Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Бершавського Івана Олександровича*
(ПІБ)

академічної групи *121-18ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js.*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18ск-1* (група) *Бершавського Івана Олександровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js.*

затверджена наказом ректора НТУ «ДП» від №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ (підпис) *доц. Приходченко С.Д.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Бершавський І.О.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 89 с., 28 рис., 3 дод., 23 джерела.

Об'єкт розробки: веб-орієнтоване програмне забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js.

Мета кваліфікаційної роботи: розробка веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js.

У вступі виконується аналіз сучасного стану проблеми, уточнюється постановка завдання, мета кваліфікаційної роботи та галузь її застосування, обґрунтовується актуальність теми.

У першому розділі проводиться дослідження предметної області та існуючих рішень, визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукту, проводиться підрахунок вартості роботи по створенню застосунку та розраховується час на його створення.

Практичне значення полягає у розробці інформаційної системи, що дозволяє користувачам знаходити необхідний товар в одному місці з багатьох магазинів, порівнювати його характеристики відносно інших товарів, відслідковувати динаміку зміни ціни на протязі певного періоду часу, та обирати найвигіднішу пропозицію, яку пропонують інтернет магазини.

Актуальність програмного продукту визначається великим ростом популярності електронної комерції, а особливо сфери продаж, та тим, що сьогодні багато людей намагаються заощаджити час та кошти і данна система допомагатиме користувачам в цьому.

Список ключових слів: ВЕБ, ВЕБ-ДОДАТОК, ПРОГРАМА, МАРКЕТПЛЕЙС, АНАЛІЗАТОР ЦІН, БРАУЗЕР, КЛІЄНТ, ІНФОРМАЦІЙНА СИСТЕМА.

ABSTRACT

Explanatory note: 89 p., 28 figs., 3 appx., 23 sources.

Object of development: web-based software of the Market Dynamics Analyzer platform for monitoring price dynamics using the Vue.js framework.

Purpose of the qualification work: Development of web-based software of the Market Dynamics Analyzer platform for monitoring price dynamics using the Vue.js framework.

In the introduction it is analyzed the current state of the problem, clarified the problem, the purpose of the qualification work and the scope of its application, substantiated the relevance of the topic.

In the first section the research of the subject area and existing decisions is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed.

In the second section the platform for development is chosen, the program design and its development is carried out, the description of algorithm and structure of functioning of system is given, input and output data are defined, characteristics of structure of parameters of technical means are given, work of the program is described.

In the economic section it is determined the complexity of the developed software product, calculated the cost of work to create an application and calculated the time to create it.

Of practical importance is the development of an information system that allows users to find the desired product in one place from many stores, compare its characteristics with other products, track the dynamics of price changes over time, and choose the best offer offered, which is offered by online stores.

The relevance of the software product is determined by the growing popularity of e-commerce, and especially sales, and the fact that many people today are trying to save time and money and this system will help users in this.

Keywords: WEB, WEB-APPLICATION, PROGRAM, MARKETPLACE, PRICE ANALYZER, BROWSER, CLIENT, INFORMATION SYSTEM.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ЕК – електронна комерція;

БД – база даних;

JSON – JavaScript Object Notation, текстовий формат обміну даними

ІС – інформаційна система;

CSS – Cascading Style Sheets ;

ОС – операційна система;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології.

ВСТУП

Поняття електронної комерції у багатьох асоціюється з інтернет-магазином. Хоча на сьогоднішній день електронна комерція пішла далеко за межі звичайних продаж, та все ж, продажі є найбільш поширеним видом електронної комерції.

Після початку пандемії, переваги торгівлі через Інтернет стали очевидними навіть для підприємців, які до останнього не вірили в переваги, або не хотіли переводити свій бізнес в онлайн з якихось інших причин. На сьогоднішній день через Інтернет можна купити все - від білета до автомобіля.

В Україні ЕК зараз знаходиться на стадії активного розвитку - кожного дня, ринок поповнюється новими гравцями, так як перейти в сферу Інтернет-торгівлі досить легко. Кількість покупців в інтернет-магазинах теж постійно збільшується, в першу чергу, за рахунок постійного росту кількості користувачів мережі Інтернет, а також за рахунок збільшення посередників в торгівлі, які теж переходять на закупівлі в інтернеті.

Перевагами інтернет магазинів є можливість придбати товар у зручний для покупця час, уникаючи затрат часу на стояння в чергах та пошуку необхідного товару на безмежних полицях. Крім того, часто в інтернет магазинах представлені залишені іншими покупцями покупцями відгуки, що подекуди більш інформативніше чим живе спілкування з консультантом. Також в інтернет-магазині можна замовити товар, якого просто немає в роздрібній торгівлі.

Насправді, не все так однозначно. Сьогодні багато людей які здійснюють купівлі в інтернеті теж стикаються з певними проблемами, так само як і магазини, особливо ті, які є новачками в цьому напрямі. З ростом кількості інтернет магазинів збільшується кількість результатів видачі в пошукових сервісах. Багато людей ніколи не переглядають результати видачі далі першої сторінки. Враховуючи шо при цьому до половини результатів видачі на

головній сторінці складає реклама, важко говорити про стовідсоткову впевненість, що ви обрали найкращу пропозицію серед наявних.

Так само і магазинам важко боротись за можливість того, щоб його магазин був відвідуваним, не кажучи про продажі. Багато магазинів пропонують якісні та відносно дешевші послуги ніж більш відомі аналоги, і все одно програють їм в боротьбі за клієнтів. Саме тому було вирішено створити інформаційну систему, яка представлятиме собою веб-застосунок для користувачів, які шукають більш вигідні пропозиції на ринку незалежно від популярності того чи іншого магазину на просторі інтернету, бажають заощадити час на перегляді безлічі пропозицій в інтернеті та кошти на обранні найвигіднішої пропозиції в даний момент часу. Також даний продукт допомагатиме просувати свої послуги та товари маленьким магазинам безкоштовно, за рахунок живої конкуренції.

Тому завданням кваліфікаційної роботи було обрано “Розробку веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js”.Завдання даної кваліфікаційної роботи та об'єкт його діяльності безпосередньо пов'язані за напрямом підготовки «Інженерія програмного забезпечення» та відповідає узагальненій тематиці кваліфікаційних робіт.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

E-commerce - це галузь яка розвивається досить швидко, саме тому бізнесу в Інтернеті приділяється особлива увага в зв'язку зі становленням інформаційного суспільства та його безпосереднім внеском в економіку. Сама по собі глобальна мережа дає можливість товарам і послугам виходити на світовий ринок торгівлі. Розвиток сектора інформаційно-комунікативних технологій також є чинником, що сприяє успішному веденню бізнесу і поштовхом до підвищення темпів економічного зростання країн. Електронна комерція активно розвивається протягом двох останніх десятиліть, що в порівнянні з багатьма галузями економіки є невеликим проміжком часу.

Одним з основних видів електронної комерції є B2C (Business-to-Consumer). Ця сфера (бізнес для споживача) передбачає торгівлю товарами і послугами між юридичними і фізичними особами. Це свого роду роздрібні продажі, але тільки за допомогою онлайн-майданчиків - магазини, сервіси, банки та інше. Перевага клієнтів в більшому асортименті вибору, зручність замовлення та доставки товарів додому або в офіс.

Інтернет-магазин дозволяє користувачам онлайн, в своєму браузері, сформувати замовлення на покупку, вибрати спосіб оплати і доставки замовлення, оплатити замовлення, а також здійснює доставку товару.

Інтернет-магазин може бути як основою створення бізнесу, так і допоміжним інструментом, націленим на розвиток і розширення вже наявного бізнесу, тим самим сприяти додатковому залученню клієнтів і капіталу.

