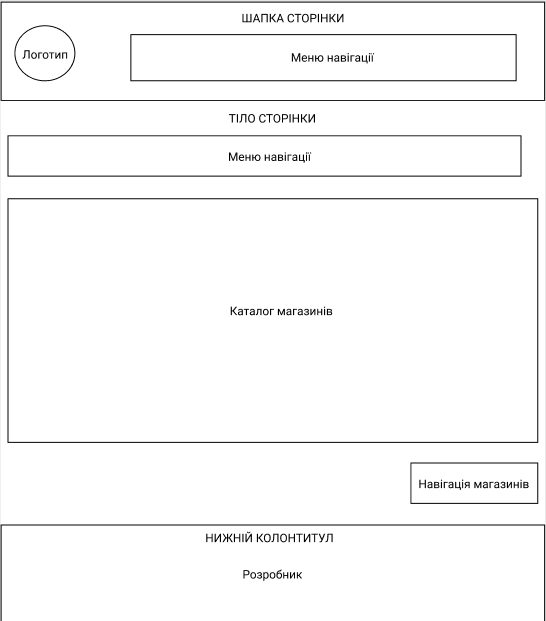


Рисунок 2.13 – Структура сторінки «Улюблені»

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.

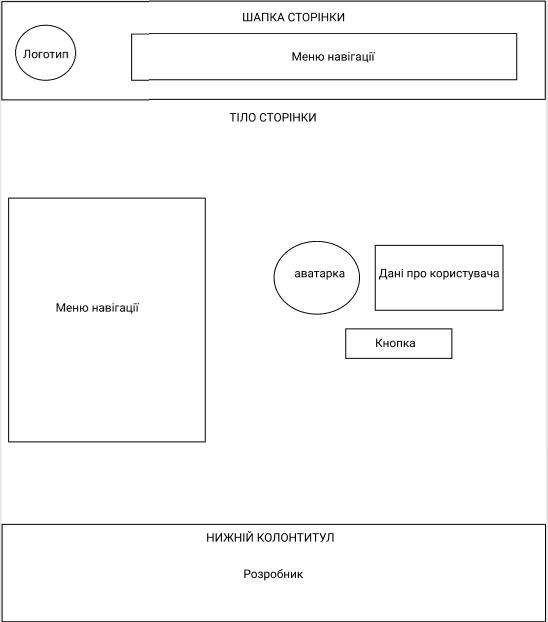


Рисунок 2.14 – Структура сторінки «Профіль»

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.

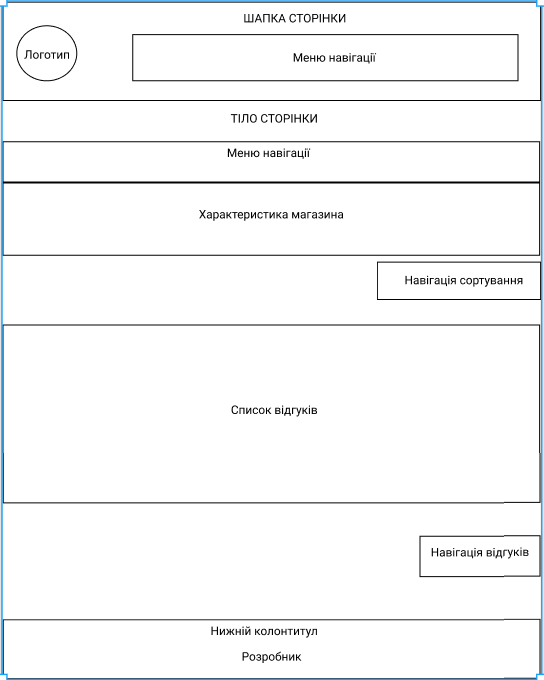


Рисунок 2.15 – Структура сторінки «Характеристика магазину»

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.

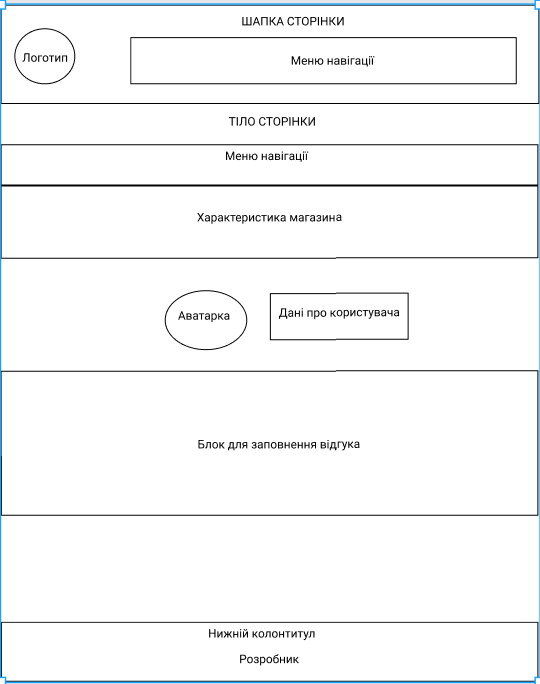


Рисунок 2.16 – Структура сторінки «Написання відгуку»

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

Отже, ми розглянули структуру інтерфейсу вебсайту.

# РОЗДІЛ 3. робочий проєкт

## 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.

Фрейморк 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 пружин.

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 С 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 наслідником JavaScript.

Для створення програмного коду обрано такі середовища розробки як Visual Studio Code і 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 відладчик, 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 І 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 ІР 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 ІР 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.

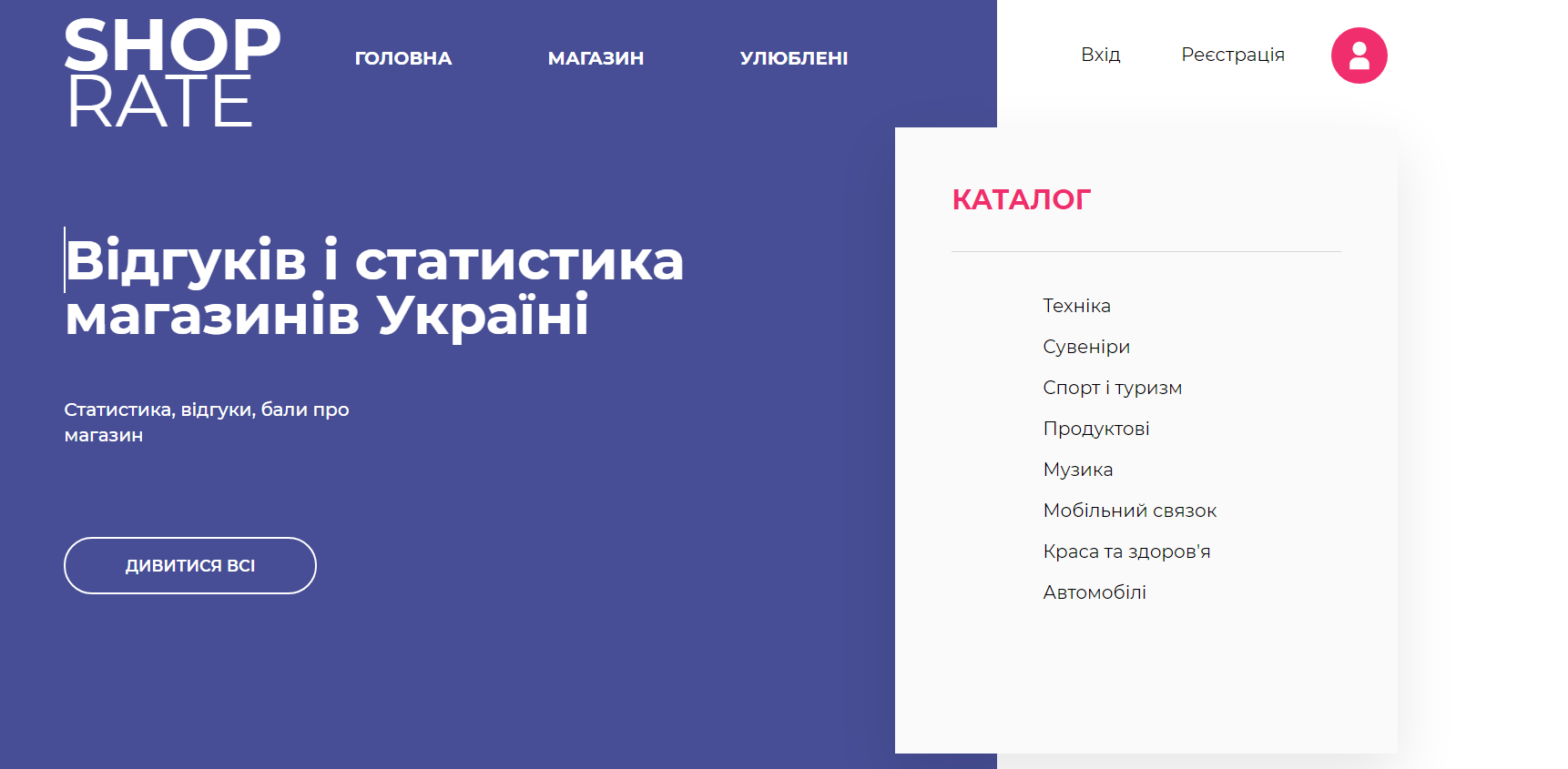


Рисунок 3.1 – Головна сторінка

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.

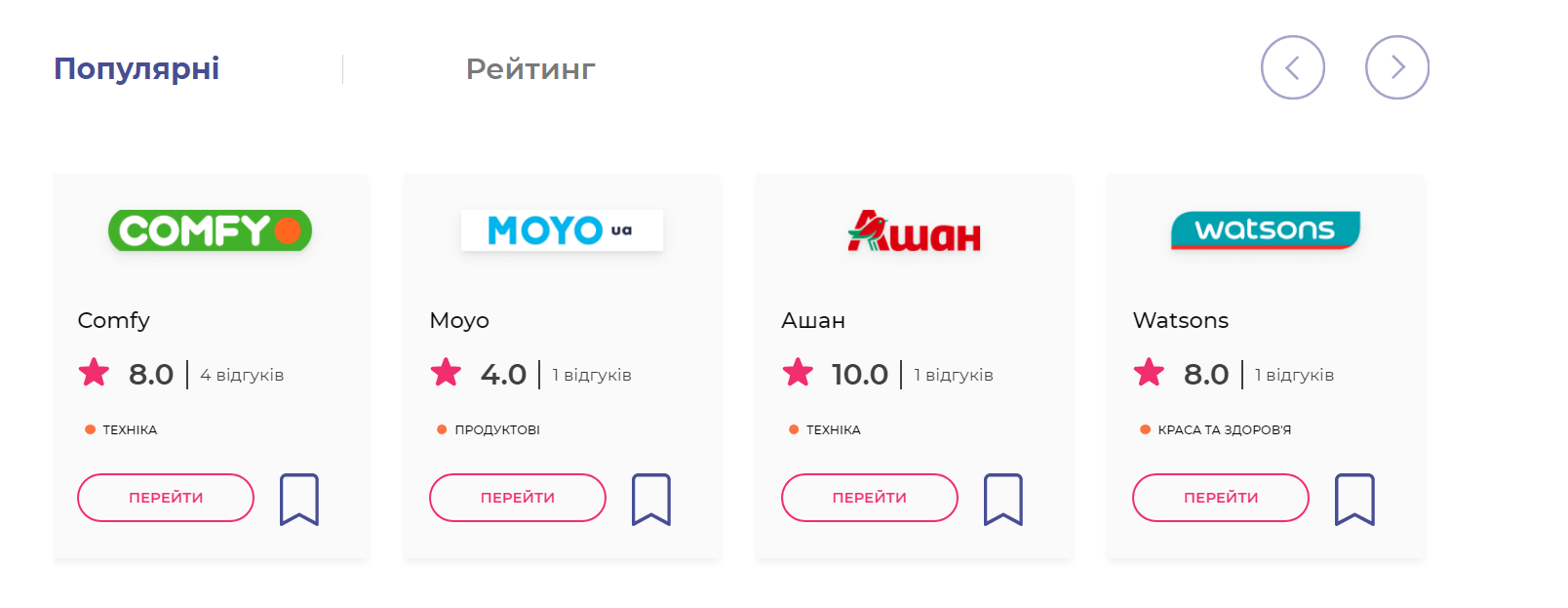


Рисунок 3.2 – Популярні магазини.

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

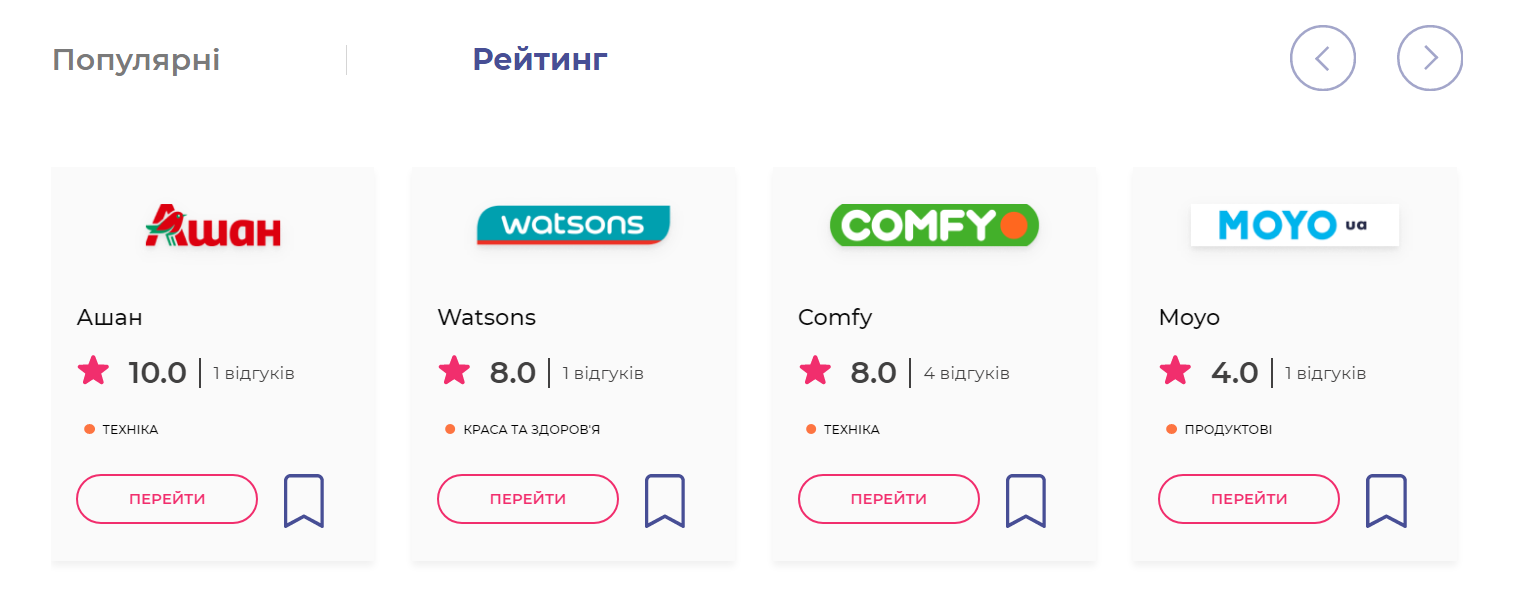


Рисунок 3.2 – Рейтинг магазинів

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.