Електронна комерція надає величезні можливості для бізнесу, щоб заробити на хвилі зростаючого попиту з боку споживачів, які цікавляться покупками в Інтернеті.

Електронна комерція, як і будь-яка ніша на ринку, має свої плюси і мінуси.

Переваги:

− зниження затрат - електронна торгівля спрощує бізнес-процеси в багатьох галузях підприємництва. Наприклад, щоб відкрити онлайн-магазин не потрібно орендувати фізичну площу, наймати штат продавців і співробітників доставки. В результаті транзакційні витрати нижче, що позначається на вартості продукції або послуг.

− розширення цільової аудиторії - через Інтернет можна продавати по всьому світу без особливих витрат. Звичайно, якщо бізнес пов'язаний з фізичними товарами, то доведеться поламати голову з доставкою. Просте рішення - це співпраця з транспортними компаніями.

− менше посередників - електронна комерція дозволяє працювати безпосередньо з виробником, виключаючи ланцюжок посередників. Так створюється прямий канал між продавцем і покупцем, що позначається на вартості товарів і якості обслуговування.

− можливість зі 100% точністю аналізувати продажу, просування, розвиток бізнесу в мережі. Системи аналітики, коллтрекінга дозволяють стежити за ситуацією і своєчасно вживати заходів.

Проблеми, які можуть виникати в ЕК:

− залежність від інформаційно-комунікаційних технологій.

− особливості законодавства, податки - відсутність правового регулювання онлайн-комерції часто служить перешкодою при укладанні тих чи інших угод.

− безпека інформації - онлайн-торгівля і бізнес в мережі вимагає високої гарантії конфіденційності даних користувачів, покупців, учасників комерційної діяльності. Активно впроваджується сертифікація, авторизація, капча і інші варіанти боротьби з шахрайством.

− захист прав власності - це давно не нова проблема для мережі Інтернет. Піратські копії програмного забезпечення, «злиті» у вільний доступ майстер-класи, книги і інша продукція інтелектуальної праці - все це стає проблемою для електронної комерції у всьому світі.

Рік тому компаніям в Україні прогнозували приріст e-commerce в 2020 році на рівні 15%, проте пандемія істотно скоригувала цей прогноз. Через інтернет почали торгувати багато нових гравців - від великих виробників до кав'ярень. Великі онлайн і офлайн-магазини і маркетплейси в більшій мірі посилили свої позиції. Тепер майже 9% всіх покупок в Україні відбувається онлайн - на маркетплейсах, в онлайн-магазинах і соціальних мережах. До Китаю, де більше 30% всіх покупок в онлайні, ще далеко, але приріст за рік чималий. Для порівняння, в 2019-му частка e-commerce в ритейлі України оцінювалася в 7%, а виріс ринок тоді на 17% за рік.

За рік кількість замовлень на маркетплейсах EVO - Prom.ua, Bigl.ua, Crafta.ua, Shafa.ua, IZI.ua і сайтах компаній Prom.ua зросла на 42%. Середній чек впав на 10%. Це повязано з тим, що люди стали купувати онлайн значно частіше і дешевші товари - одяг, товари повсякденного вжитку, їжу, маски.

Значно більшим попитом, ніж в минулому році, користувалися товари для садівництва і городництва - тут купували все: від насіння до міні-комбайнів. Також в топ найпопулярніших товарів увірвалися зоотовари - корм та аксесуари для домашніх тварин, а також продукти харчування. Також не дивно те, що найбільший приріст показали медичні товари (маски, санітайзери, пульсометри), а саме +225%. На початку пандемії склалася така ситуація, що деякі товари взагалі знайти можна було лише в Інтернеті.

1.2. Призначення розробки та галузь застосування

Хоча ЕК досить кардинально змінює підходи бізнесу, вона поки що досить погано вирішує деякі важливі для покупця проблеми.

Основною проблемою для більшості людей є обрання необхідного продукту серед величезної кількості магазинів. Враховуючи те, що за статистикою люди більше дивляться саме на співвідношення ціни та якості товару, для того, щоб вибрати необхідну річ необхідно оглянути велику кількість магазинів, в яких ціна може відрізнятися досить суттєво.

Також досить важливою проблемою є те, що ціна може відрізнятися не тільки між магазинами а й в межах одного магазину в різний проміжок часу в залежності від різних чинників (сезонність, дефіцит, спекуляції та ін.).

Також можна враховувати що на онлайн закупівлі переходять не тільки прості люди, а й різні інші бізнеси (магазини роздрібної торгівлі, кафе, офісні працівники). Для таких користувачів які проводять закупки було б зручно обрати необхідні товари та цілі категорії до відслідковування, для зручності та без необхідності щоразу шукати однакові речі.

Для вирішення подібних проблем існує декілька аналогів на ринку України, проте жоден з них не вирішує перераховані проблеми повністю.

Метою даного дипломного проекту є реалізація системи, яка у разі створення буде допомагати заощаджувати як час так і кошти, при пошуку необхідних товарів, що покращить життя людей

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

− освітня програма 121 «Інженерія програмного забезпечення»;

− навчальний план та графік навчального процесу;

− наказ ректора Національного технічного університету «Дніпровська політехніка» № 317 від 07.06.2021 р;

− завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js.

1.4. Постановка завдання

Завданням даної роботи є створення клієнтської частини веб-додатку для моніторингу динаміки цін, з використанням фреймворку Vue.js.

В результаті необхідно спроектувати та розробити клієнтську частину веб-додатку для автоматичного і спрощеного порівняння цін та їх динаміки між магазинами, для спрощення пошуку найбільш вигідної пропозиції.

Кінцевий продукт матиме два рівні доступу до функціоналу:

- рівень доступу «Гість»,
- рівень доступу «Клієнт».

Розглянутий веб-додаток буде створений на мультипарадигменній мові програмування JavaScript, так як альтернативи для створення клієнтської частини їй відсутні.

Для пришвидшення роботи було обрано фреймворк Vue.js, який підходить як для невеликих проектів, яким необхідно додати трохи реактивності так і для великих односторінкових додатків, завдяки своїм основним компонентам, таким як Router і Vuex.

Vue підходить для розробки рішень, які використовують зовнішні API для обробки даних.

Так як у нашому випадку для передачі даних з серверу клієнту буде використано REST API – підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів, Vue добра підходить для даної задачі.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт матиме два рівні доступу до функціоналу:

- рівень доступу «Гість»,
- рівень доступу «Клієнт».

На рівні доступу Гість повинен містити наступні функції:

- реєстрація та логін;
- перегляд категорій;
- перегляд товарів;
- перегляд динаміки цін на товар.

На рівні доступу Клієнт повинен містити наступні функції:

- вихід з аккаунту;
- додати/видалити категорію до обраних, для подальшої швидкого пошуку;
- перегляд обраних категорій, які будуть описані на вкладці в профілі користувача;
- додати/видалити товар з обраних;
- перегляд обраних товарів;
- порівняння цін між магазинами;
- редагування даних користувача.

Веб-застосунок повинен мати зручний та зрозумілий інтерфейс, надавати необхідну користувачам інформацію в зручному вигляді. Мати можливість порівняння товарів для неавторизованих користувачів, відображати історію переглянутих товарів у вигляді каруселі.

1.5.2. Вимоги до інформаційної безпеки

Основні вимоги до інформаційної безпеки:

- конфіденційність інформації;
- цілісність даних;
- доступність інформації;
- вірогідність даних;
- автентичність даних;
- впровадження політики прав доступу.

Під час роботи додатку є два режими:

- авторизований користувач;
- неавторизований користувач.

Неавторизований користувач має змогу переглядати товари, слідкувати за динамікою цін.