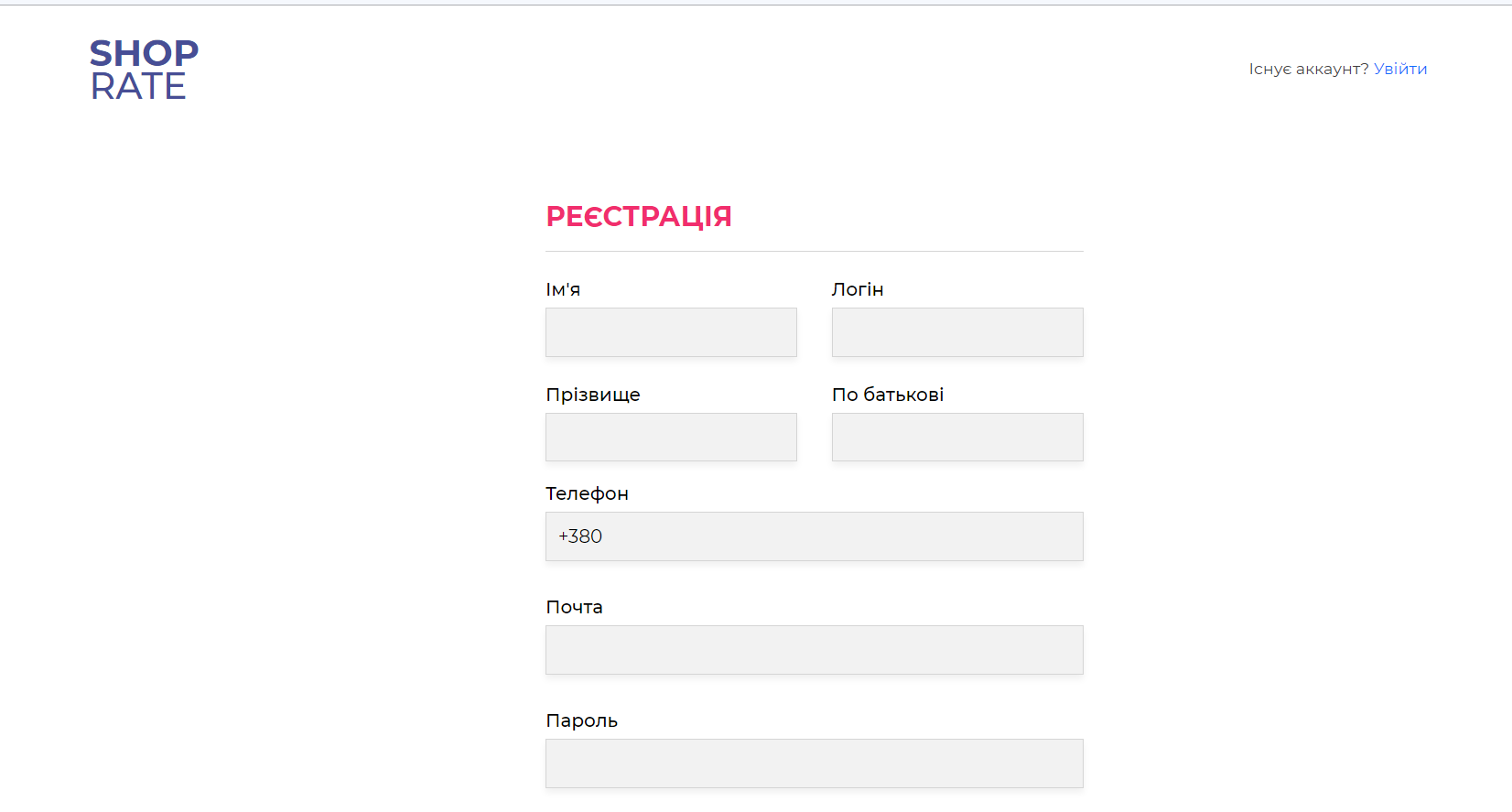


Рисунок 3.4 – Реєстрація користувача

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

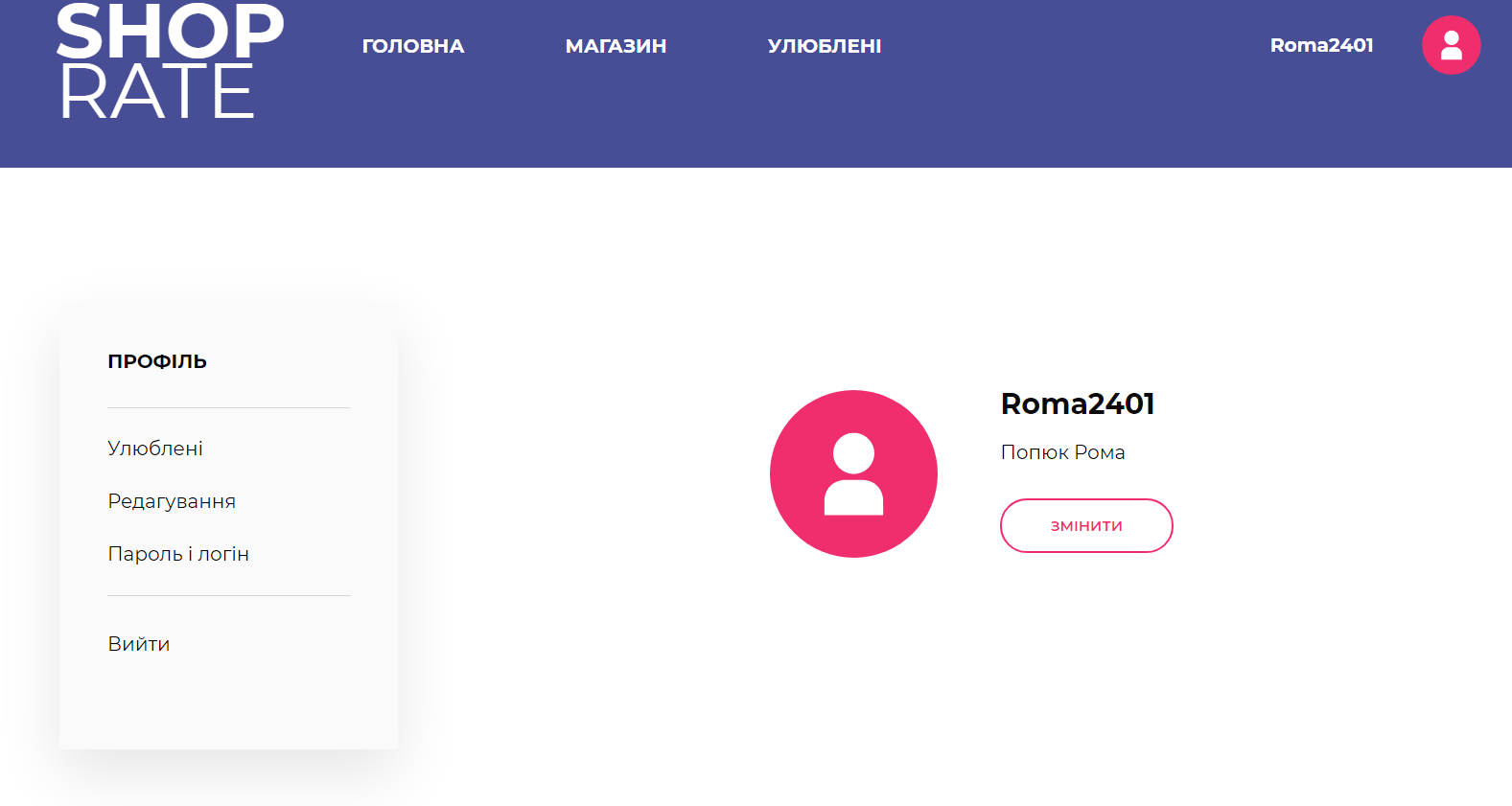


Рисунок 3.5 – Успішна авторизація

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.

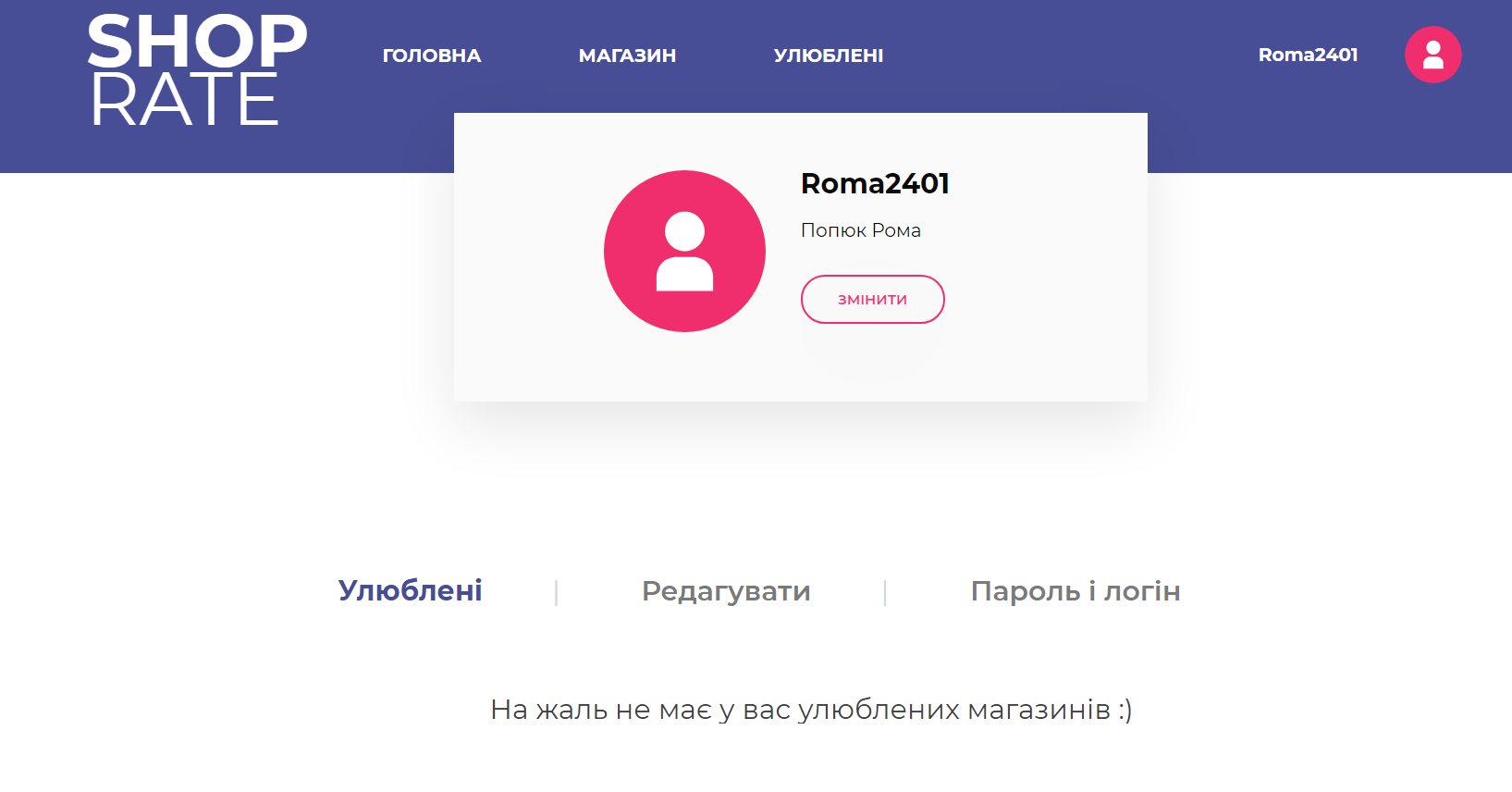
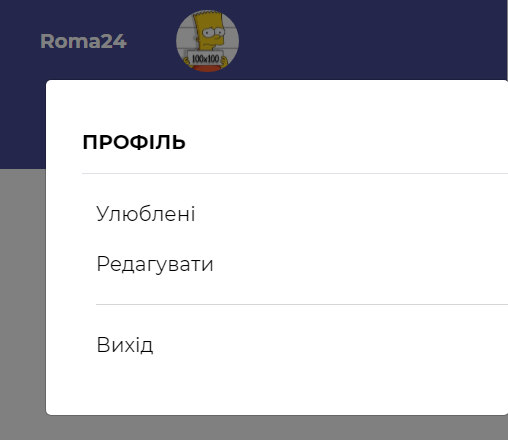
 

Рисунок 3.6 – Сторінка редагування профілю

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.

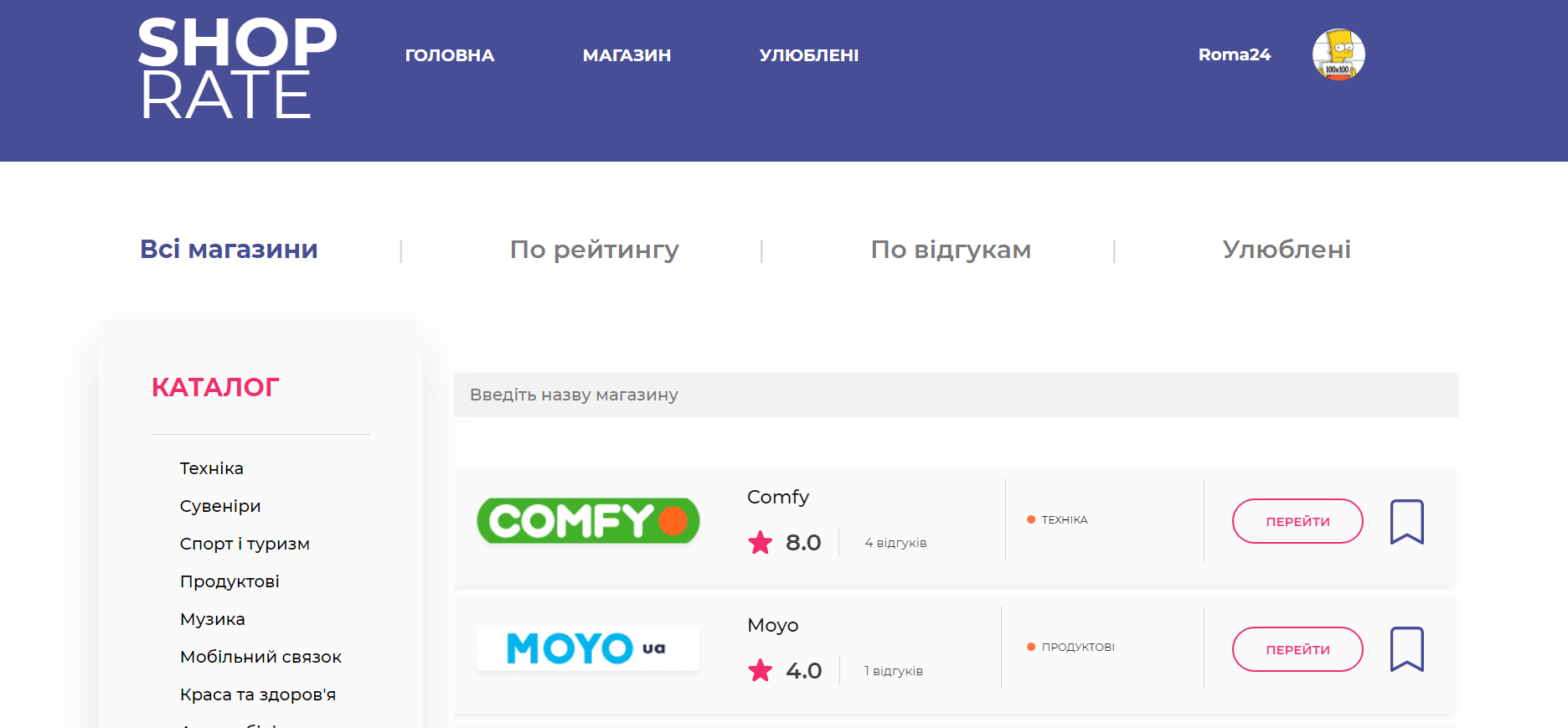


Рисунок 3.7 – Каталог магазинів для користувача

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 (рисунок3.8).

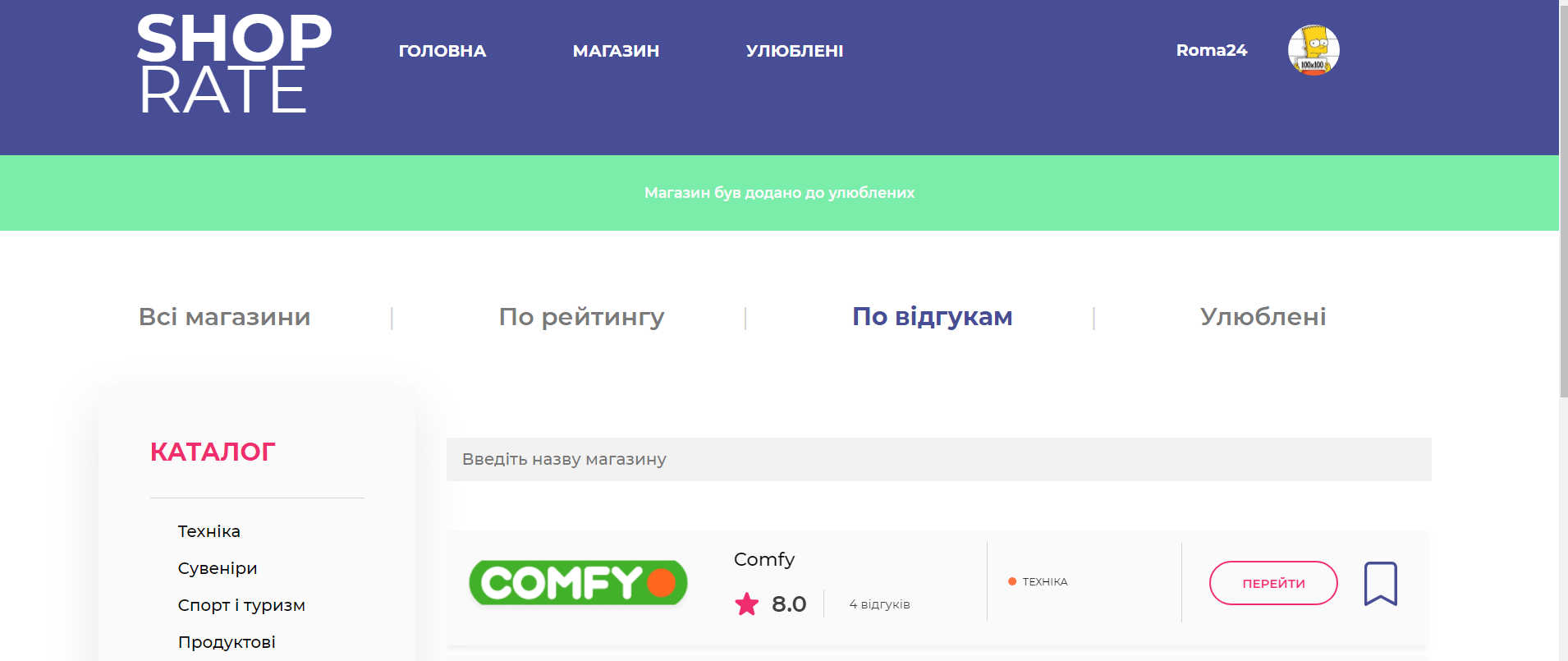


Рисунок 3.8 – Додавання до улюблених

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

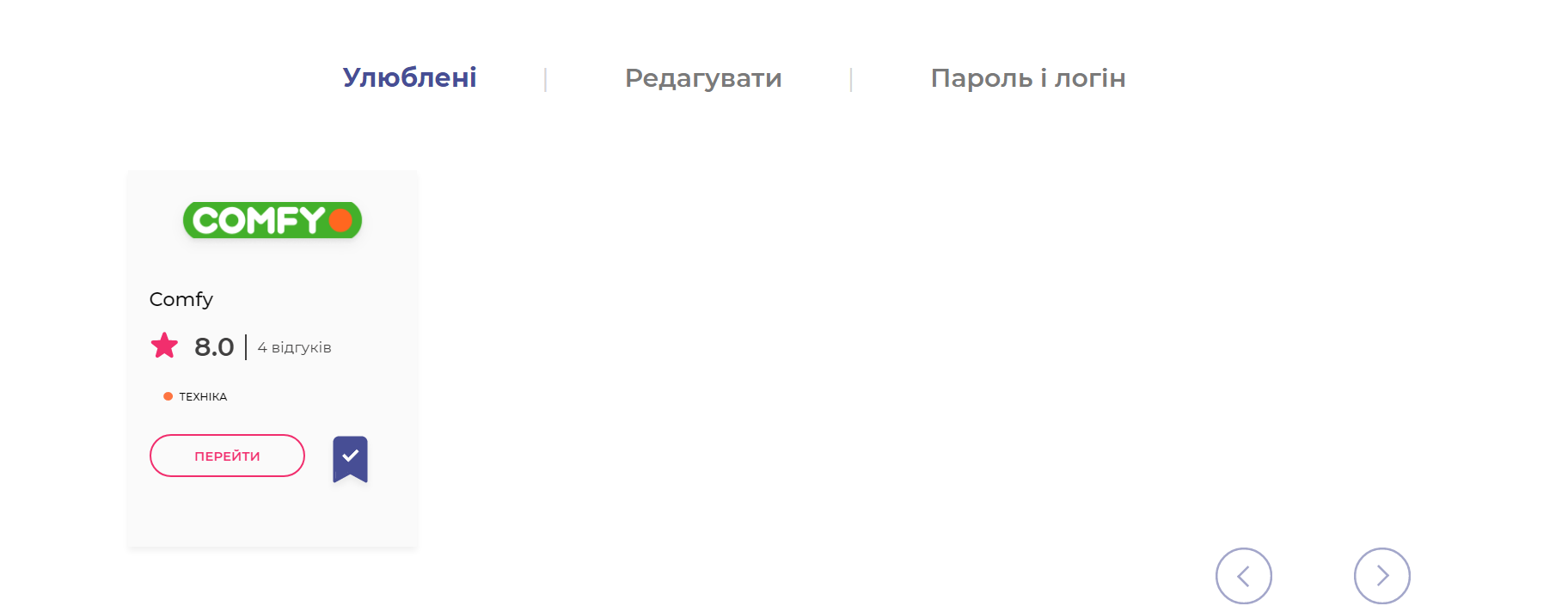


Рисунок 3.9 – Улюблені магазини

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

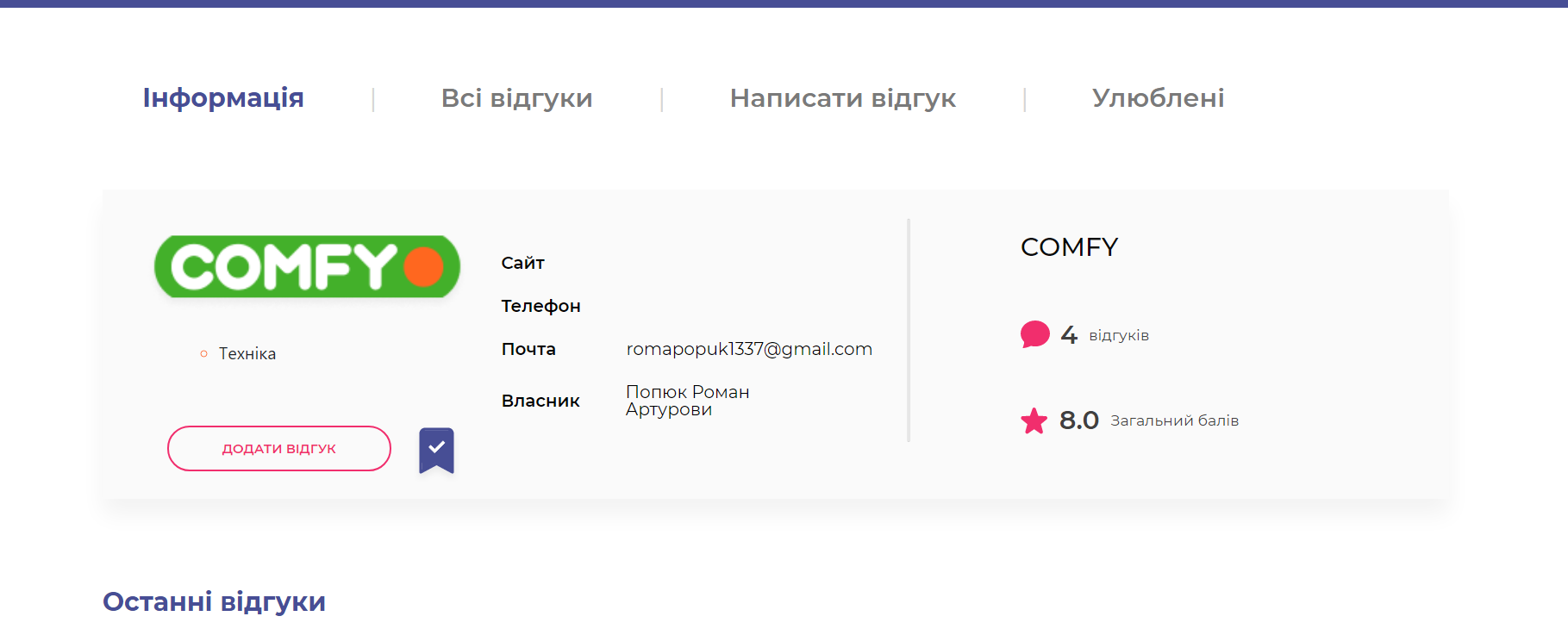


Рисунок 3.10 – Повна інформація про магазин

Сторінка для відображення відгуків зображена на рисунку 3.11.

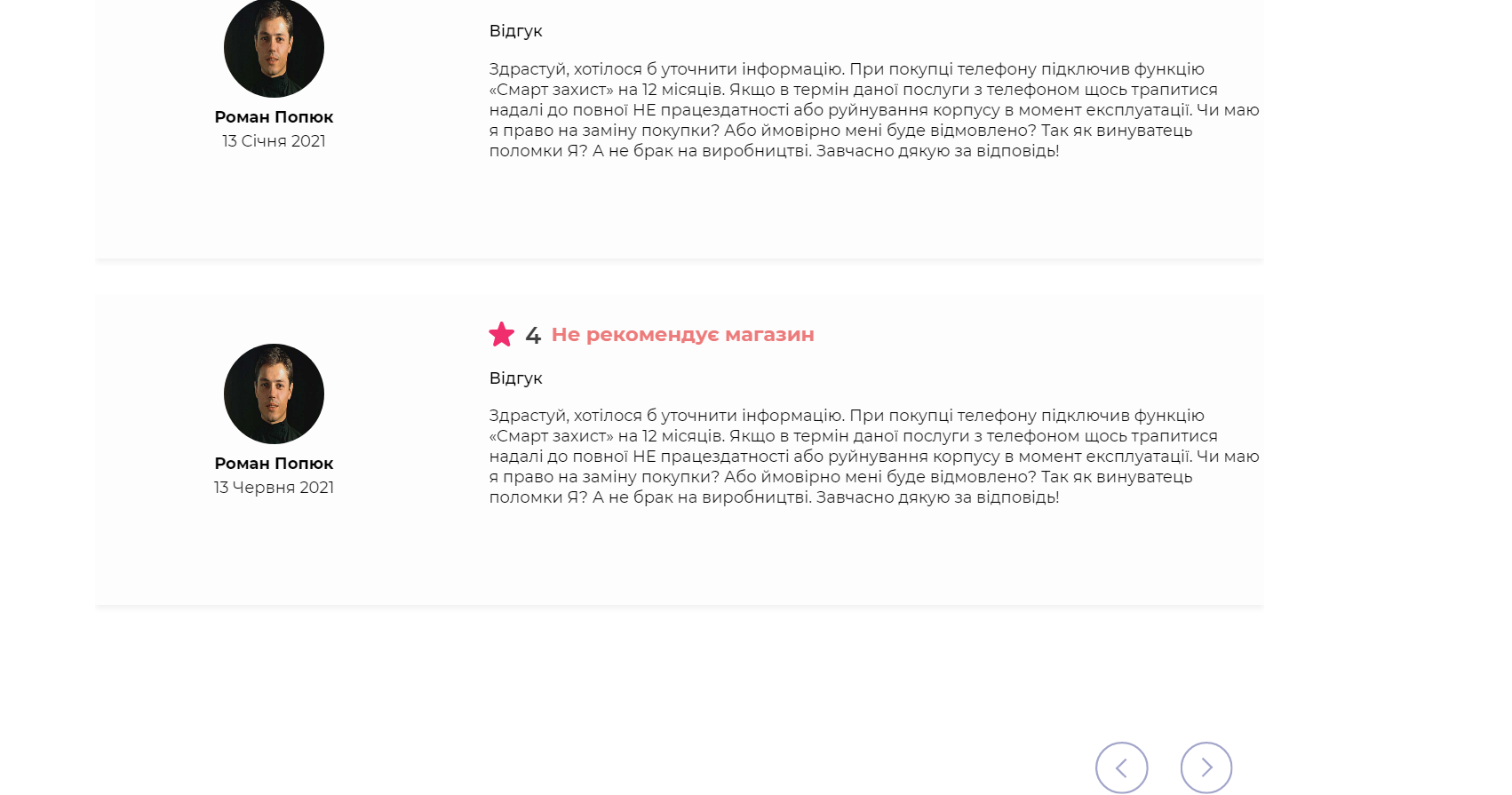


Рисунок 3.11 – Відгуки про магазин

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

Отже, ми розгляну інструкцію користувачеві.

## 3.3 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.

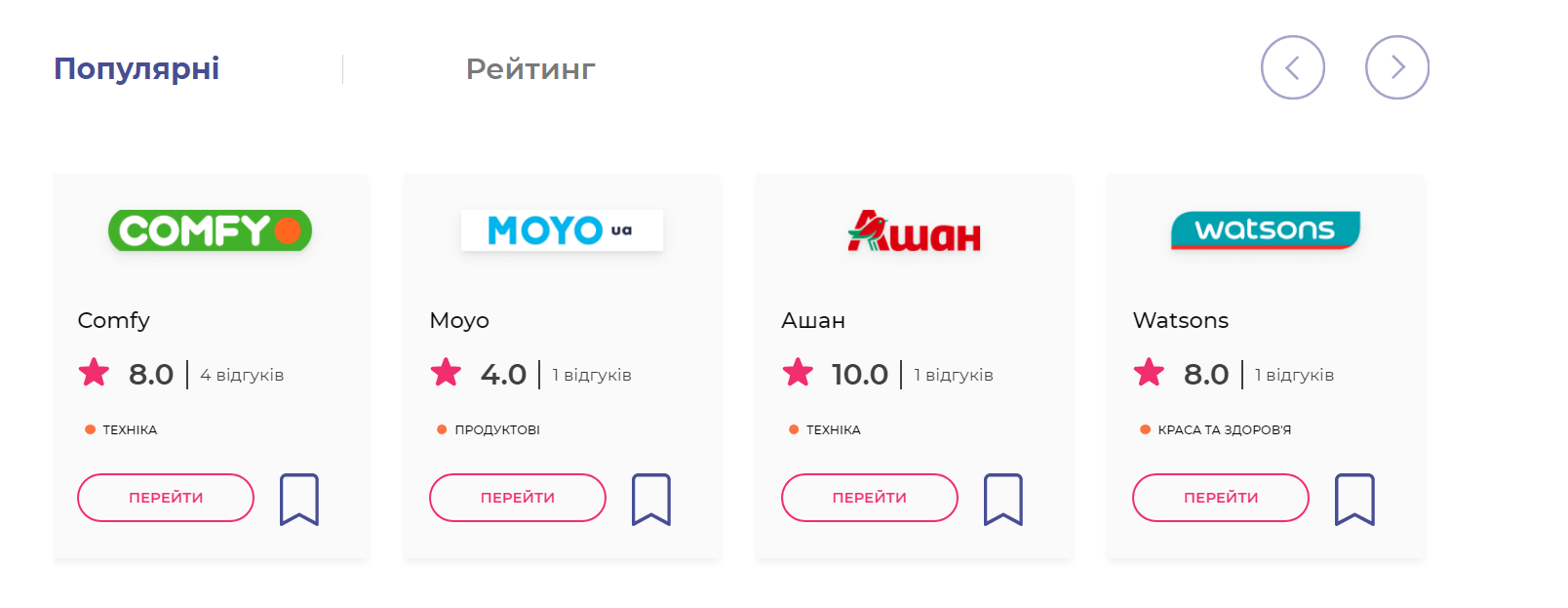


Рисунок 3.16 – Список популярних магазинів

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

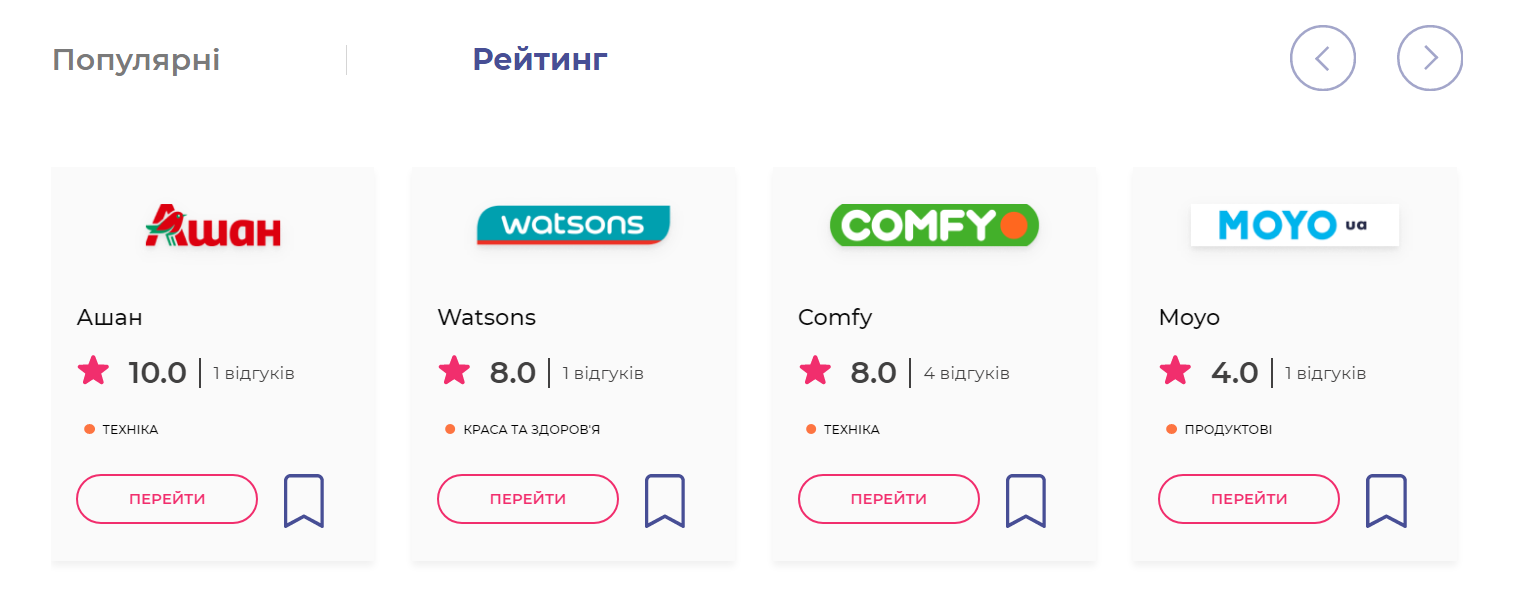


Рисунок 3.17 – Рейтинг магазинів

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.