Авторизований користувач має змогу обирати товар для відслідковування, обирати цікаві продукти для їх порівняння.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для роботи з веб-додатком використовується браузер. Відповідні характеристики необхідні для роботи більшості браузерів для Windows:

- ЦП: Pentium 4;
- відеоадаптер: 3D адаптер nVidia, Intel, AMD/ATI;
- відеопам’ять: 128 МБ;
- накопичувач: 150 Гб;
- оперативна пам’ять: 2 Гб.

У випадку з мобільними платформами необхідні мінімальні версії операційних систем:

- iOS 11.0 і вище;
- Android 4.4 і вище.

1.5.4. Вимоги до інформаційної та програмної сумісності

Веб-додаток – клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. Це дозволяє вирішити багато проблем, так як немає необхідності хвилюватись про окрему платформу, так як браузер має можливість працювати на персональних комп’ютерах, ноутбуках, смартфонах.

Сучасні браузери мають функцію автоматичного оновлення, тому проблеми у користувачів не повинні виникати при роботі в популярних

браузерах.

Для підтримки були обрані такі браузери:

- Chrome від 87 версії;
- Opera від 74 версії;
- Edge від 88 версії;
- Safari від 13.1 версії;
- IE від 11 версії;
- Android від 90 версії;
- Baidu від 7.12 версії;
- Firefox від 78 версії;
- iOS Safari від 13.4версії;
- Opera mobile від 62 версії;
- Samsung від 13.0 версії.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом виконання кваліфікаційної роботи повинен бути додаток сумісний для роботи в різних середовищах, за рахунок того, що додаток буде запускатися в сучасних браузерах і не залежатиме від операційної системи та від того десктоп це чи мобільний пристрій.

Основне призначення додатку - це порівняння ціни товарів відносно магазина та дати у зручному графічному представленні.

Додаток міститиме два рівні доступу до функціоналу:

- рівень доступу «Гість»
- рівень доступу «Клієнт»

На рівні доступу Гість повинен містити наступні функції:

- реєстрація та логін;
- перегляд категорій;
- перегляд товарів;
- перегляд динаміки цін на товар.
- порівняння товарів у межах категорії.

На рівні доступу Клієнт повинен містити наступні функції:

- додавання/видалення категорії до обраних, для подальшої швидкого пошуку;
- перегляд обраних категорій, які будуть описані в профілі користувача;
- додати/видалити товар з обраних;
- перегляд обраних товарів;
- порівняння цін між магазинами;
- редагування даних користувача.

2.2. Опис застосованих математичних методів

Під час проектування та розробки даної інформаційної системи використовувалися лише прості арифметичні дії. Математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Під час проектування ІС, для обміну даними між сервером та клієнтом, було вирішено обрати архітектурний стиль REST.

REST (від англ. Representational State Transfer) - так називається спосіб взаємодії та обміну даними сервера, своєрідна оболонка взаємодії компонентів розподіленого додатку в мережі. REST є узгоджений набір обмежень, що враховуються при проектуванні розподіленої системи. У певних випадках (інтернет-магазини, пошукові системи, інші системи, засновані на даних) це призводить до підвищення продуктивності і спрощення архітектури. У широкому сенсі компоненти в REST взаємодіють на зразок взаємодії клієнтів і серверів. Єдиного стандарту у неї немає, тому, що це не протокол, а цілий архітектурний стиль. Цим вона відрізняється від багатьох аналогічних.

Більшість великих компаній розробляють для внутрішнього використання або для своїх клієнтів цей інтерфейс, адже він здатний забезпечити спілкування між двома системами і надавати зручну для обробки інформацію.

В мережі Інтернет виклик віддаленої процедури може являти собою звичайний HTTP-запит (зазвичай GET або POST; такий запит називають «REST-запит»), а необхідні дані передаються в якості параметрів запиту.

Для веб-служб, побудованих з урахуванням REST (тобто ті, що не порушують накладених їм обмежень), застосовують термін «RESTful». Більшість RESTful-реалізацій використовують такі стандарти, як HTTP, URL, JSON і рідше, XML.

Переваги:

- компоненти систем взаємодіють в набагато більшому масштабі;
- всі інтерфейси загальні;
- частини можна впроваджувати незалежно одна від одної;
- є проміжні елементи, які знижують відсоток затримки і посилюють безпеку з'єднання.

Недоліками архітектури можуть бути:

- надлишкова вибірка - коли запит на кінцеву точку API надає інформацію, яка не вимагається клієнтами, або ними не використовується.
- недостатня вибірка - ситуація, коли клієнт отримує кілька запитів, щоб отримати всю інформацію, яка йому потрібна.

Клієнтську частину веб-додатку (FrontEnd) створено за шаблоном MVVM (рис. 2.1). Шаблон MVVM (Модель - Вид - ViewModel - «модель - представлення - модель представлення») - це наступний етап розвитку MVC. Його особливість - це модель презентацій (ViewModel, VM) та зв'язок даних.

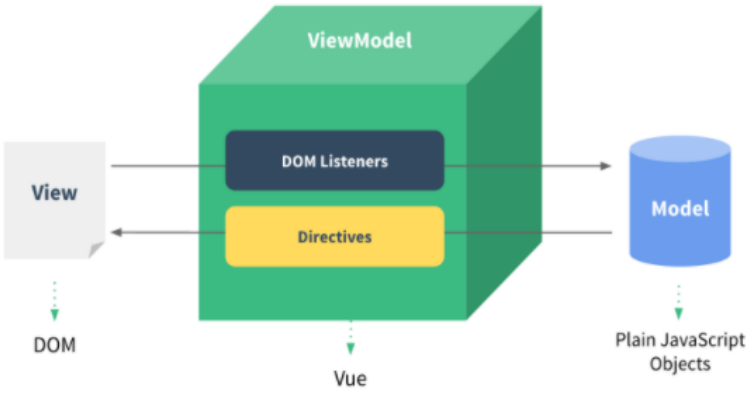

Рис 2.1. – Графічне представлення шаблону MVVM

MVVM забезпечує основу для побудови клієнтських програм, які здійснюють взаємодію з користувачем та зворотній зв'язок з більш інтерактивними, уникаючи при цьому затратного дублювання коду в рамках всієї архітектури.

Технічно Vue.js орієнтований на шар ViewModel шаблону MVVM. Філософією Vue є надання переваги реактивній прив'язці даних та компонентів представлення за допомогою якомога простішого API. Він з'єднує Вигляд і Модель за допомогою двосторонньої прив'язки даних. Фактичні маніпуляції з DOM та форматування виводу абстрагуються в Директиви та Фільтри.

2.4. Опис використаних технологій та мов програмування

Дана інформаційна система була розроблена з використанням таких технологій:

- Vue 3;
- Vue Router;
- Vuex;
- Vue cli;
- JavaScript;
- Node.js;
- Axios;
- Chart JS;
- SCSS.

Vue - це фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue придатний для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня представлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами.

Vue.JS реалізує всі сучасні підходи до розробки web UI і є легким в освоєнні, гнучким і високо інтегрованим.

Даний фреймворк повністю підходить для створення складних односторінкових додатків (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Під час створення додатку на Vue можна використовувати лише знання JavaScript та HTML, що робить його надзвичайно гнучким, та дозволяє розробнику налаштувати структуру додатків відповідно до власних вимог.

У Vue.js реалізується шаблон MVVM, Vue.js пропонує можливість прив'язки даних на Javascript, так що виведення та введення даних передаються безпосередньо з джерел даних. Таким чином, режим ручного визначення даних (наприклад, через jQuery) з HTML-DOM не потрібен. При цьому немає необхідності в будь-яких додаткових анотаціях.

Vue не можна назвати повномасштабним фреймворком - він розроблений, щоб бути простим і гнучким шаром перегляду. Його можна використовувати окремо для швидкого прототипування або поєднувати його з іншими бібліотеками для власного стеку побудови інтерфейсу.

API Vue.js знаходиться під сильним впливом AngularJS, KnockoutJS, Ractive.js та Rivets.js. Незважаючи на подібність, Vue.js пропонує цінну альтернативу цим існуючим бібліотекам.

Vue Router - офіційна бібліотека маршрутизації для Vue.js. Вона глибоко інтегрується з Vue.js і дозволяє легко створювати SPA-додатки. Включає можливості зручного відображення вкладених маршрутів, створення та налаштування модульної конфігурації маршрутизатора проекту, представляє зручний доступ до параметрів маршруту, (query, wildcards), створення програмної навігації в проекті. Надає автоматичне встановлення класів для посилань за активним маршрутом, роботи і історією переглядів.

Всі можливості глибоко інтегруються в ядро Vue, що надає можливість працювати з маршрутизацією з будь-якої точки програми.

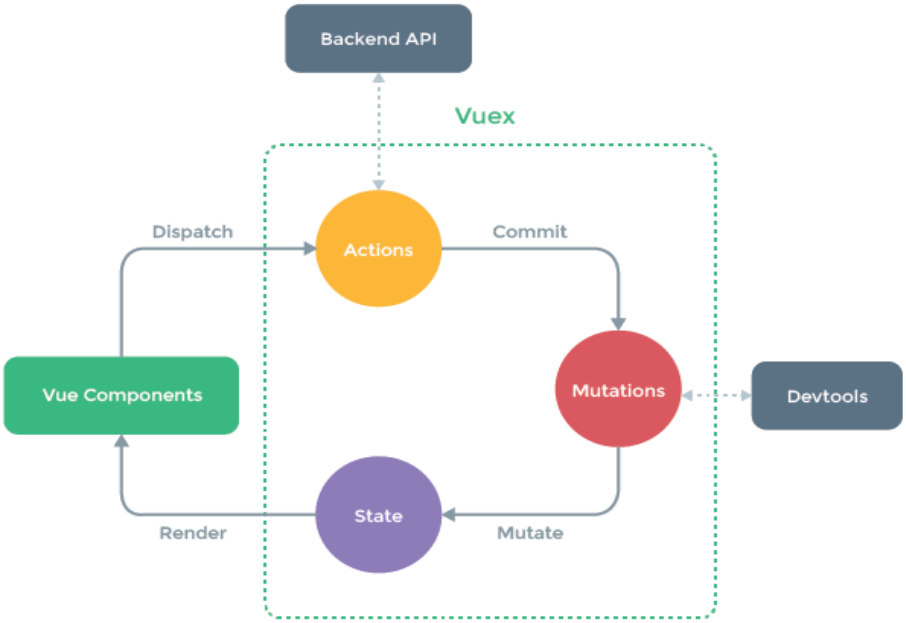

Рис. 2.2. – Схема роботи Vuex

Vuex - патерн управління станом та бібліотека для додатків на Vue.js. Він служить централізованим сховищем даних для всіх компонентів програми з правилами, що гарантують, що стан може бути змінено тільки передбачуваним чином (рис. 2.2.). Vuex інтегрується з офіційним розширенням vue-devtools (opens new window), надаючи «з коробки» такі просунуті можливості, як «машину часу» для налагодження і експорт / імпорт зліпків стану даних.

Vuex допомагає керувати спільним станом компонентів ціною принесення нових концепцій і допоміжного коду. Компроміс, коли короткочасна продуктивність страждає на благо довгостроковій.

Два моменти відрізняють сховище Vuex від простого глобального об'єкта:

а) Сховище Vuex реактивно. Коли компоненти Vue покладаються на його стан, то вони будуть реактивно і ефективно оновлюватися, якщо стан сховища змінюється.

б) Не можна безпосередньо змінювати стан сховища. Єдиний спосіб внести зміни - явно викликати мутацію. Це гарантує, що будь-яка зміна стану залишає слід і дозволяє використовувати інструментарій, щоб краще розуміти хід роботи програми.

JavaScript - мультипарадигмена мова програмування. Підтримує об'єктно-орієнтований, імперативний і функціональний стилі програмування.

Понад 97% веб-сайтів використовують її на стороні клієнта для поведінки веб-сторінок, часто включаючи сторонні бібліотеки. Усі основні веб-браузери мають спеціальний механізм JavaScript для виконання коду на пристрої користувача.

У листопаді 1996 р. компанія Netscape представила JavaScript Ecma International як вихідну точку для стандартних специфікацій, яким можуть відповідати всі розробники браузерів. Це призвело до офіційного випуску першої специфікації мови ECMAScript у червні 1997 року.

Процес стандартизації тривав кілька років, із випуском ECMAScript 2 у червні 1998 р. Та ECMAScript 3 у грудні 1999 р. Робота над ECMAScript 4 розпочалась у 2000 р.

Тим часом Microsoft завойовувала дедалі домінуючі позиції на ринку браузерів. На початку 2000-х ринкова частка Internet Explorer досягла 95%. Це означало, що JavaScript став фактичним стандартом для сценаріїв на стороні клієнта в Інтернеті.

Трохи заплутаним може здаватися через те, що поряд часто вживаються терміни JavaScript та ECMAScript. JavaScript створений як скриптова мова для Netscape. Після чого вона була відправлена в ECMA International для стандартизації (ECMA - це асоціація, діяльність якої здійснює стандартизацію інформаційних та комунікаційних технологій). Це привело до появи нового мовного стандарту, відомого як ECMAScript.

Після оновлення версії JavaScript вже були засновані на стандартах ECMAScript. Таким чином ECMAScript - стандарт, а JavaScript - найпопулярніша реалізація цього стандарту.

Node.js - платформа з відкритим кодом для виконання високо-продуктивних мережевих застосунків, написаних мовою JavaScript.

Модульність Node.js дає можливість створювати невеликі додатки без необхідності підтримки величезної інфраструктури, більша частина

функціоналу якої в результаті виявиться не задіяною. При розробці додатку на Node.js розробник сам може вирішувати, що йому потрібно, і, за необхідності, розширити проект.

Окрім вбудованого функціоналу, який постачає сам Node.js, в його склад входить також менеджер модулів npm (node package manager). Його основна ідея – це створення маленького програмного блоку, який вирішує одну конкретну проблему, але робить це швидко та якісно. Це дає можливість компонувати власні великі проекти з цих маленьких побудованих блоків. До того ж, це задовольняє принцип відкритості та багаторазового використання коду.

У Vue розробники вирішили, що наявність вбудованого клієнтського модуля HTTP непотрібна і може краще обслуговуватися сторонніми бібліотеками.

Axios - це клієнтська бібліотека HTTP. Вона використовує проміси за замовчуванням і працює як на клієнті, так і на сервері, що робить її придатною для отримання даних під час рендерингу на стороні сервера. Оскільки вона використовує проміси, то її можна поєднати з async/await, щоб отримати стислий і простий у використанні API.

JavaScript бібліотека Chart.js дозволяє генерувати на стороні клієнта графіки та діаграми за допомогою засобів HTML5 (Canvas). Бібліотека підтримує створення лінійних графіків, стовпцевих, кругових та радіальних діаграм. Chart.js підтримує використання анімованих ефектів.

Важливою перевагою Chart.js є її невеликий розмір і відсутність яких-небудь залежностей. До того часу ми маємо можливість навіть зменшити розмір бібліотеки за допомогою включення в неї лише тих модулів, які необхідні в конкретному випадку. До прикладу, якщо потрібно створити лише діаграму кругового типу (кругової діаграми), можна підключити ядро бібліотеки Chart.js та модуль за допомогою якого створюються діаграми відповідного типу. Таким чином, буде зменшено загальний розмір бібліотек Chart.js та збільшена швидкість завантаження сторінок у цілому. Іншою

перевагою бібліотеки є адаптивність діаграм, що дозволяє змінювати свій розмір при зміні розмірів вікна браузера.