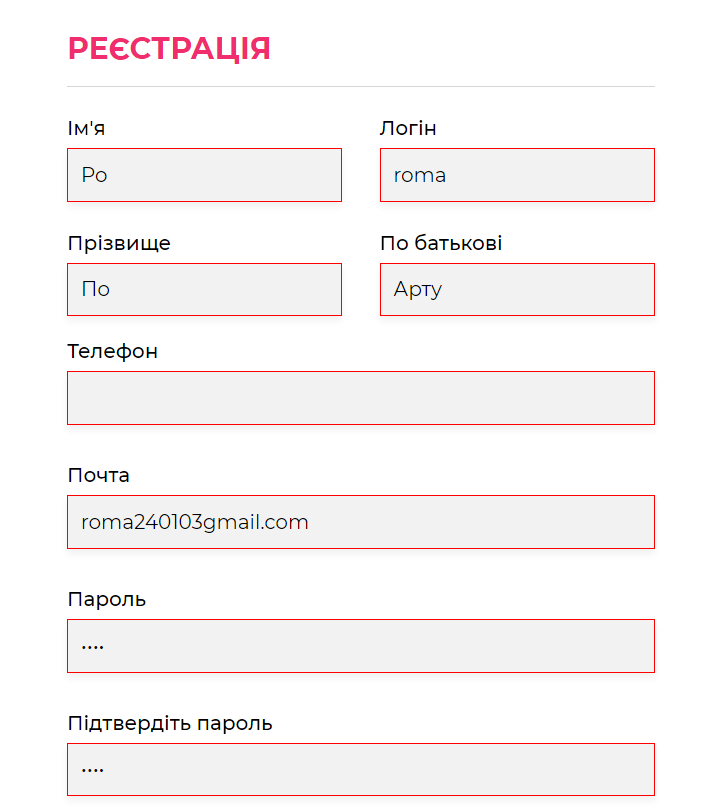


Рисунок 3.18 – Введення некоректних даних

Введені коректно дані зображено на рисунку 3.19.

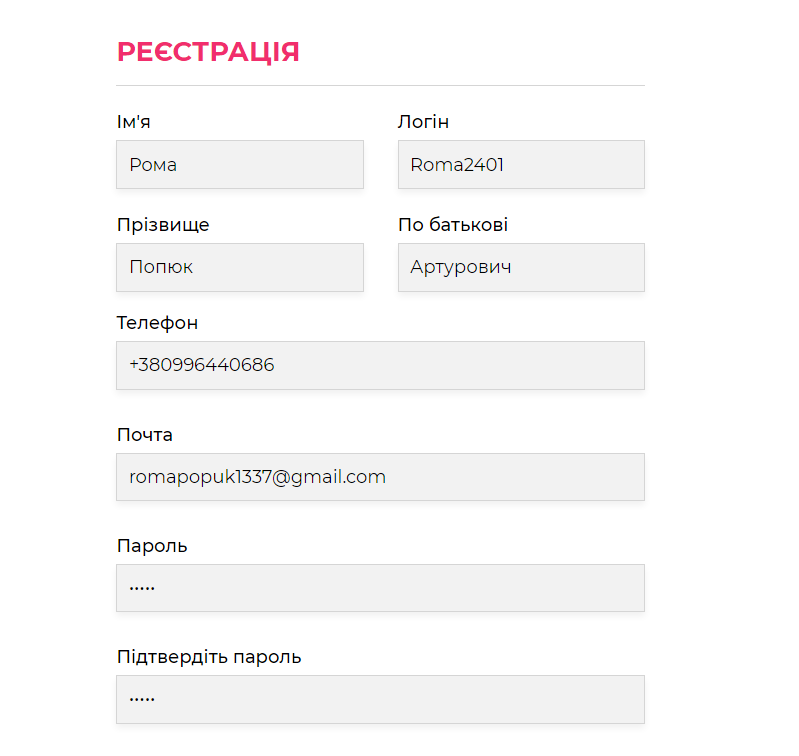


Рисунок 3.19 – Введення коректних даних

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

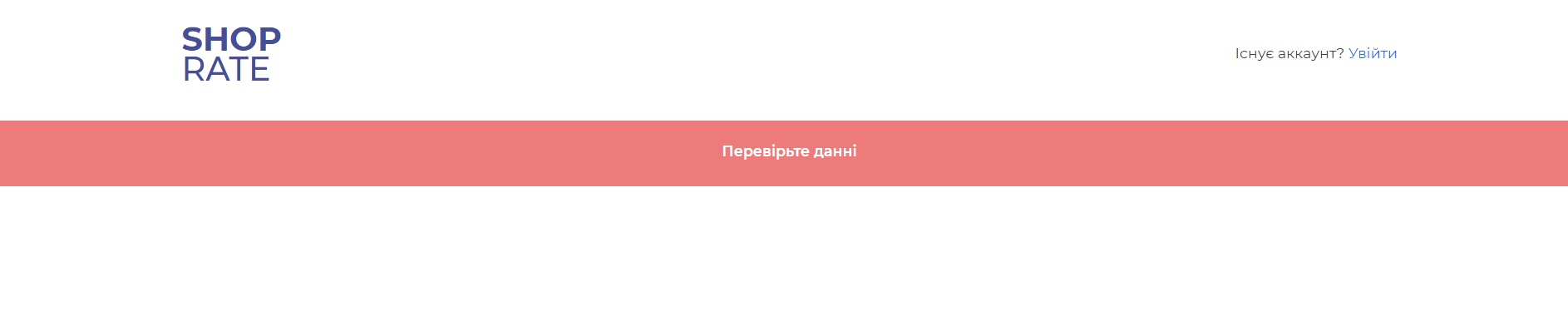


Рисунок 3.20 – Помилка реєстрації

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.

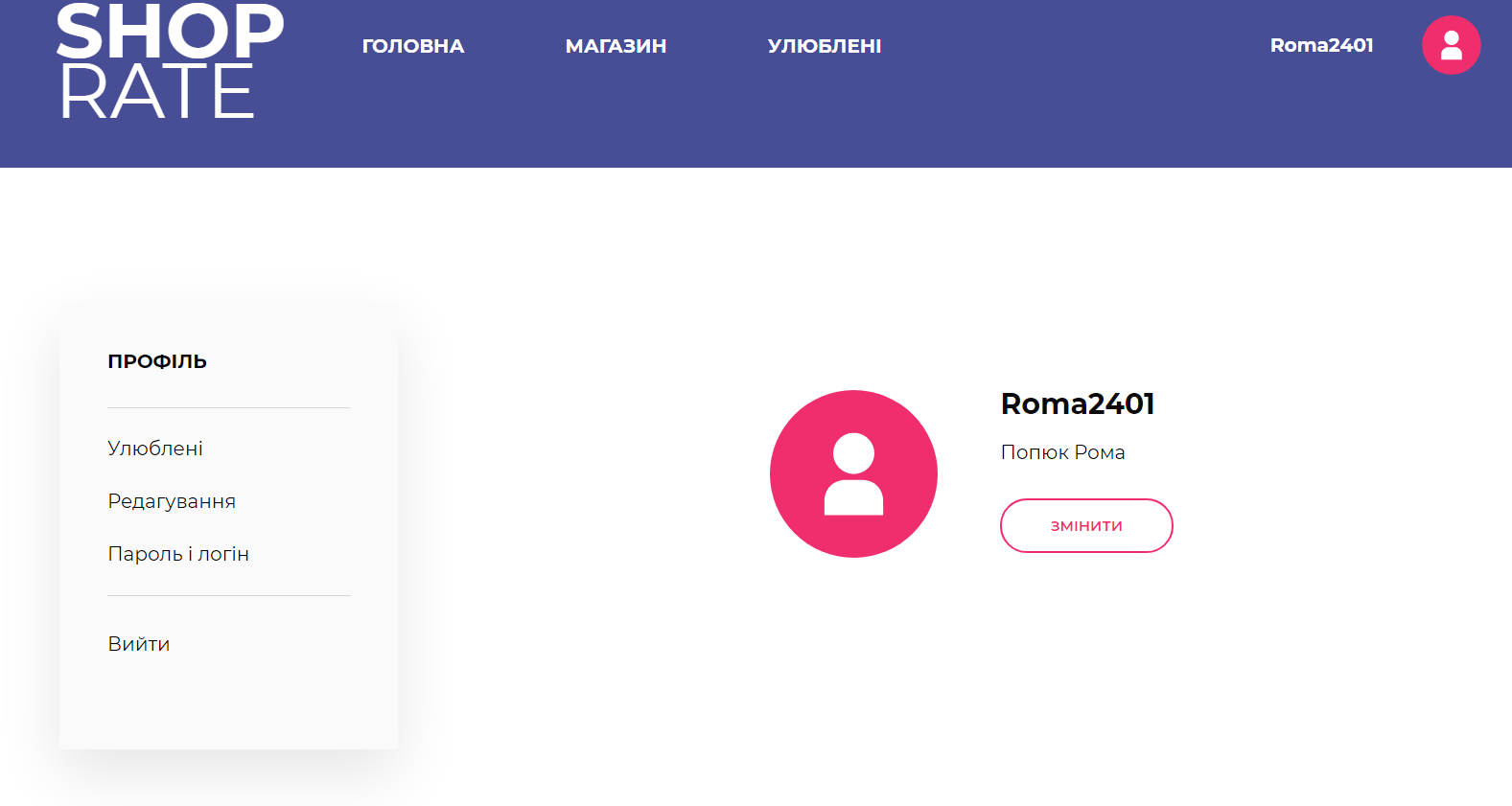


Рисунок 3.21 – Успішна авторизація, перехід на профіль користувача

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

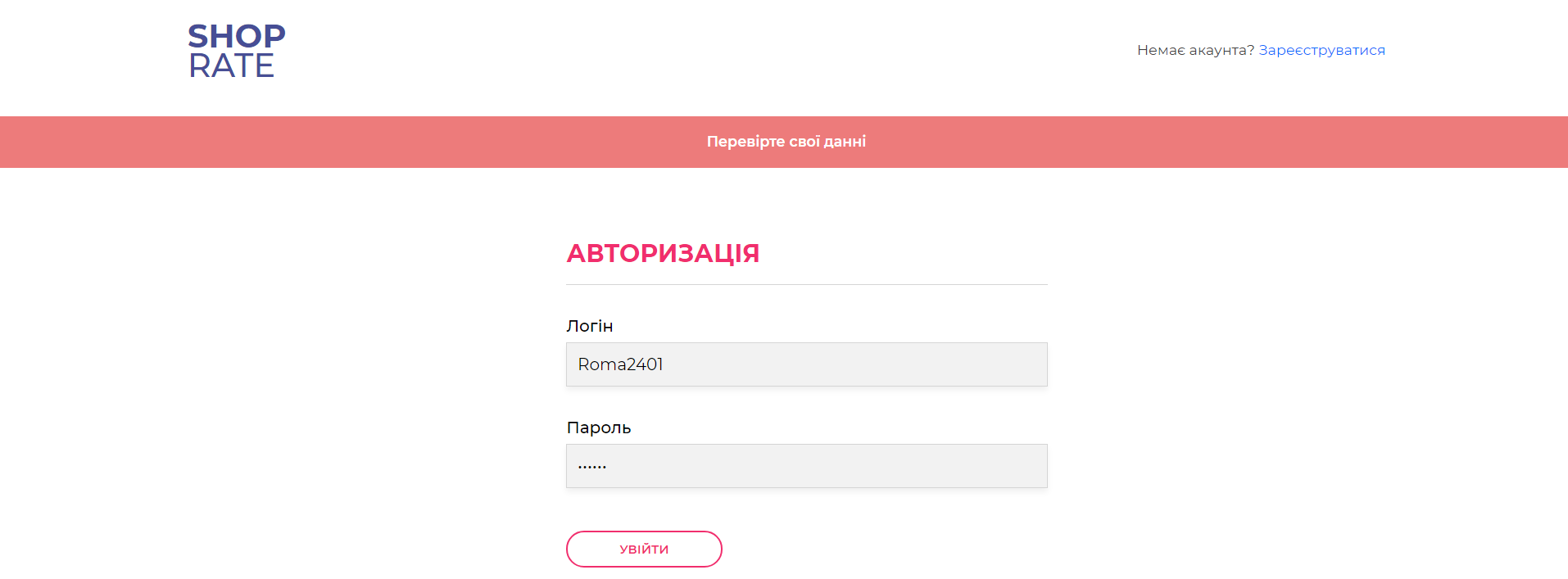


Рисунок 3.22 – Помилка при авторизації

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).

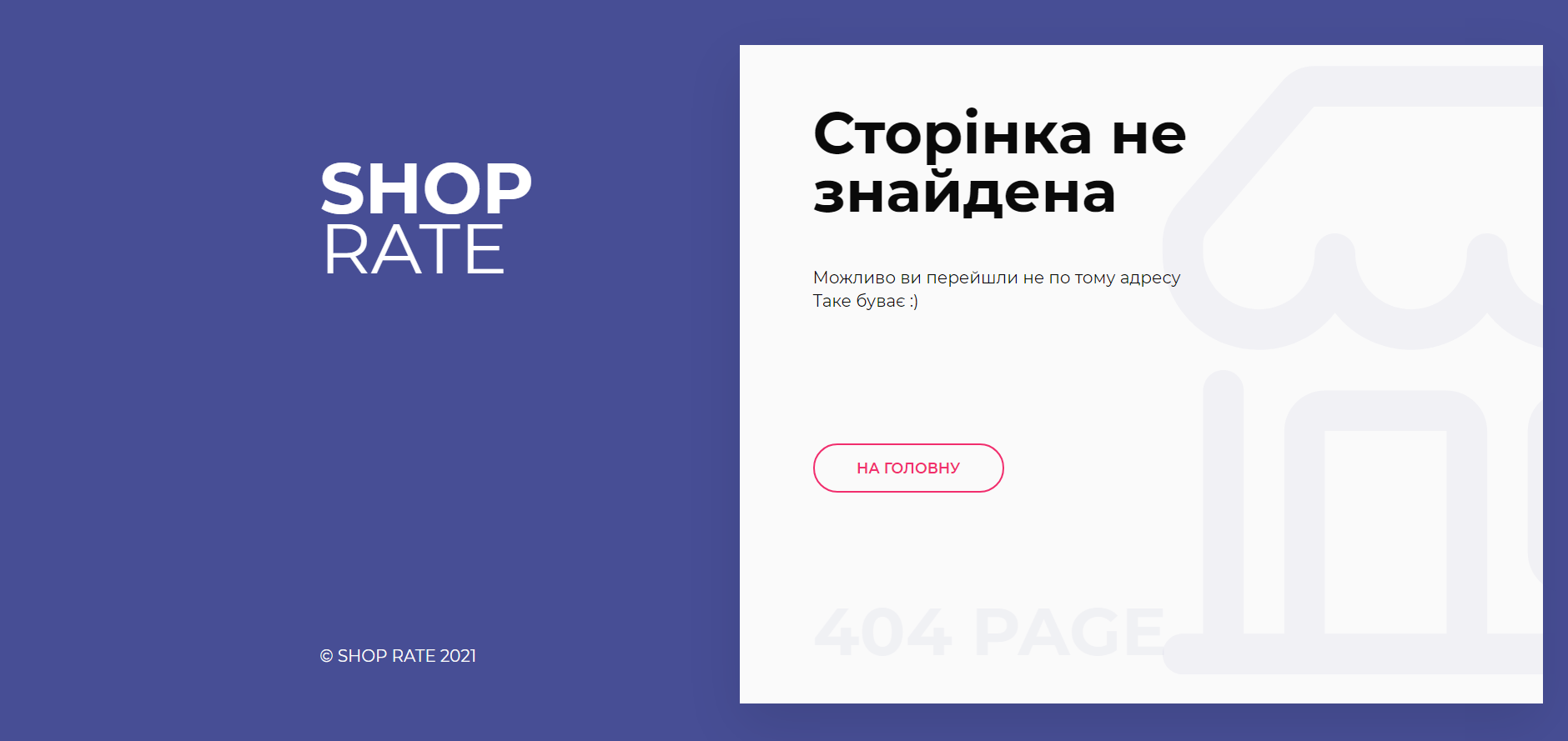


Рисунок 3.23 – Помилка 404

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

## 3.4 Охорона праці

Виконання вимог інструкції є обов’язковим для осіб, які працюють з програмним забезпеченням. Інструкції можуть бути видані на руки, розміщені на робочих місцях, чи зберігатися у визначеному місці, доступному для працюючих.