Sass - це метамова на основі CSS, призначена для збільшення рівня абстракції CSS-коду і спрощення роботи з файлами каскадних таблиць стилів.

Синтаксис цієї мови дуже гнучкий, він пропонує вирішення безлічі проблем, які присутні в звичайному CSS. В даній мові на надається можливість додати певну логіку (@if, each), математику (можна складати як числа, рядки, так і кольори). Важливим також є те що з'являється можливість вкладеного стилізування компонентів, що збільшує швидкість отримання очікуваного результату від заданих стилів, шляхом значного прискорення пошуку усіх стилів, що впливають на певний елемент.

2.5. Опис структури програми та алгоритмів її функціонування

Діаграми варіантів використання (use case diagrams) використовуються для відображення сценаріїв використання системи (use cases) та користувачів системи (actors), які використовують її функції.

Актори на діаграмі варіантів використання позначаються символом людини, а варіанти використання – еліпсом.

У нашому випадку актором може виступати новий або уже зареєстрований у системі користувач (рис. 2.3.).

Незареєстрований користувач має обмежені функції, такі як пошук та перегляд необхідного товару. Також кожен користувач має можливість переглянути графік історії цін товару.

Для того щоб використовувати розширений функціонал, користувачеві необхідно авторизуватися. Відповідно незареєстрованому користувачеві необхідно спочатку зареєструватися.

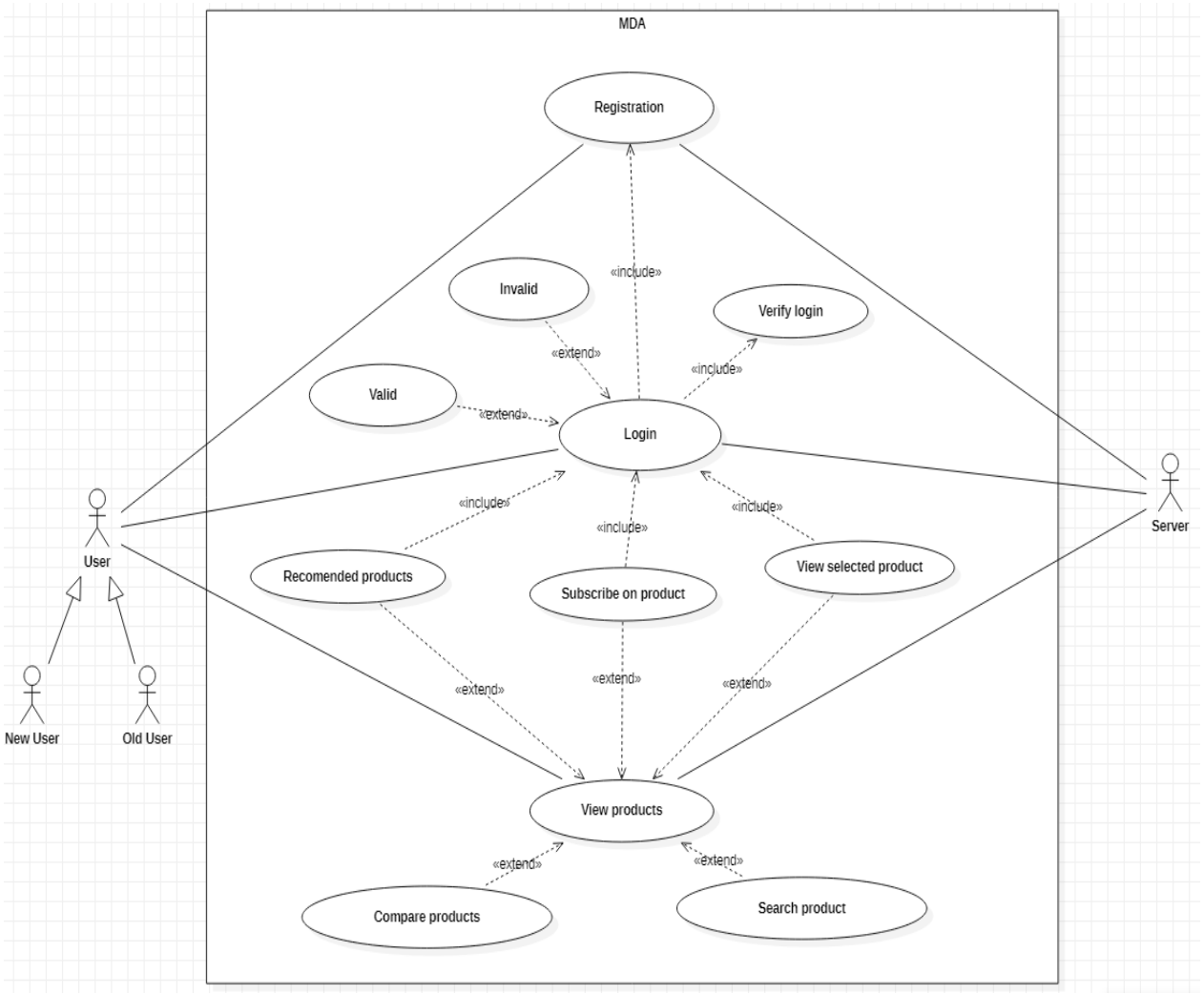

Рис. 2.3. – Діаграма варіантів використання (Use Case).

Після авторизації користувач має можливість в профілі обрати категорії, які цікаві саме йому, для того, щоб додаток мав змогу коригувати свою роботу під користувача.

Після авторизації з’являються можливості:

− підписка/відписка на товар, для того, щоб кожного разу не проводити пошук, так як продукт після підписки буде відображатися в профілі користувача у відповідному розділі;

Діаграми послідовності (Sequence diagram) є видом діаграми взаємодії мови UML, які описують відношення об'єктів у різних умовах. Взаємодія задається сценарієм, отриманим на етапі розробки діаграми варіантів використання.

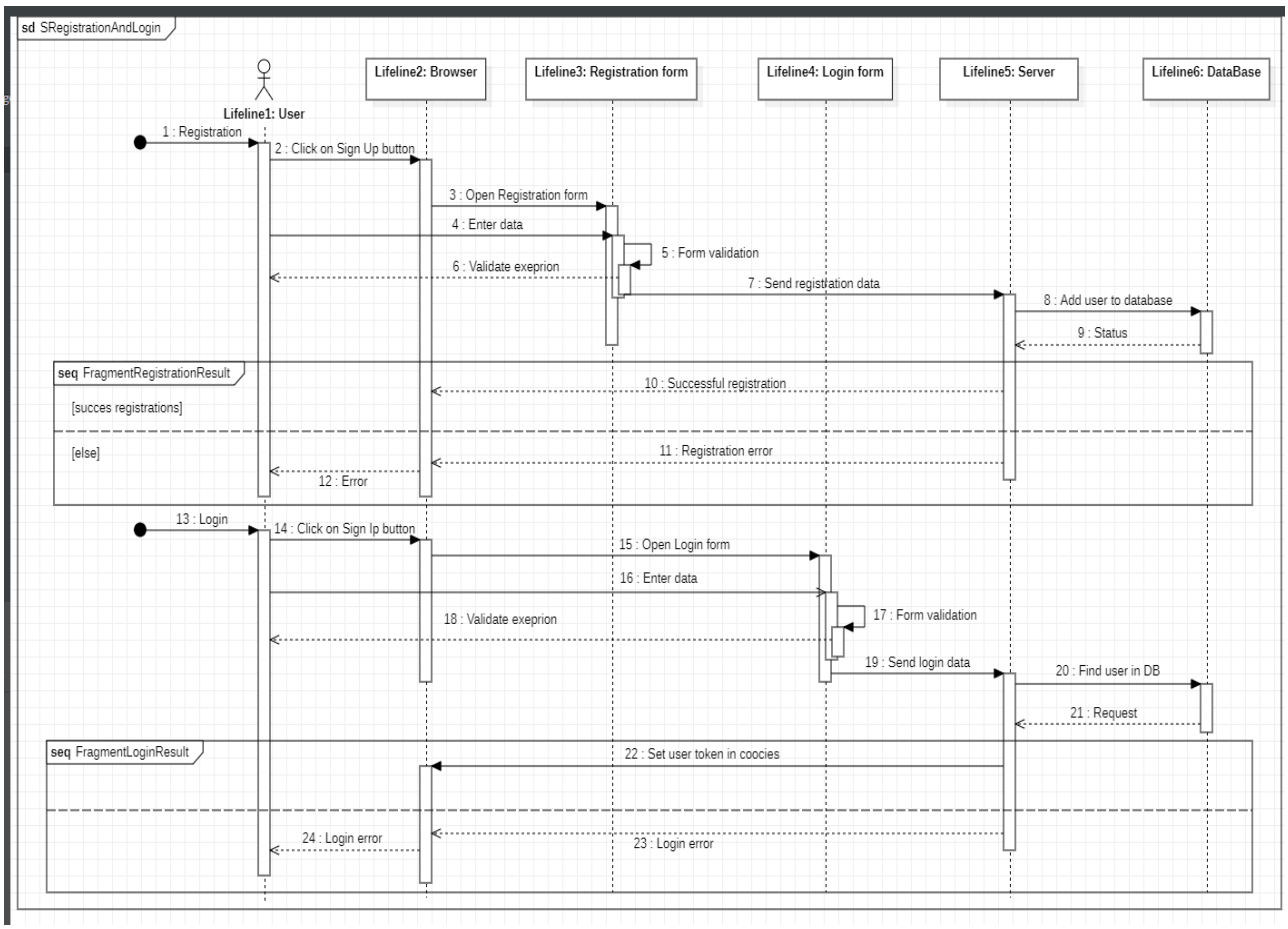

Рис. 2.4. – Діаграма послідовності (реєстрація та авторизація).

На діаграмі відображено послідовність дій які необхідні для реєстрації користувача та його входу в систему (рис. 2.4.).

Для реєстрації користувачеві необхідно відкрити форму реєстрації, заповнити дані необхідні для реєстрації. Після заповнення форми та натиснення кнопки «Реєстрація», почнеться етап валідації. Після невдалої валідації даних користувачеві буде відправлено повідомлення про помилки при заповненні полів і запропоновано виправити помилки. Після вдалої реєстрації користувачеві буде виведено повідомлення.

Для входу в систему користувачеві необхідно відкрити форму логіну, заповнити необхідні дані. Після заповнення форми та натиснення кнопки «Увійти», почнеться етап валідації. Після невдалої валідації данних користувачеві буде відправлено повідомлення про помилки при заповненні полів і запропоновано виправити помилки. Після успішної валідації запит на

вхід буде відправлено на сервер, з серверу буде відправлено результат, повідомлення про відсутність користувача з введеними даними – після невдалого входу, або авторизаціонний токен token – після успішного входу.

На зображенні (рис. 2.5.) відображено діаграму послідовності дій для:

- додавання товару в обрані;
- видалення товару з обраних;

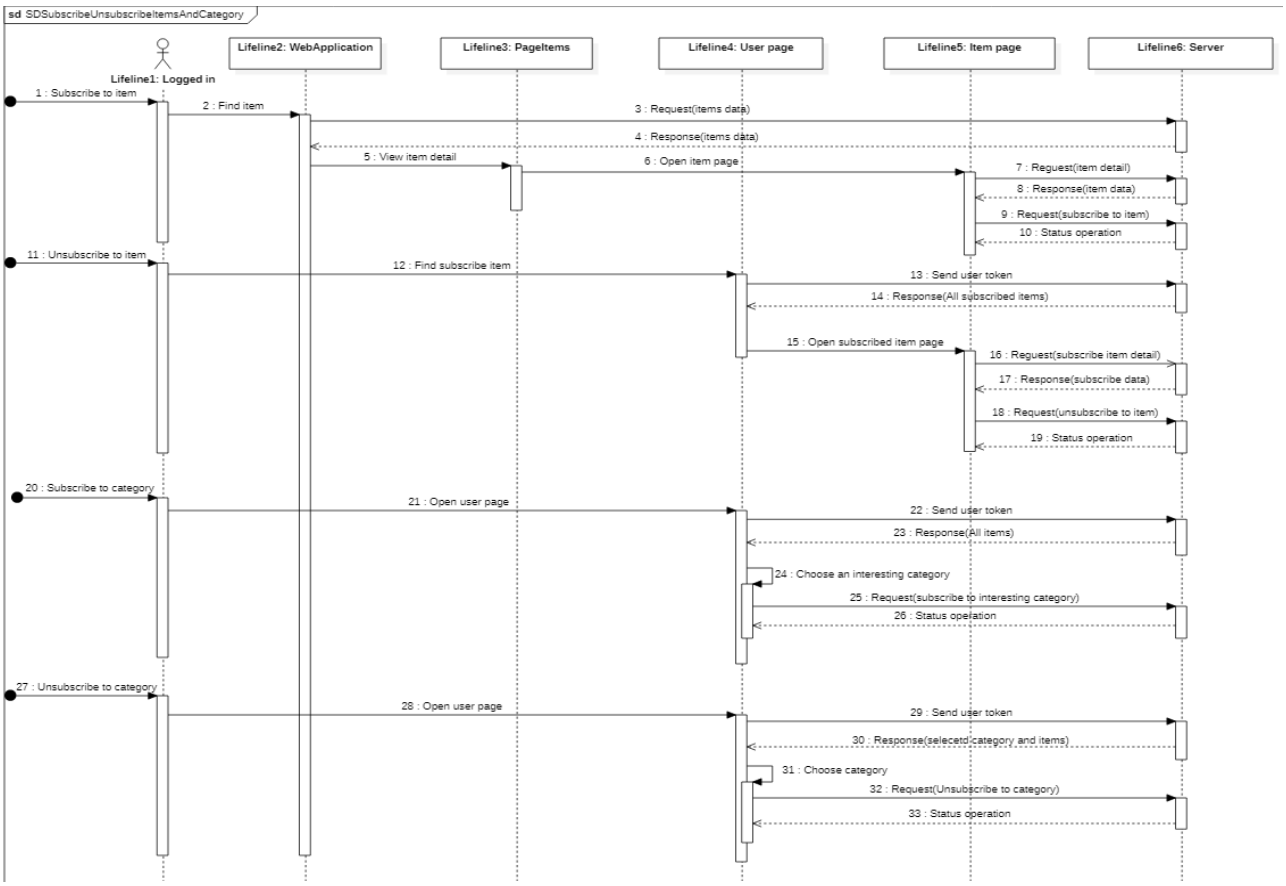

Рис. 2.5. – Діаграма послідовності (додавання/видалення обраних категорій/товарів).

Веб-застосунок створений за компонентним підходом, де кожен елемент може існувати в системі окремо і в різних місцях (рис 2.6.).

Компоненти розвивають ідею плагінів. Кожен з них реалізує якусь свою можливість (а якщо немає існуючих, то можна написати і свій). Якщо знадобиться реалізувати подібне в іншому місці - легко перевикористати плагін

знову. Взаємодію можна описати простим інтерфейсом: відправляємо в плагін вхідні параметри, а для зворотного зв'язку можемо відстежувати події.