Напруга живлення 220 В є небезпечною для життя людини. Тому, незважаючи на те, що в конструкції комп’ютера передбачена достатня ізоляція від струмопровідних ділянок, необхідно знати та чітко виконувати ряд правил техніки безпеки.

Забороняється:

* торкатися екрана і тильного боку дисплея, проводів живлення та заземлення, з’єднувальних кабелів;
* порушувати порядок увімкнення й вимикання апаратних блоків;
* класти на апаратуру сторонні предмети;
* працювати за комп’ютером у вологому одязі та вологими руками;
* палити в приміщенні, де знаходяться комп’ютери.

Під час роботи за комп’ютером необхідно:

* суворо дотримуватися інструкції з експлуатації апаратури;
* працювати на клавіатурі чистими сухими руками, не натискати на клавіші без потреби чи навмання;
* коректно завершувати роботу з тим чи іншим програмним засобом.

У разі появи запаху горілого, самовільного вимикання апаратури, незвичних звуків треба негайно вимкнути комп’ютер від’єднати від системи електропостачання та повідомити про можливе виникнення небезпечної ситуації обслуговуючий персонал. Не можна працювати за комп’ютером при недостатньому освітленні, високому рівні шуму тощо.

Організація робочого місця оператора повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування вимогам НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації обчислювальних машин».

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана, зручність ручного керування пристроєм введення-виведення інформації у зоні досяжності моторного поля.

Конструкція робочого місця має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів.

Екран має розташовуватися на оптимальній відстані від очей користувача, що становить 600..700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Розташування екрана має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30° до нормальної лінії погляду працюючого.

Клавіатуру слід розташовувати на поверхні столу на відстані 100..300 мм від краю, зверненого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений з матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5..15°. Висота середнього рядка клавіш не має перевищувати 30 мм. Поверхня клавіатури має бути матовою з коефіцієнтом відбиття 0,4.

Розташування пристроїв введення-виведення інформації має забезпечувати їх добру видимість та зручність ручного керування.

# ВИСНОВКИ

На мою думку, інтернет на сьогоднішній день – найпопулярніше джерело інформації. Саме створення вебресурсів займає чільне місце серед розробок програмного забезпечення. Вдало організований інформаційний сайт – це великий плюс як для користувачів (пошукачів інформації), так і для власників сайтів (компаній, фірм).

В результаті даного дипломного проекту було створено вебресурс «Shop Rate», який 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.

Під час роботи над дипломним проектом спершу було здійснено аналіз предметної області та огляд відповідних аналогів. Також було проведено маркетингове дослідження даної розробки. Наступним етапом було реалізовано моделювання програмного забезпечення та моделювання даних, проектування інтерфейсу.

У результаті виконаної роботи були поглиблено знання мови програмування JavaScript, Typescript та Java та принципи правильного їх використання при створені веб-додатків. Вдосконалено навички при роботі з HTML та CSS. Cтворено макет веб-сайту у додатку для веб-дизайнерів «Figma». Самостійно розглянуто сервіс GitHub, на його платформі створений репозиторій з дипломною роботою.

Розроблена система пройшла тестування та може бути використана на просторах інтернету.

В майбутньому для даного вебресурсу можна добавити новини для магазинів, акції, які в них проводяться. Для користувачів можна додати сповіщення про новини і акції улюблених магазинів.

# перелік джерел

1. Дудзяний І.М. Об’єктно-орієнтоване моделювання програмних систем: Навч. посібник. – Видавничий центр ЛНУ ім. І. Франка, 2007. – 108с.
2. Бегун А.В., Камінський О.Є. Web-програмування. Навч. посіб. – КНЕУ, 2011. – 436с.
3. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. – «Магнолія», 2011. – 456с.
4. Глинський Я.М., Ряжська В.А. Інтернет. Сервіси, HTML і web-дизайн. Навчальний посібник. – «Деол», 2003. – 184с.
5. Грайворонський М.В., Новіков О.М. Безпека інформаційно-комунікаційних систем. – Видавнича група ВНУ, 2009. – 608с.
6. Форум з зразками коду [Eлектронний ресурс]. – Режим доступу: <https://www.stackoverflow.com/>
7. Гончаров С.М., Кушнір Н.Б. Практикум з маркетингу. – «Центр учбової літератури», 2012. – 208с.

# ДОДАТОК

Додаток A

**Лістинг програмного коду**

**А.1. Spring Boot(BackEnd)**

**А.1.1 UserService**

@Service

public class UserService {

@Autowired

UserRepository userRepository;

@Autowired

AdminService adminService;

@Autowired

InfoShopRepository infoShopRepository;

@Autowired

KategoryRepository kategoryRepository;

@Autowired

FavouriteRepository favouriteRepository;

@Autowired

ReviewRepository reviewRepository;

public void saveImageShop(MultipartFile file, Long id) {

InfoShop infoShop = infoShopRepository.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();

}

infoShopRepository.save(infoShop);

}

public void saveImage(MultipartFile file, String username) {

UserEntity userEntity = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

String filename = StringUtils.cleanPath(file.getOriginalFilename());

if (filename.contains("..")) {

new Exception("Error");

}

try {

userEntity.setLogo(Base64.getEncoder().encodeToString(file.getBytes()));

} catch (IOException e) {

e.printStackTrace();

}

userRepository.save(userEntity);

}

public void setInfo(String username, UserRequestUpdateData userRequestUpdateData) {

UserEntity userEntity = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

userEntity.setName(userRequestUpdateData.getName());

userEntity.setLastname(userRequestUpdateData.getLastname());

userEntity.setSurname(userRequestUpdateData.getSurname());

userEntity.setLogo(userRequestUpdateData.getLogo());

userEntity.setPhone(userRequestUpdateData.getPhone());

userRepository.save(userEntity);

}

public void setInfoLoginAndPassword(String username, UserRequestLoginAndPassword userRequestLoginAndPassword) {

UserEntity userEntity = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

userEntity.setUserName(userRequestLoginAndPassword.getLogin());

userEntity.setPassword(userRequestLoginAndPassword.getPassword());

userRepository.save(userEntity);

}

public void deleteImg(String username) {

UserEntity userEntity = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

userEntity.setLogo(null);

userRepository.save(userEntity);

}

public UserRequestUpdateData getUser(String username) {

UserEntity user = userRepository.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) {

int sum = 0, count = 0;

UserEntity user = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

Favourite[] favourites = favouriteRepository.findAllByUserId(user).toArray(new Favourite[0]);

InfoShop[] infoShops = new InfoShop[favourites.length];

for (int i = 0; i < favourites.length; i++) {

infoShops[i] = favourites[i].getId\_shop();

}

InfoShopRequest[] infoShopRequests = new InfoShopRequest[favourites.length];

for (int i = 0; i < infoShops.length; i++) {

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShops[i]).toArray(new ReviewEntity[0]);

infoShopRequests[i] = infoShopRequest(infoShops[i]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

count++;

sum = sum + entity.getBall();

}

infoShopRequests[i].setBall(sum/count);

infoShopRequests[i].setResponse(count);

infoShopRequests[i].setSubscribe(true);

infoShopRequests[i].setSubscribeID(favourites[i].getId());

count=0;

sum=0;

}else{

infoShopRequests[i].setBall(0);

infoShopRequests[i].setResponse(0);

infoShopRequests[i].setSubscribe(true);

infoShopRequests[i].setSubscribeID(favourites[i].getId());

}

}

return Arrays.asList(infoShopRequests.clone());

}

public void deleteFavourite(Long id) {

favouriteRepository.delete(favouriteRepository.findById(id).orElseThrow(() ->

new UsernameNotFoundException("Улюблені не знайдено")));

}

public List<InfoShopRequest> selectAllShop() {

int sum = 0, count = 0;

InfoShop[] infoShops = infoShopRepository.findAll().toArray(new InfoShop[0]);

InfoShopRequest[] infoShopRequests = new InfoShopRequest[infoShops.length];

for (int i = 0; i < infoShops.length; i++) {

InfoShopRequest infoShopRequest = new InfoShopRequest();

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShops[i]).toArray(new ReviewEntity[0]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

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<KategoryDto> selectallKategory() {

Kategory[] kategories = kategoryRepository.findAll().toArray(new Kategory[0]);

KategoryDto[] kategoryDtos = new KategoryDto[kategories.length];

if(kategories.length>0){

for (int i = 0; i < kategories.length; i++) {

KategoryDto kategoryDto = new KategoryDto();

kategoryDto.setId\_kategory(kategories[i].getId\_kategory());

kategoryDto.setName\_kategory(kategories[i].getName\_kategory());

kategoryDtos[i] = kategoryDto;

}

}

return Arrays.asList(kategoryDtos);

}

public InfoShopRequest findShop(Long id) {

int sum = 0, count = 0;

InfoShop infoShop = infoShopRepository.findById(id).orElseThrow(() ->

new UsernameNotFoundException("Магазин не знайдено"));

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShop).toArray(new ReviewEntity[0]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

count++;

sum = sum + entity.getBall();

}

InfoShopRequest infoShopRequest = new InfoShopRequest();

infoShopRequest.setBall(sum / count);

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;

}

else {

InfoShopRequest infoShopRequest = new InfoShopRequest();

infoShopRequest.setBall(0);

infoShopRequest.setResponse(0);

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) {

int sum = 0, count = 0;

InfoShop infoShop = infoShopRepository.findById(id).orElseThrow(() ->

new UsernameNotFoundException("Магазин не знайдено"));

UserEntity user = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShop).toArray(new ReviewEntity[0]);

Favourite [] favourites= favouriteRepository.findAllByUserId(user).toArray(new Favourite[0]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

count++;

sum = sum + entity.getBall();

}

InfoShopRequest infoShopRequest = new InfoShopRequest();

infoShopRequest.setBall(sum / count);

infoShopRequest.setResponse(count);

for (Favourite favourite : favourites) {

if (favourite.getId\_shop().getId().equals(id)) {

infoShopRequest.setSubscribe(true);

infoShopRequest.setSubscribeID(id);

}

}

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;

}

else {

InfoShopRequest infoShopRequest = new InfoShopRequest();

infoShopRequest.setBall(0);

infoShopRequest.setResponse(0);

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 List<InfoShopRequest> findByKategory(Long id){

Kategory kategory= kategoryRepository.findById(id).orElseThrow(() ->

new UsernameNotFoundException("Магазин не знайдено"));

InfoShop[] infoShops=infoShopRepository.findAllByKategoryName(kategory).toArray(new InfoShop[0]);

InfoShopRequest[] infoShopRequests = new InfoShopRequest[infoShops.length];

int sum = 0, count = 0;

for(int i=0;i<infoShops.length;i++){

InfoShopRequest infoShopRequest = new InfoShopRequest();

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShops[i]).toArray(new ReviewEntity[0]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

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.setContacts\_people(infoShops[i].getContacts\_people());

infoShopRequest.setEmail(infoShops[i].getEmail());

infoShopRequest.setLogo(infoShops[i].getLogo());

infoShopRequest.setName\_shop(infoShops[i].getName\_shop());

infoShopRequest.setPhone(infoShops[i].getPhone());

infoShopRequest.setWebsite(infoShops[i].getWebsite());

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.setContacts\_people(infoShops[i].getContacts\_people());

infoShopRequest.setEmail(infoShops[i].getEmail());

infoShopRequest.setLogo(infoShops[i].getLogo());

infoShopRequest.setName\_shop(infoShops[i].getName\_shop());

infoShopRequest.setPhone(infoShops[i].getPhone());

infoShopRequest.setWebsite(infoShops[i].getWebsite());

infoShopRequests[i]=infoShopRequest;

}

}

return Arrays.asList(infoShopRequests);

}

public List<InfoShopRequest> findByKategoryForUser(Long id,String username){

Kategory kategory= kategoryRepository.findById(id).orElseThrow(() ->

new UsernameNotFoundException("Магазин не знайдено"));

UserEntity user = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

InfoShop[] infoShops=infoShopRepository.findAllByKategoryName(kategory).toArray(new InfoShop[0]);

InfoShopRequest[] infoShopRequests = new InfoShopRequest[infoShops.length];

int sum = 0, count = 0;

for(int i=0;i<infoShops.length;i++){

Favourite [] favourites= favouriteRepository.findAllByUserId(user).toArray(new Favourite[0]);

InfoShopRequest infoShopRequest = new InfoShopRequest();

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShops[i]).toArray(new ReviewEntity[0]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

count++;

sum = sum + entity.getBall();

}

for(Favourite favourite:favourites){

if(favourite.getId\_shop().getId().equals(infoShops[i].getId())){

infoShopRequest.setSubscribeID(infoShops[i].getId());

infoShopRequest.setSubscribe(true);

}

}

infoShopRequest.setBall(sum/count);

infoShopRequest.setResponse(count);

infoShopRequest.setId(infoShops[i].getId());

infoShopRequest.setId\_Kategory(infoShops[i].getId\_Kategory());

infoShopRequest.setContacts\_people(infoShops[i].getContacts\_people());

infoShopRequest.setEmail(infoShops[i].getEmail());

infoShopRequest.setLogo(infoShops[i].getLogo());

infoShopRequest.setName\_shop(infoShops[i].getName\_shop());

infoShopRequest.setPhone(infoShops[i].getPhone());

infoShopRequest.setWebsite(infoShops[i].getWebsite());

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.setContacts\_people(infoShops[i].getContacts\_people());

infoShopRequest.setEmail(infoShops[i].getEmail());

infoShopRequest.setLogo(infoShops[i].getLogo());

infoShopRequest.setName\_shop(infoShops[i].getName\_shop());

infoShopRequest.setPhone(infoShops[i].getPhone());

infoShopRequest.setWebsite(infoShops[i].getWebsite());

infoShopRequests[i]=infoShopRequest;

}

}

return Arrays.asList(infoShopRequests);

}

public void favourite(String username, Long id) {

UserEntity userEntity = userRepository.findByUserName(username).orElseThrow(() ->

new UsernameNotFoundException("Користувача не знайдено"));

InfoShop infoShop = infoShopRepository.findById(id).orElseThrow(() ->

new UsernameNotFoundException("Магазин не знайдено"));

Favourite favourite = new Favourite();

favourite.setId\_shop(infoShop);

favourite.setId\_users(userEntity);

favouriteRepository.save(favourite);

}

public InfoShopRequest[] getallFacult(String username){

List<InfoShopRequest> subscribesList = getFavourite(username);

InfoShopRequest[] subscribesArray = subscribesList.toArray(new InfoShopRequest[0]);

InfoShopRequest[] allshops = selectAllShop().toArray(new InfoShopRequest[0]);

if(subscribesArray.length>0) {

int ij=0;

boolean z=true;