Все це справедливо і для компонентів. З тією лише відмінністю, що компонент може являти собою не тільки одну річ, але і якусь частину програми, яка повинна працювати і виглядати всюди однаково (наприклад, форма коментування, з аватарки, редактором і красивим select'ом).

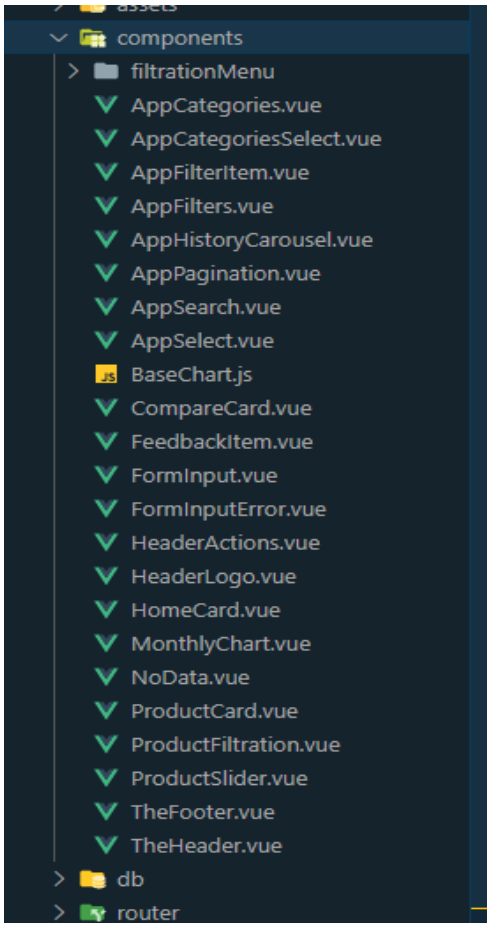

Рис. 2.6. – Каталог в якому знаходяться компоненти.

Загальна структура каталогу проекту (рис. 2.7.) складається з таких основних розділів:

– .vscode – каталог в якому зберігаються налаштування редактору, для того щоб незалежно від комп’ютера, у програмістів які працюють над проектом були встановлені необхідні налаштування за замовчуванням, так як налаштування які зберігаються в проекті мають пріоритет над стандартними;

− node_modules – директорія де зберігаються npm модулі, які використовуються під час створення проекту;

− public – директорія для html документу;

− src – директорія, де зберігаються основні створюваного елементи веб-застосунку, такі як директорія з компонентами, директорія з файлами, що описують роботу Vuex та Vue Router (store та router відповідно) та директорія views, в якій описані основні представлення інтерфейсу.

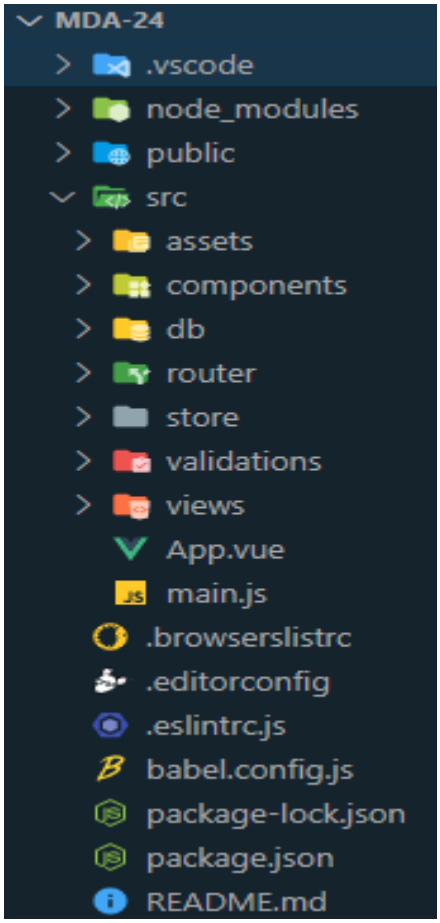

Рис. 2.7. – Структура каталогу проекту

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними інформаційної системи можуть бути різні дані для пошуку та фільтрації товару, які представляються користувачеві для вибору. До

таких даних відносяться код категорії, код товару, та коди характеристик для фільтрації товарів.

Вихідними даними є представлення даних які зберігаються у базі даних, у зручному для графічного представлення вигляді, які, наприклад, дозволяють побудувати графік динаміки цін, або заповнити таблицю порівняння товарів за їх характеристиками.

Обмін даних між сервером та клієнтським за стосунком відбувається у форматі JSON. Так як такий тип даних зручний як для обробки на сервері, так і для роботи на клієнті.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для користувача є важливим мати систему, яка може запускатися та працювати з сучасними браузерами, тож необхідно мати такі мінімальні параметри ЕОМ які рекомендують розробники більшості браузерів:

- ЦП: Pentium 4.
- Відеоадаптер: 3D адаптер nVidia, Intel, AMD/ATI.
- Відеопам’ять: 128 МБ.
- Накопичувач: 150 Гб.
- Оперативна пам’ять: 2 ГБ.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

- VS Code ;
- Node.js
- Git, GitHub

Visual Studio Code - це легкий, проте потужний редактор коду, який доступний для популярних операційних систем таких як Windows, macOS та Linux. Він постачається із вбудованою підтримкою JavaScript, TypeScript та Node.js і має багату екосистему розширень дял цих мов та середовищ виконання.

Підтримує виділення синтаксису та автозаповнення за допомогою IntelliSense, який забезпечує розумні доповнення на основі типів та змінних, визначень функцій та імпортованих модулів.

Node.js — це платформа, призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Платформа широко використовується для створення серверних програм.

В Node.js використовується розроблений компанією Google двигун V8. Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в не блокуючому режимі та визначенні обробників зворотніх викликів (callback).

Git - це популярна система управління версіями з розподіленою архітектурою. На відміну від інших систем на кшталт CVS і Subversion (SVN), де повна історія версій проекту доступна лише в одному місці, в Git кожна робоча копія коду сама по собі є репозиторієм. Це дозволяє всім розробникам зберігати історію змін в повному обсязі.

Розробка в Git орієнтована на забезпечення високої продуктивності, безпеки і гнучкості розподіленої системи. Також Git показує високу продуктивність в порівнянні альтернативами. Це можливо завдяки оптимізації процедур фіксації коммітов, створення гілок, злиття і порівняння попередніх версій. Алгоритми Git розроблені з урахуванням глибокого знання атрибутів, характерних для реальних дерев файлів вихідного коду, а також типовою динаміки їх змін і послідовностей доступу.

GitHub — один з найбільших ресурсів, що пропонує можливості віддаленого зберігання репозиторіїв, для спільної розробки програмного забезпечення та отримувати доступ до сумісної роботи над ними.

2.7.3. Виклик та завантаження програми

Розроблений веб-застосунок завантажується після переходу по URL - стандартизованій адресі за якою додаток буде доступний в мережі інтернет, за допомогою одного з сучасних браузерів, на усіх популярних операційних системах. Прикладом на основних десктопних ОС, таких як Windows 10, Linux, Mac OS, є такі браузери, як Google Chrome, Firefox, Brave. Для мобільних платформ підтримуються браузери Google Chrome, Dolphin, Opera Mobile, Mozilla Firefox, Safari.

Додаток гарантовано запускається на останніх двох версіях усіх браузерів які офіційно підтримуються розробниками, на практиці додаток працює і на старіших версіях та гарантій для їх успішної роботи немає.