InfoShopRequest[] tmp = new InfoShopRequest[allshops.length-subscribesArray.length];

for (InfoShopRequest allshop : allshops) {

for (InfoShopRequest infoShopRequest : subscribesArray) {

if (!allshop.getId().equals(infoShopRequest.getId())) {

z = true;

} else {

z = false;

break;

}

}

if (z) tmp[ij++] = allshop;

z = true;

}

return tmp;

}

else return allshops;

}

}

**А.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);

}

}

**А.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());

}

}

**А.1.4 AdminService**

@Service

public class AdminService {

@Autowired

UserRepository userRepository;

@Autowired

RoleRepository roleRepository;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

KategoryRepository kategoryRepository;

@Autowired

InfoShopRepository infoShopRepository;

@Autowired

ReviewRepository reviewRepository;

public List<KategoryDto> selectallKategory() {

Kategory[] kategories = kategoryRepository.findAll().toArray(new Kategory[0]);

KategoryDto[] kategoryDtos = new KategoryDto[kategories.length];

if(kategories.length>0){

for (int i = 0; i < kategories.length; i++) {

KategoryDto kategoryDto = new KategoryDto();

kategoryDto.setId\_kategory(kategories[i].getId\_kategory());

kategoryDto.setName\_kategory(kategories[i].getName\_kategory());

kategoryDtos[i] = kategoryDto;

}

}

return Arrays.asList(kategoryDtos);

}

public List<InfoShopRequest> selectAllShop() {

int sum = 0, count = 0;

InfoShop[] infoShops = infoShopRepository.findAll().toArray(new InfoShop[0]);

InfoShopRequest[] infoShopRequests = new InfoShopRequest[infoShops.length];

for (int i = 0; i < infoShops.length; i++) {

InfoShopRequest infoShopRequest = new InfoShopRequest();

ReviewEntity[] reviewEntityShop = reviewRepository.findAllById(infoShops[i]).toArray(new ReviewEntity[0]);

if(reviewEntityShop.length>0){

for (ReviewEntity entity : reviewEntityShop) {

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<ReviewRequest> selectallreview(){

ReviewEntity[] reviewEntities = reviewRepository.findAll().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);

}

public List<UserFullInfoDto> selectalluser(){

UserEntity[] userEntities = userRepository.findAll().toArray(new UserEntity[0]);

UserFullInfoDto[] userFullInfoDto = new UserFullInfoDto[userEntities.length];

for(int i=0;i<userEntities.length;i++){

UserFullInfoDto userFullInfoDto1 = new UserFullInfoDto();

userFullInfoDto1.setId(userEntities[i].getId());

userFullInfoDto1.setLogin(userEntities[i].getUserName());

userFullInfoDto1.setEmail(userEntities[i].getEmail());

userFullInfoDto1.setLastname(userEntities[i].getLastname());

userFullInfoDto1.setName(userEntities[i].getName());

byte[] bytesEncoded = Base64.getEncoder().encode(userEntities[i].getPassword().getBytes(StandardCharsets.UTF\_8));

byte [] barr = Base64.getDecoder().decode(bytesEncoded);

userFullInfoDto1.setPassword(userEntities[i].getPassword());

userFullInfoDto1.setSurname(userEntities[i].getSurname());

userFullInfoDto1.setPhone(userEntities[i].getPhone());

userFullInfoDto1.setLogo(userEntities[i].getLogo());

userFullInfoDto[i]=userFullInfoDto1;

}

return Arrays.asList(userFullInfoDto);

}

public RoleDto roleEntityToDto(RoleEntity roleEntity){

RoleDto role = new RoleDto();

role.setName(roleEntity.getName());

role.setId(roleEntity.getId());

return role;

}

public void AddShop( ShopDto shopDto){

Kategory kategory = kategoryRepository.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);

infoShopRepository.save(infoShop);

}

public void AddKategory(String kategory){

Kategory kategory1 = new Kategory();

kategory1.setName\_kategory(kategory);

kategoryRepository.save(kategory1);

}

public void UpdateShop(Long id,ShopDto shopDto){

InfoShop infoShop = infoShopRepository.findById(id).orElseThrow(() ->

new RuntimeException("Магазин не знайдено"));

Kategory kategory = kategoryRepository.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);

infoShopRepository.save(infoShop);

}

public KategoryDto Findkategory(Long id){

Kategory kategory = kategoryRepository.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 = kategoryRepository.findById(id).orElseThrow(() ->

new RuntimeException("Магазин не знайдено"));

kategory1.setName\_kategory(kategory);

kategoryRepository.save(kategory1);

}

public void DeleteShop(Long id ){

infoShopRepository.delete(infoShopRepository.findById(id).orElseThrow(() ->

new RuntimeException("Магазин не знайдено")));

}

public void DeleteReview(Long id){

reviewRepository.delete(reviewRepository.findById(id).orElseThrow(() ->

new RuntimeException("Відгук не знайдено")));

}

public void DeleteKategory(Long id){

kategoryRepository.delete(kategoryRepository.findById(id).orElseThrow(() ->

new RuntimeException("Категорія не знайдено")));

}

public void DeleteUser(Long id){

userRepository.delete(userRepository.findById(id).orElseThrow(() ->

new RuntimeException("Категорія не знайдено")));

}

}

**А.2 Angular(FrontEnd)**

**А.2.1 Main**

**А.2.1.1 Main.html**

<header>

<div class="background\_color">

<div class="main\_text">

<div class="main\_text\_header">

Відгуки і рейтинги магазинів України

</div>

<div class="main\_text\_text">

Характеристика, відгуки, бали про магазин

</div>

<div class="main\_text\_button">

<button (click)="HrefToKatalog()">Дивитися всі</button>

</div>

</div>

<div class="container">

<div class="nav-brand">

<div class="nav-brand-name">SHOP<p>RATE

</div>

<div class="burger">

<input id="menu\_\_toggle" type="checkbox" />

<label class="menu\_\_btn" for="menu\_\_toggle">

<span></span>

</label>

<ul class="menu\_\_box">

<li><a class="menu\_\_item" href="main">Головна</a></li>

<li><a class="menu\_\_item" href="katalog">Магазин</a></li>

<li><a class="menu\_\_item" href="favourite">Улюблені</a></li>

<div \*ngIf="!isLogin">

<li><a class="menu\_\_item" href="login">Авторизація</a></li>

<li><a class="menu\_\_item" href="registrashion">Реєстрація</a></li>

</div>

<div \*ngIf="isLogin" style="display: flex;">

<li class="menu\_\_item">{{username}}</li>

<li class="menu\_\_item"><img \*ngIf="!isImg" alt="" src="../../assets/images/Anonimus.svg">

<img \*ngIf="isImg" alt="" [src]="user.logo"></li>

</div>

</ul>

</div>

<div class="nav-link">

<div class="nav-link-list">

<ul>

<li><a href="main">Головна</a></li>

<li><a href="katalog">Магазин</a></li>

<li><a href="favourite">Улюблені</a></li>

</ul>

</div>

</div>

<div class="nav-link-list-right">

<ul \*ngIf="!isLogin">

<li><a href="login">Вхід</a></li>

<li><a href="registrashion">Реєстрація</a></li>

<li><img alt="" src="../../assets/images/Anonimus.svg">

</li>

</ul>

<ul \*ngIf="isLogin">

<li style="color: black; font-weight: 400;">{{username}}</li>

<li class="logo" data-bs-toggle="modal" data-bs-target="#exampleModal"> <img \*ngIf="!isImg" alt="" src="../../assets/images/Anonimus.svg">

<img \*ngIf="isImg" alt="" [src]="user.logo"></li>

</ul>

</div>

</div>

</div>

</div>

</header>

<main>

<div class="katalog">

<div class="katalog\_block">

<div class="katalog\_block\_header">

Каталог

<div class="katalog\_block\_header\_line"></div>

</div>

<div class="katalog\_block\_list">

<ul >

<li \*ngFor="let kategorys of kategory"><a href="/katalog/{{kategorys.id\_kategory}}">{{kategorys?.name\_kategory}}</a></li>

</ul>

</div>

</div>

</div>

<div class="content\_link">

<ul>

<li class="selectedLink"><a href="main">Популярні</a></li>

<li><a href="main/rating">Рейтинг</a></li>

</ul>

</div>

<div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel" >

<div class="carousel-inner" id="carousel\_block" >

<div class="carousel-item " \*ngFor="let shops of arr;let i=index" [ngClass]="{'active': i == 0 }" id="{{i/4}}" >

<div class="content\_shop" >

<ul>

<li \*ngFor="let item of shops">

<div class="shop\_block">

<div class="shop\_block\_logo">

<img alt="" [src]="item.logo">

</div>

<div class="shop\_block\_text\_block">

<div class="shop\_block\_text\_block\_header">

{{item?.name\_shop}}

</div>

<div class="shop\_block\_text\_block\_star">

<div class="test">

<svg width="28" height="27" viewBox="0 0 28 27" fill="none" xmlns="http://www.w3.org/2000/svg" style="margin-bottom: 13px;">

<path d="M25.6003 12.585C25.8791 12.3773 26.0859 12.0876 26.1917 11.7565C26.2975 11.4254 26.297 11.0695 26.1903 10.7387C26.0836 10.4079 25.8761 10.1187 25.5967 9.91184C25.3174 9.70494 24.9804 9.59063 24.6328 9.58496L17.8828 9.32996C17.8497 9.32767 17.8178 9.31583 17.7912 9.29587C17.7646 9.27591 17.7443 9.24868 17.7328 9.21746L15.4003 2.91746C15.2832 2.59694 15.0703 2.32017 14.7906 2.12463C14.5109 1.92909 14.1779 1.82422 13.8366 1.82422C13.4953 1.82422 13.1623 1.92909 12.8826 2.12463C12.6029 2.32017 12.39 2.59694 12.2728 2.91746L9.94784 9.23996C9.93635 9.27118 9.91608 9.29841 9.88946 9.31837C9.86285 9.33833 9.83103 9.35017 9.79784 9.35246L3.04784 9.60746C2.7003 9.61313 2.36324 9.72744 2.08394 9.93434C1.80464 10.1412 1.59709 10.4304 1.4904 10.7612C1.38371 11.092 1.38324 11.4479 1.48903 11.779C1.59483 12.1101 1.8016 12.3998 2.08034 12.6075L7.37534 16.77C7.40184 16.7908 7.42165 16.819 7.43231 16.851C7.44298 16.883 7.44403 16.9174 7.43534 16.95L5.61284 23.4075C5.5184 23.7363 5.52755 24.0862 5.63905 24.4097C5.75054 24.7331 5.95897 25.0144 6.23598 25.2152C6.51299 25.416 6.84513 25.5265 7.18721 25.5319C7.52929 25.5372 7.86471 25.437 8.14784 25.245L13.7428 21.495C13.7704 21.4759 13.8031 21.4657 13.8366 21.4657C13.8701 21.4657 13.9028 21.4759 13.9303 21.495L19.5253 25.245C19.8046 25.4436 20.1389 25.5503 20.4816 25.5503C20.8243 25.5503 21.1585 25.4436 21.4378 25.245C21.7149 25.0461 21.9236 24.7662 22.035 24.4438C22.1465 24.1215 22.1553 23.7725 22.0603 23.445L20.2228 16.965C20.2131 16.9324 20.2136 16.8977 20.2244 16.8655C20.2351 16.8333 20.2555 16.8051 20.2828 16.785L25.6003 12.585Z" fill="#F12E6D"/>

</svg>

<div class="shop\_block\_text\_block\_ball">

<ul>

<li>{{item.ball}}.0</li>

<li>{{item.response}} відгуків</li>

</ul>

</div>

<div class="kategory">

<ul>

<li> {{item.id\_Kategory.name\_kategory}}</li>

</ul>

</div>

</div>

<div class="shop\_block\_text\_block\_button">

<ul>

<li><a href="infoshop/{{item.id}}">Перейти</a></li>

<li class="favourite" (click)="AddFavourite(item.id)"><svg width="33" height="43" viewBox="0 0 33 43" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M0.773193 5.37509C0.773193 3.94953 1.33719 2.58235 2.3411 1.57433C3.34501 0.566302 4.70661 0 6.12635 0L27.539 0C28.9587 0 30.3203 0.566302 31.3242 1.57433C32.3282 2.58235 32.8922 3.94953 32.8922 5.37509V41.6569C32.892 41.9 32.8262 42.1385 32.7018 42.347C32.5774 42.5555 32.3989 42.7262 32.1855 42.8409C31.9721 42.9556 31.7317 43.0101 31.4899 42.9985C31.2481 42.9869 31.014 42.9096 30.8124 42.7749L16.8327 35.2095L2.8529 42.7749C2.65136 42.9096 2.41723 42.9869 2.17543 42.9985C1.93363 43.0101 1.69322 42.9556 1.47981 42.8409C1.2664 42.7262 1.08797 42.5555 0.963536 42.347C0.839102 42.1385 0.773318 41.9 0.773193 41.6569V5.37509ZM6.12635 2.68754C5.41648 2.68754 4.73568 2.97069 4.23373 3.47471C3.73177 3.97872 3.44977 4.66231 3.44977 5.37509V39.1468L16.0913 32.4763C16.3109 32.3295 16.5688 32.2512 16.8327 32.2512C17.0965 32.2512 17.3544 32.3295 17.5741 32.4763L30.2156 39.1468V5.37509C30.2156 4.66231 29.9336 3.97872 29.4316 3.47471C28.9297 2.97069 28.2489 2.68754 27.539 2.68754H6.12635Z" fill="#474E95"/>

</svg></li>

</ul>

</div>

</div>

</div>

</div>

</li>

</ul>

</div>

</div>

</div>

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content" style="width: 372px;height: 270px;">

<div class="modal-header">

<a href="profile">Профіль</a>

</div>

<div class="modal-body">

<ul>

<li><a href="favourite">Улюблені</a></li>

<li><a href="accountSetting">Редагувати</a></li>

<li><a href="#" (click)="LogOut()">Вихід</a></li>

</ul>

</div>

</div>

</div>

</div>

<div class="arrow\_block">

<div class="Arrow">

<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev">

<div class="styleArrowRight">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Предыдущий</span>

</div>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next">

<div class="styleArrowLeft">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Следующий</span>

</div>

</button>

</div>

</div>

</div>

</main>

<app-footer></app-footer>

**А.2.1.2 Main.css**

\*{

padding: 0;

margin: 0;

}

body{

height: 100%;

overflow-y: auto;

}

header {

left: 132px;

top: 21px;

}

.background\_color{

position: relative;

background-color: #474E95;

width: 953px;

height: 722px;

}

main{

max-width: 1170px;

margin-left: 133px;

}

.container{

width: 100%;

max-width: 1170px;

margin: 0 auto;

}

.nav-brand{

width: 1173px;

height: 116px;

left: 132px;

top: 21px;

position: absolute;

display: flex;

}

.nav-brand-name{

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 64px;

line-height: 85%;

color: white;

margin-top: 21px;

margin-bottom: 21px;

}

.nav-brand-name p{

margin-top: 7px;

font-family: Montserrat;

font-style: normal;

font-weight: 100;

font-size: 64px;

line-height: 50%;

color: white;

}

.nav-link{

display: inline-flex;

margin-left: 63px;

}

.nav-link-list{

margin-top: 48px;

margin-bottom: 90px;

}

.nav-link-list ul{

list-style: none;

display: inline-flex;

}

.nav-link-list li{

text-transform: uppercase;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 16px;

line-height: 20px;

display: flex;

align-items: center;

text-align: center;

color: #FFFFFF;

margin-right: 84px;

}

.nav-link-list a{

text-decoration: none;

color: #FFFFFF;

}

.nav-link-list a:focus{

font-weight: normal;

}

.nav-link-list-right{

text-align: right;

margin-left: 120px;

margin-top: 33px;

margin-bottom: 75px;

}

.nav-link-list-right ul{

list-style: none;

display: inline-flex;

}

.nav-link-list-right li{

text-align: right;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 16px;

line-height: 20px;

display: flex;

align-items: center;

text-align: center;

color: #000000;

color: white;

}

.nav-link-list-right li:last-child{

margin-right: 135px;

margin-left: 40px;

}

.nav-link-list-right li:first-child{

margin-right: 53px;

}

.logo img{

width: 50px;

height: 50px;

border-radius: 60px;

}

.nav-link-list-right a{

text-decoration: none;

color: #000000;

font-weight: 400;

}

.nav-link-list-right a:focus{

font-weight: 500;

}

/\* Body \*/

.main\_text{

position: absolute;

width: 638px;

height: 196px;

left: 133px;

top: 235px;

}

.main\_text\_header{

position: absolute;

width: 638px;

height: 96px;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 48px;

line-height: 100%;

display: flex;

align-items: center;

color: #FFFFFF;

}

.main\_text\_text{

position: absolute;

width: 301px;

height: 69px;

top: 132px;

font-family: Montserrat;

font-style: normal;

font-weight: 500;

font-size: 16px;

line-height: 142%;

display: flex;

align-items: center;

color: #FFFFFF;

}

.main\_text\_button{

position: absolute;

width: 222px;

height: 51px;

top: 267px;

}

.main\_text\_button button{

position: absolute;

width: 222px;

height: 51px;

left: 0px;

outline: 0;

cursor: pointer;

border: 2px solid #FFFFFF;

box-sizing: border-box;

background-color: transparent;

border-radius: 50px;

font-family: Montserrat;

font-style: normal;

font-weight: 600;

font-size: 14px;

line-height: 100%;

text-transform: uppercase;

align-items: center;

text-align: center;

color: #FFFFFF;

transition: all 0.45s ease-out;

}

.main\_text\_button button:hover{

background-color: white;

color: #474E95;

box-shadow: 0px 20px 50px rgba(0, 0, 0, 0.1);

}

.main\_text\_button button:hover{

color: #474E95;

}

.katalog{

position: absolute;

width: 442px;

height: 551px;

left: 863px;

top: 142px;

}

.katalog\_block{

position: absolute;

display: block;

width: 442px;

height: 551px;

left: 0px;

top: 0px;

background: #FAFAFA;

box-shadow: 0px 20px 50px rgba(0, 0, 0, 0.1);

}

.katalog\_block\_header{

text-transform: uppercase;

padding: 50px;

padding-bottom: 24px;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 24px;

line-height: 29px;

display: flex;

color: #F12E6D;

display: block;

}

.katalog\_block\_header\_line{

margin-top: 30px;

border-bottom: 1px solid #D5D5D5;

}

.katalog\_block\_list{

margin-left: 100px;

}

.katalog\_block\_list ul{

display: inline-block;

list-style: none;

}

.katalog\_block\_list li{

text-align: start;

margin: 15px;

font-family: Montserrat;

font-style: normal;

font-size: 16px;

line-height: 100%;

height: 100%;

margin-bottom: 20px;

}

.katalog\_block\_list li:last-child{

margin-bottom: 0;

}

.katalog\_block\_list li:hover{

text-transform: uppercase;

background: url(../../assets/images/arrow.svg) no-repeat 100% ;

padding-right: 30px;

font-weight: bold;

transition: all 0.35s ease-out;

}

.katalog\_block\_list a{

text-decoration: none;

color: #000000;

text-align: start;

margin: 15px;

font-family: Montserrat;

font-style: normal;

font-size: 16px;

line-height: 100%;

height: 100%;

margin-bottom: 20px;

}

.burger{

display: none;

}

.content\_link{

margin-top: 85px;

}

.content\_link ul{

display: inline-flex;

list-style: none;

}

.content\_link li{

margin-right: 100px;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 24px;

line-height: 100%;

display: flex;

align-items: center;

}

.content\_link a{

text-decoration: none;

color: #7A7A7A;

font-weight: 600;

}

.content\_link li:last-child{

border-left: 1px solid #D5D5D5;

padding-inline: 100px;

}

.arrow\_block{

width: 200px;

height: 200px;

position: absolute;

right: 0;

top: 10px;

}

.Arrow{

position: absolute;

right: 60px;

top: -94px;

z-index: 1;

display: flex;

justify-content: flex-end;

width: 50%;

text-align: center;

transition: opacity .15s ease

}

.styleArrowLeft span{

width: 53px;

height: 53px;

background-image:url("../../assets/images/Вправонезар.svg");

}

.styleArrowRight span{

width: 53px;

height: 53px;

background-image:url("../../assets/images/Влевонезар.svg")

}

.styleArrowLeft span:active{

background-image:url("../../assets/images/Вправо.svg")

}

.styleArrowRight span:active{

background-image:url("../../assets/images/Влево.svg")

}

.content\_shop ul{

display: inline-flex;

flex-wrap: wrap;

list-style: none;

}

.content\_shop li{

margin-bottom: 30px;

margin-right: 30px;

}

.shop\_block{

background: #FAFAFA;

box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.06);

}

.shop\_block\_logo{

text-align: center;

}

.shop\_block\_logo img{

max-width: 213.79px;

max-height: 43px;

margin-top: 28px;

filter: drop-shadow(0px 4px 6px rgba(0, 0, 0, 0.06));

}

.shop\_block\_text\_block{

padding: 20px;

margin-top: 20px;

}

.shop\_block\_text\_block\_header{

font-family: Montserrat;

font-style: normal;

font-weight: 500;

font-size: 18px;

line-height: 100%;

display: flex;

align-items: center;

color: #0A0A0A;

}

.shop\_block\_text\_block\_star{

margin-top: 20px;

}

.shop\_block\_text\_block\_ball{

margin-left: 5px;

height: 43px;

display: inline-flex;

}

.shop\_block\_text\_block\_ball ul{

display: inline-flex;

text-decoration: none;

margin-bottom: 10px;

}

.shop\_block\_text\_block\_ball li{

margin-right: 10px;

}

.shop\_block\_text\_block\_ball li:first-child{

border-right: 2px solid #414040;

padding-inline: 9px;

font-family: Montserrat;

font-style: normal;

font-weight: 600;

font-size: 24px;

line-height: 100%;

align-items: center;

color: #414040;

}

.shop\_block\_text\_block\_ball li:last-child{

font-family: Montserrat;

font-style: normal;

font-weight: normal;

font-size: 14px;

margin-top: 5px;

line-height: 100%;

align-items: center;

color: #414040;

}

.kategory{

margin-left: 21px;

}

.kategory ul{

list-style-image: url("../../assets/images/Ellipse 2.svg");

}

.kategory li{

font-family: Montserrat;

text-transform: uppercase;

font-style: normal;

font-weight: 500;

font-size: 10px;

line-height: 100%;

color: #000000;

}

.shop\_block\_text\_block\_button{

height: 50px;

}

.shop\_block\_text\_block\_button a{

cursor: pointer;

text-decoration: none;

border: 2px solid #F12E6D;

box-sizing: border-box;

padding: 12px 40px;

background: transparent;

border-radius: 50px;

font-family: Montserrat;

font-style: normal;

text-transform: uppercase;

font-weight: 600;

font-size: 12px;

line-height: 100%;

display: flex;

align-items: center;

color: #F12E6D;

}

.modal{

left: 1150px;

top:110px;

}

.modal-body li:first-child{

margin-top: 24px;

}

.modal-body li{

font-family: Montserrat;

font-style: normal;

font-weight: normal;

font-size: 16px;

line-height: 100%;

margin-left: 40px;

margin-bottom: 24px;

}

.modal-body a{

text-decoration: none;

color: #000000;

}

.modal-body a:hover{

text-transform: uppercase;

background: url(../../assets/images/arrow.svg) no-repeat 100% ;

padding-right: 30px;

font-weight: bold;

transition: all 0.35s ease-out;

}

.modal-body li:last-child{

border-top: 1px solid #D5D5D5 ;

padding-top: 24px;

margin-bottom: 40px;

}

.modal-header{

margin-left: 29px;

margin-top: 40px;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 16px;

line-height: 20px;

color: #000000;

text-transform: uppercase;

padding-bottom: 14px;

}

.modal-header a{

text-decoration: none;

color: #000000;

}

.modal-header a:hover{

text-transform: uppercase;

background: url(../../assets/images/arrow.svg) no-repeat 100% ;

padding-right: 30px;

font-weight: bold;

transition: all 0.35s ease-out;

}

.modal-body ul{

list-style: none;

}

.shop\_block\_text\_block\_button li{

margin-right: 20px;

}

.shop\_block\_text\_block\_button a:hover{

background-color: #F12E6D;

color: white;

filter: drop-shadow(0px 20px 50px rgba(0, 0, 0, 0.1));

transition: all 0.45s ease-out;

}

.shop\_block\_text\_block\_button li{

margin-right: 20px;

}

.favourite{

cursor: pointer;

}

footer{

width: 100%;

margin-top: 50px;

background-color: #474E95;

text-align: center;

color: white;

}

.footer\_text{

padding: 40px;

}

#menu\_\_toggle {

opacity: 0;

}

#menu\_\_toggle:checked ~ .menu\_\_btn > span {

transform: rotate(45deg);

}

#menu\_\_toggle:checked ~ .menu\_\_btn > span::before {

top: 0;

transform: rotate(0);

}

#menu\_\_toggle:checked ~ .menu\_\_btn > span::after {

top: 0;

transform: rotate(90deg);

}

#menu\_\_toggle:checked ~ .menu\_\_box {

visibility: visible;

left: 0;

}

.menu\_\_btn {

display: flex;

align-items: center;

position: absolute;

top: 20px;

left: 200px;

width: 26px;

height: 26px;

cursor: pointer;

z-index: 1;

}

.menu\_\_btn > span,

.menu\_\_btn > span::before,

.menu\_\_btn > span::after {

display: block;

position: absolute;

width: 100%;

height: 2px;

background-color: #FFFFFF;

transition-duration: .25s;

}

.menu\_\_btn > span::before {

content: '';

top: -8px;

}

.menu\_\_btn > span::after {

content: '';

top: 8px;

}

.menu\_\_box {

display: block;

position: absolute;

visibility: hidden;

top: 0;

left: 100%;

width: 250px;

height: 350px;

margin: 0;

padding: 75px 0;

list-style: none;

background-color: #474E95;

box-shadow: 1px 0px 6px rgba(0, 0, 0, .2);

transition-duration: .25s;

}

.menu\_\_item {

display: block;

padding: 12px 24px;

color: #333;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 20px;

line-height: 20px;

color: #FFFFFF;

text-decoration: none;

transition-duration: .25s;

}

.menu\_\_item img{

width: 50px;

height: 50px;

border-radius: 60px;

}

.menu\_\_item:hover {

font-weight: normal;

}

#carouselExampleControls{

margin-top: 72px;

}

.selectedLink a{

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 24px;

line-height: 100%;

color: #474E95;

}

@media (min-width: 320px) and (max-width: 480px) {

.background\_color{

max-width: 480px;

overflow: hidden;

}

.nav-brand{

max-width: 480px;

}

.katalog\_block\_list{

margin-left: 0;

}

main{

overflow: hidden;

margin-left: 0px;

}

.container{

max-width: 480px;

overflow: hidden;

}

header{

max-width: 480px;

overflow: hidden;

}

.nav-brand-name{

font-size: 30pt;

text-align: center;

margin-left:-50px;

}

.nav-brand-name p::after{

margin-top: 50px;

}

.nav-brand-name p{

font-size: 30pt;

}

.main\_text{

max-width: 400px;

top: 141px;

left: 0px;

text-align: center;

}

.main\_text\_header{

max-width: 400px;

height: 96px;

left: 44px;

text-align: left;

top: 20px;

font-size: 20pt;

}

.main\_text\_button button{

left: 44px;

top:-40px;

}

.main\_text\_text{

left: 44px;

text-align: left;

}

.nav-link{

display: none;

}

.burger{

display: block;

margin-left: 100px;

margin-top: 40px;

}

.nav-link-list{

display: none;

}

.nav-link-list-right{

display: none;

}

.Arrow{

top:-50px;

width: 40%;

left: -50px;

right: 30px;

}

.katalog{

position: absolute;

max-width: 400px;

max-height: 440px;

left:0px;

top: 472px;

}

.katalog\_block{

padding: 5px;

width: 300px;

height:460px;

left: 44px;

top: -20px;

}

.katalog\_block\_header{

font-size: 16pt;

padding: 10px;

text-align: center;

}

#carouselExampleControls{

margin-top: 100px;

margin-left: 60px;

}

.katalog\_block\_header\_line{

margin-top: 10px;

}

.katalog\_block\_list a:hover{

font-weight: normal;

font-size: 12pt;

}

.content\_link{

margin-left: 40px;

margin-top: 240px;

margin-right: 0px;

padding: 0;

}

.content\_link li{

margin-right:30px;

font-size: 13pt;

line-height: 100%;

}

.content\_link li:last-child{

margin-right: 0;

padding-inline: 50px;

}

.content\_link ul{

display: flex;

}

.content\_link li:first-child{

border: 0;

padding-inline: 0px;

}

.content\_shop{

max-width: 250px;

margin-left: 10px;

}

.shop\_block\_text\_block\_button ul{

flex-wrap: nowrap;

}

footer{

margin-top: 183px ;

}

}

**А.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='data:image/jpeg;base64,'+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 ='data:image/jpeg;base64,'+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='data:image/jpeg;base64,'+this.shop[j].logo;

}

})

}

LogOut(){

this.authService.Logout()

}

HrefToKatalog(){

this.\_router.navigate(['katalog'])

}

}

**А.2.2 Login**

**А.2.2.1 Login.html**

<header>

<div class="container">

<div class="nav-brand">

<div class="nav-register">

Немає акаунта?

<a href="/registrashion">Зареєструватися</a>

</div>

<div class="nav-brand-name"> SHOP<p>RATE</div>

</div>

</div>

</header>

<main>

<div class="alert-danger " \*ngIf="FailToLogin">

<div class="alert-danger-text">

Перевірте свої данні

</div>

</div>

<div class="login\_block">

<div class="login\_block\_header">

Авторизація

</div>

<form class="login\_block\_form" [formGroup]='loginForm' (ngSubmit)="OnLogin()">

<label for="username">Логін</label>

<input type="text" [formControlName]="'username'" name="username">

<label for="password">Пароль</label>

<input type="password" [formControlName]="'password'" name="password">

<input type="submit" value="Увійти">

</form>

</div>

</main>

**А.2.2.2 Login.сss**

\*{

padding: 0;

margin: 0;

}

.nav-brand{

margin-top: 28px;

margin-bottom: 42px;

}

button{

outline: none;

}

.container{

max-width: 1170px;

margin: 0 auto;

}

.nav-brand-name{

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 32px;

line-height: 85%;

color: #474E95;

}

.nav-brand-name p{

margin-top: 7px;

font-family: Montserrat;

font-style: normal;

font-weight: 500;

font-size: 32px;

line-height: 50%;

color: #474E95;

}

.nav-register{

margin-top: 20px;

text-align: center;

float: right;

font-family: Montserrat;

font-style: normal;

font-weight: normal;

font-size: 14px;

line-height: 100%;

color: #414040;

}

.nav-register a{

text-decoration: none;

}

.login\_block{

text-align: center;

width: 470px;

height: 359px;

margin: auto;

padding: 20px 20px;

margin-top: 50px;

}

.login\_block\_header{

text-align: left;

font-family: Montserrat;

font-style: normal;

font-weight: bold;

font-size: 24px;

line-height: 29px;

text-transform: uppercase;

color: #F12E6D;

}

.login\_block\_header:first-of-type{

padding-bottom: 15px;

width: 470px;

border-bottom:1px solid #D5D5D5;

}

.login\_block\_form{

margin-top: 32px;

text-align: left;

}

.alert-warning{

color: red;

font-family: Montserrat;

font-style: normal;

font-weight: normal;

font-size: 16px;

line-height: 100%;

}

.login\_block\_form input[type=text]{

background: #F2F2F2;

border: 1px solid #D5D5D5;

box-sizing: border-box;

padding: 10px;

margin-bottom: 32px;

font-family: Montserrat;

font-style: normal;

font-weight: normal;

font-size: 16px;

line-height: 100%;

color: #000000;

width: 470px;

outline: none;

height: 43px;

box-shadow: 0px 4px 6px rgb(0 0 0 / 6%)

}

.login\_block\_form input[type=password]{

background: #F2F2F2;

border: 1px solid #D5D5D5;

box-sizing: border-box;

padding: 10px;

font-family: Montserrat;

font-style: normal;

font-weight: normal;

font-size: 16px;

line-height: 100%;

color: #000000;

width: 470px;

outline: none;

margin-bottom: 42px;

height: 43px;

box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.06);

}

.login\_block\_form input[type=password]:focus,.login\_block\_form input[type=text]:focus{

background: #FFFFFF;

}

.login\_block\_form label{

font-family: Montserrat;

font-style: normal;

font-weight: 500;

font-size: 16px;

line-height: 100%;

margin-bottom: 8px;

display: flex;

align-items: center;

color: #000000;

}

.login\_block\_form input[type=submit]{

padding: 10px 50px;

text-transform: uppercase;

border-radius: 50px;

font-family: Montserrat;

font-style: normal;

font-weight: 600;

font-size: 12px;

background-color: transparent;

line-height: 100%;

border: 2px solid #F12E6D;

cursor: pointer;

color: #F12E6D;

margin-bottom: 30px;

}

.login\_block\_form input[type=submit]:hover{

background-color: #F12E6D;

color: white;

filter: drop-shadow(0px 20px 50px rgba(0, 0, 0, 0.1));

transition: all 0.45s ease-out;

}

.alert-danger{

margin-top: 20px;

text-align: center;

width: 100%;

height: 50px;

color: #842029;

background-color: #ED7B7B;

background-color: #ED7B7B;

margin-bottom: 20px;

}

.alert-danger-text{

margin: auto;

font-family: Montserrat;

font-style: normal;

padding: 17px;

color: white;

font-weight: 600;

font-size: 14px;

line-height: 100%;

}

@media (min-width: 320px) and (max-width: 480px){

.nav-brand{

text-align: left;

margin-left: 10px;

}

.nav-register{

text-align: right;

margin-right: 10px;

}

.login\_block{

width: 350px;

height: 450px;

margin-left: 5px;

}

.alert-danger{

width: 350px;

margin-left: 30px;

}

.alert-danger-text{

font-size: 14px;

}

.login\_block\_form input[type=text],.login\_block\_form input[type=password]{

width: 350px;

}

}

**А.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)

}

)

}

}

**А.2.3 Favourite**

**А.2.3.1 Favourite.html**

<app-header></app-header>

<main>

<div class="content\_account">

<div class="content\_account\_block">

<div class="content\_account\_block\_text">

<div class="content\_account\_block\_text\_img">

<img \*ngIf="!isImg" alt="" src="../../assets/images/Anonimus.svg">

<img\*ngIf="isImg" alt="" [src]="img" width="50px" height="50px">

</div>

<div class="content\_account\_block\_text\_details">

<div class="content\_account\_block\_text\_username">

{{username}}

</div>

<div class="content\_account\_block\_text\_name\_surname">

{{user?.surname}} {{user?.name}}

</div>

<div class="content\_account\_block\_text\_button">

<button (click)="HrefToProfile()">Змінити</button>

</div>

</div>

</div>

</div>

</div>

<div class="content\_shop\_list">

<div class="content\_link\_list">

<ul>

<li class="selectedLink"><a href="favourite">Улюблені</a></li>

<li><a href="accountSetting">Редагувати</a></li>

<li><a href="accountLoginAndPassword">Пароль і логін</a></li>

</ul>

</div>

</div>

<div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel">

<div class="carousel-inner" id="carousel\_block">

<div class="empty\_favourite" \*ngIf="arr?.length==0">

На жаль не має у вас улюблених магазинів :)

</div>

<div class="carousel-item " \*ngFor="let shops of arr;let i=index"[ngClass]="{'active': i == 0 }" id="{{i/4}}" >

<div class="content\_shop" >

<ul>

<li \*ngFor="let item of shops">

<div class="shop\_block">

<div class="shop\_block\_logo">

<img alt="" [src]="item.logo">

</div>

<div class="shop\_block\_text\_block">

<div class="shop\_block\_text\_block\_header">

{{item?.name\_shop}}

</div>

<div class="shop\_block\_text\_block\_star">

<div class="test">

<svg width="28" height="27" viewBox="0 0 28 27" fill="none" xmlns="http://www.w3.org/2000/svg" style="margin-bottom: 13px;">

<path d="M25.6003 12.585C25.8791 12.3773 26.0859 12.0876 26.1917 11.7565C26.2975 11.4254 26.297 11.0695 26.1903 10.7387C26.0836 10.4079 25.8761 10.1187 25.5967 9.91184C25.3174 9.70494 24.9804 9.59063 24.6328 9.58496L17.8828 9.32996C17.8497 9.32767 17.8178 9.31583 17.7912 9.29587C17.7646 9.27591 17.7443 9.24868 17.7328 9.21746L15.4003 2.91746C15.2832 2.59694 15.0703 2.32017 14.7906 2.12463C14.5109 1.92909 14.1779 1.82422 13.8366 1.82422C13.4953 1.82422 13.1623 1.92909 12.8826 2.12463C12.6029 2.32017 12.39 2.59694 12.2728 2.91746L9.94784 9.23996C9.93635 9.27118 9.91608 9.29841 9.88946 9.31837C9.86285 9.33833 9.83103 9.35017 9.79784 9.35246L3.04784 9.60746C2.7003 9.61313 2.36324 9.72744 2.08394 9.93434C1.80464 10.1412 1.59709 10.4304 1.4904 10.7612C1.38371 11.092 1.38324 11.4479 1.48903 11.779C1.59483 12.1101 1.8016 12.3998 2.08034 12.6075L7.37534 16.77C7.40184 16.7908 7.42165 16.819 7.43231 16.851C7.44298 16.883 7.44403 16.9174 7.43534 16.95L5.61284 23.4075C5.5184 23.7363 5.52755 24.0862 5.63905 24.4097C5.75054 24.7331 5.95897 25.0144 6.23598 25.2152C6.51299 25.416 6.84513 25.5265 7.18721 25.5319C7.52929 25.5372 7.86471 25.437 8.14784 25.245L13.7428 21.495C13.7704 21.4759 13.8031 21.4657 13.8366 21.4657C13.8701 21.4657 13.9028 21.4759 13.9303 21.495L19.5253 25.245C19.8046 25.4436 20.1389 25.5503 20.4816 25.5503C20.8243 25.5503 21.1585 25.4436 21.4378 25.245C21.7149 25.0461 21.9236 24.7662 22.035 24.4438C22.1465 24.1215 22.1553 23.7725 22.0603 23.445L20.2228 16.965C20.2131 16.9324 20.2136 16.8977 20.2244 16.8655C20.2351 16.8333 20.2555 16.8051 20.2828 16.785L25.6003 12.585Z" fill="#F12E6D"/>

</svg>

<div class="shop\_block\_text\_block\_ball">

<ul>

<li>{{item.ball}}.0</li>

<li>{{item.response}} відгуків</li>

</ul>

</div>

<div class="kategory">

<ul>

<li> {{item.id\_Kategory.name\_kategory}}</li>

</ul>

</div>

</div>

<div class="shop\_block\_text\_block\_button">

<ul>

<li><a href="infoshop/{{item.id}}">Перейти</a></li>

<li class="favourite" ><svg width="44" height="55" viewBox="0 0 44 55" fill="none" xmlns="http://www.w3.org/2000/svg" (click)="DeleteFavourite(item.subscribeID)">

<g filter="url(#filter0\_d)">

<path d="M6 7.37509C6 5.94953 6.5619 4.58235 7.5621 3.57433C8.56229 2.5663 9.91885 2 11.3333 2H32.6667C34.0812 2 35.4377 2.5663 36.4379 3.57433C37.4381 4.58235 38 5.94953 38 7.37509V43.6569C37.9999 43.9 37.9343 44.1385 37.8104 44.347C37.6864 44.5555 37.5086 44.7262 37.296 44.8409C37.0834 44.9556 36.8439 45.0101 36.603 44.9985C36.3621 44.9869 36.1288 44.9096 35.928 44.7749L22 37.2095L8.072 44.7749C7.87121 44.9096 7.63795 44.9869 7.39704 44.9985C7.15614 45.0101 6.91662 44.9556 6.704 44.8409C6.49138 44.7262 6.31361 44.5555 6.18964 44.347C6.06566 44.1385 6.00012 43.9 6 43.6569V7.37509ZM11.3333 4.68754C10.6261 4.68754 9.94781 4.97069 9.44772 5.47471C8.94762 5.97872 8.66667 6.66231 8.66667 7.37509V41.1468L21.2613 34.4763C21.4802 34.3295 21.7371 34.2512 22 34.2512C22.2629 34.2512 22.5198 34.3295 22.7387 34.4763L35.3333 41.1468V7.37509C35.3333 6.66231 35.0524 5.97872 34.5523 5.47471C34.0522 4.97069 33.3739 4.68754 32.6667 4.68754H11.3333Z" fill="#474E95"/>

<path d="M8.5064 23.795L24.2873 32.9775L8.5064 42.1599V23.795Z" fill="#474E95"/>

<path d="M20.8899 35.0062L36.0608 24.8329L37.2172 43.1608L20.8899 35.0062Z" fill="#474E95"/>

<path d="M7.6732 4.52945H36.1176V34.8829H7.6732V4.52945Z" fill="#474E95"/>

</g>

<path d="M20.0044 25.3888L14.9911 20.4037C14.9622 20.3749 14.9393 20.3407 14.9236 20.3031C14.908 20.2655 14.8999 20.2251 14.8999 20.1843C14.8999 20.1435 14.908 20.1032 14.9236 20.0655C14.9393 20.0279 14.9622 19.9937 14.9911 19.9649L16.2568 18.6979C16.3777 18.577 16.5732 18.577 16.6941 18.6979L19.9946 21.9787C20.1155 22.0996 20.3124 22.0982 20.4333 21.9773L27.7261 14.6142C27.8471 14.4918 28.0439 14.4918 28.1663 14.6127L29.4333 15.8798C29.5543 16.0007 29.5543 16.1962 29.4347 16.3171L21.4894 24.3342L21.4908 24.3356L20.4432 25.3888C20.3222 25.5098 20.1253 25.5098 20.0044 25.3888Z" fill="#FFFCFC" stroke="white" stroke-width="0.75" stroke-miterlimit="10"/>

<defs>

<filter id="filter0\_d" x="0" y="0" width="44" height="55" filterUnits="userSpaceOnUse" color-interpolation-filters="sRGB">

<feFlood flood-opacity="0" result="BackgroundImageFix"/>

<feColorMatrix in="SourceAlpha" type="matrix" values="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 127 0"/>

<feOffset dy="4"/>

<feGaussianBlur stdDeviation="3"/>

<feColorMatrix type="matrix" values="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.06 0"/>

<feBlend mode="normal" in2="BackgroundImageFix" result="effect1\_dropShadow"/>

<feBlend mode="normal" in="SourceGraphic" in2="effect1\_dropShadow" result="shape"/>

</filter>

</defs>

</svg>

</li>

</ul>

</div>

</div>

</div>

</div>

</li>

</ul>

</div>

</div>

</div>

<div class="Arrow" \*ngIf="arr?.length!=0">

<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls"data-bs-slide="prev">

<div class="styleArrowLeft">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Предыдущий</span>

</div>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls"data-bs-slide="next">

<div class="styleArrowRight">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Следующий</span>

</div>

</button>

</div>

</div>

</main>

<app-footer></app-footer>

**А.2.3.2Favourite.css**

\*{

padding:0;

margin:0;

}

.burger{

display:none;

}

button{

outline:none;

}

header{

width:100%;

background:#474E95;

}

.container{

max-width:1170px;

margin:0auto;

}

.nav-brand{

width:100%;

display:flex;

}

.nav-brand-name{

font-family:Montserrat;

font-style:normal;

font-weight:bold;

font-size:64px;

line-height:85%;

color:white;

margin-top:21px;

margin-bottom:21px;

}

.nav-brand-namep{

margin-top:7px;

font-family:Montserrat;

font-style:normal;

font-weight:100;

font-size:64px;

line-height:50%;

color:white;

}

.nav-link{

display:inline-flex;

margin-left:63px;

}

.nav-link-list{

margin-top:48px;

margin-bottom:90px;

}

.nav-link-listul{

list-style:none;

display:inline-flex;

}

.nav-link-listli{

text-transform:uppercase;

font-family:Montserrat;

font-style:normal;

font-weight:bold;

font-size:16px;

line-height:20px;

display:flex;

align-items:center;

text-align:center;

color:#FFFFFF;

margin-right:84px;

}

.nav-link-lista{

text-decoration:none;

color:#FFFFFF;

}

.nav-link-list-right{

text-align:right;

margin-left:292px;

margin-top:33px;

margin-bottom:75px;

}

.nav-link-list-rightul{

list-style:none;

display:inline-flex;

}

.nav-link-list-rightli{

text-align:right;

font-family:Montserrat;

font-style:normal;

font-weight:bold;

font-size:16px;

line-height:20px;

display:flex;

align-items:center;

text-align:center;

color:#000000;

color:white;

}

.nav-link-list-rightli:last-child{

margin-right:135px;

}

.nav-link-list-rightli:first-child{

margin-right:40px;

}

.logoimg{

width:50px;

height:50px;

border-radius:60px;

}

.nav-link-list-righta{

text-decoration:none;

color:#000000;

}

.content\_account{

top:140px;

left:450px;

position:absolute;

margin-top:-30px;

}

.content\_account\_block{

width:600px;

height:250px;

background:#FAFAFA;

margin:auto;

box-shadow:0px20px50pxrgba(0,0,0,0.1);

}

.content\_account\_block\_text{

margin:auto;

padding:50px;

display:flex;

margin-left:50px;

}

.content\_account\_block\_text\_imgimg{

width:140px;

height:140px;

margin-left:30px;

border-radius:60px;

}

.content\_account\_block\_text\_img{

text-align:left;

}

.content\_account\_block\_text\_details{

margin-left:30px;

}

.content\_account\_block\_text\_username{

font-family:Montserrat;

font-style:normal;

font-weight:bold;

font-size:24px;

line-height:100%;

display:flex;

align-items:center;

color:#0A0A0A;

}

.content\_account\_block\_text\_name\_surname{

margin-top:20px;

text-align:left;

font-family:Montserrat;

font-style:normal;

font-weight:normal;

font-size:16px;

line-height:100%;

display:flex;

align-items:center;

color:#000000;

}

.content\_account\_block\_text\_button{

text-align:left;

margin-top:30px;

}

.content\_account\_block\_text\_buttonbutton{

width:125px;

height:43px;

text-transform:uppercase;

background-color:transparent;

border:2pxsolid#F12E6D;

box-sizing:border-box;

color:#F12E6D;

font-family:Montserrat;

font-style:normal;

font-weight:600;

font-size:12px;

line-height:100%;

cursor:pointer;

filter:drop-shadow(0px20px50pxrgba(0,0,0,0.1));

border-radius:50px;

}

.content\_account\_block\_text\_buttonbutton:hover{

background-color:#F12E6D;

color:white;

filter:drop-shadow(0px20px50pxrgba(0,0,0,0.1));

transition:all0.45sease-out;

}

**А.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='data:image/jpeg;base64,'+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='data:image/jpeg;base64,'+this.user.logo;

if(this.user.logo){

this.isImg=true;

}

if(!this.user.logo){

this.isImg=false;

}

});

}

HrefToProfile(){

this.router.navigate(['accountSetting']);

}

}

**Анотація**

Пояснювальна записка складається з 49 сторінок, 23 рисунків, 5 таблиць, 5 діаграм, 1 додатка та 7 джерел.

Об’єкт проектування – Вебсайт «Shop Rate»

Мета проектування – розробити вебсайт для визначення кращого магазину.

Метод проектування – FIGMA, HTML, CSS, Java, JavaScript,Typescript, Angular, MAMP.

У результаті дипломного проектування створена база в яку приходять дані для рейтенгу магазину і їх оцінки, створений вебсайт.

Ключові слова: ВЕБСАЙТ, FIGMA, HTML, CSS, Java, JavaScript, Typescript, Angular, MAMP.

**Декларація академічної чесності**

Я, Попюк Роман Артурович, підтверджую, що мій дипломний проект на тему вебсайт «Shop Rate» написаний моїми власними словами, що відбивають особисті думки, судження та висновки. Я розумію та пам’ятаю, що під час проведення дослідження та оформлення його результатів необхідно уникати плагіату – представлення думок та слів іншої людини як своїх, з особливою увагою та обережністю працювати із інформацією, яка надана у вільному доступі. У разі використання й / або цитування продуктів інтелектуальної праці інших осіб мною було обов’язково подано посилання на джерело інформації.

Я усвідомлюю, що у разі виявлення порушення цих правил розроблений мною дипломний проект буде миттєво відхилений від права повторного його захисту із одночасним виставленням оцінки «незадовільно».

*19 червня* 2021 р.

**БІБЛІОГРАФІЧНА ДОВІДКА**

Тема дипломного проекту: Вебсайт «Shop Rate»

Обсяг пояснювальної записки: 94 аркушів

Дата закінчення проекту 19 червня 2021 р.

Підпис студента-дипломника \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_