2.7.4. Опис інтерфейсу користувача

Після запуску веб-додатку користувач потрапляє на головну сторінку, яка програмно має назву Home (рис. 2.8). Вона головною в ієрархії компонентів і в ній будуть з’являтися інші компоненти залежно від типу взаємодії користувача з інтерфейсом додатку. Такими компонентами взаємодії в більшості своїй є кнопки, форми, поля вводу та чекбокси. Інтерфейс є адаптивним до екранів як звичайних моніторів так і мобільних пристроїв.

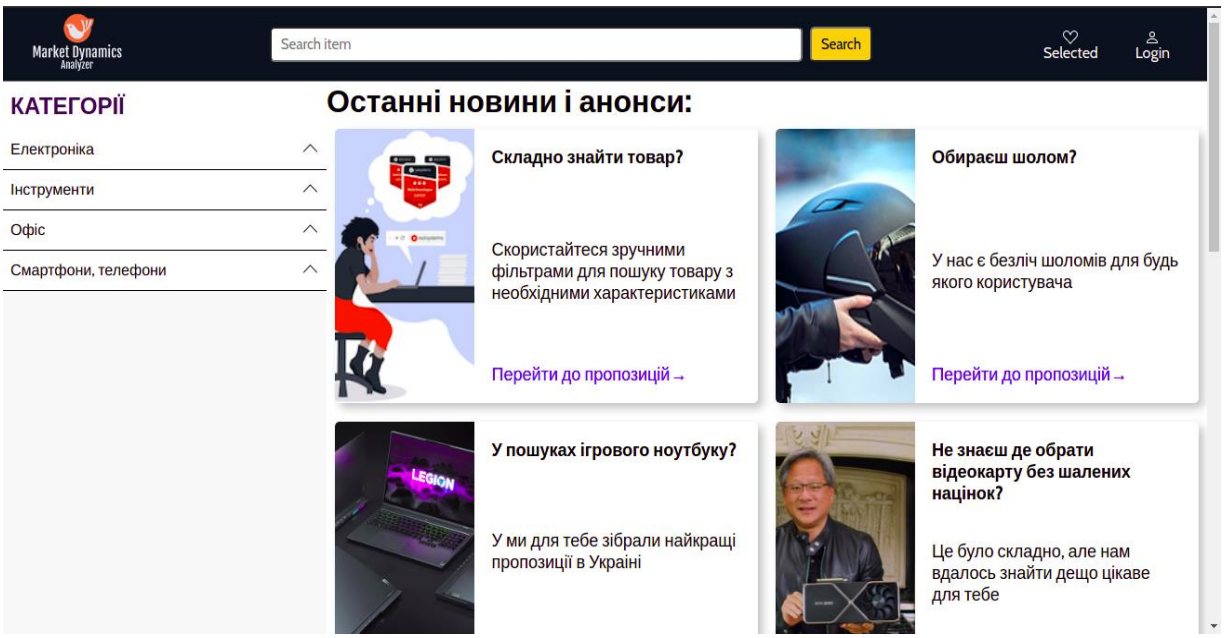

Рис. 2.8. - Головне вікно додатку

В головному меню користувач може ввести назву товару, для пошуку, перейти до форми авторизації, або відразу вибрати необхідну категорію товара, для переходу на сторінку (рис. 2.9.), де будуть відображені товари за обраною категорією (рис. 2.12.).

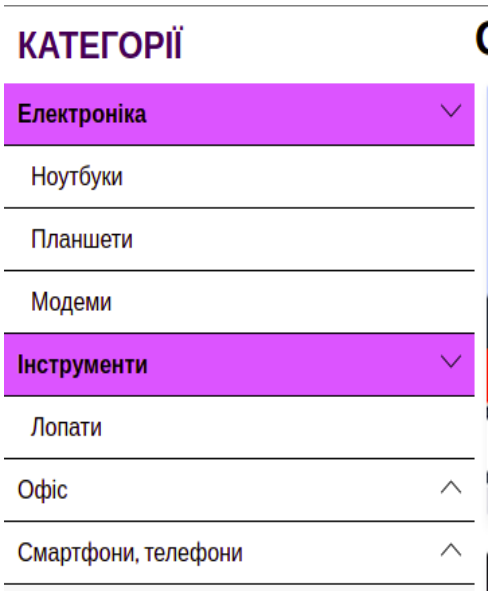

Рис. 2.9. - Меню категорій

На головній сторінці також відображається історія переглянутих товарів(рис. 2.10).

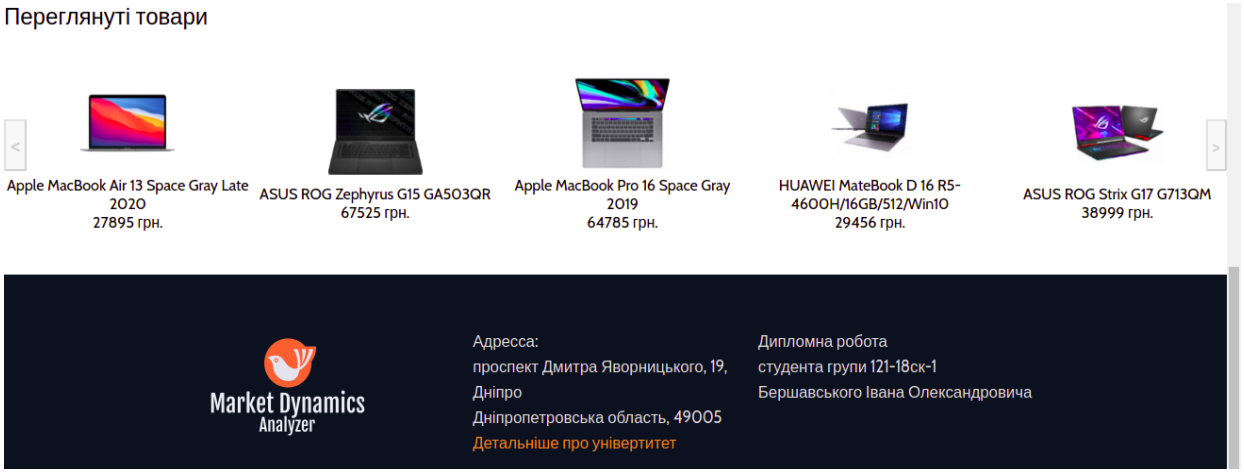

Рис. 2.10. - Історія переглянутих та футер сторінки

Головна сторінка адаптується під розмір екрану (рис. 2.11.)

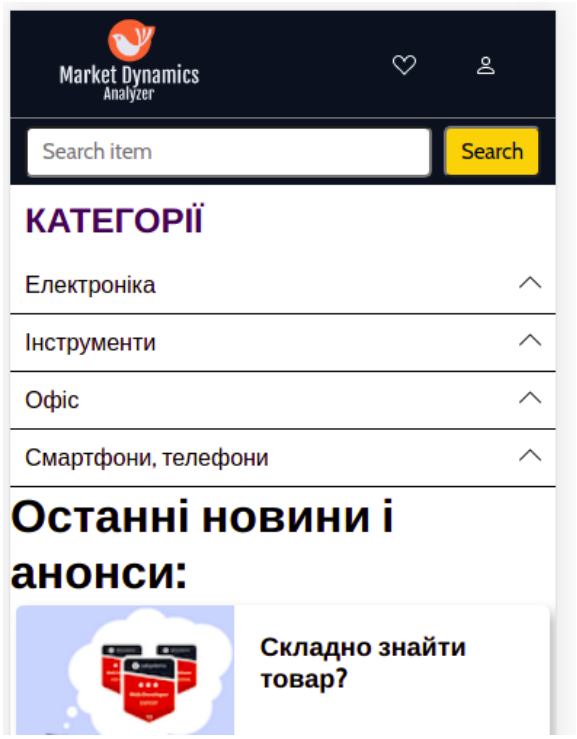

Рис. 2.11 - Головна сторінка на iPhone 6/7/8

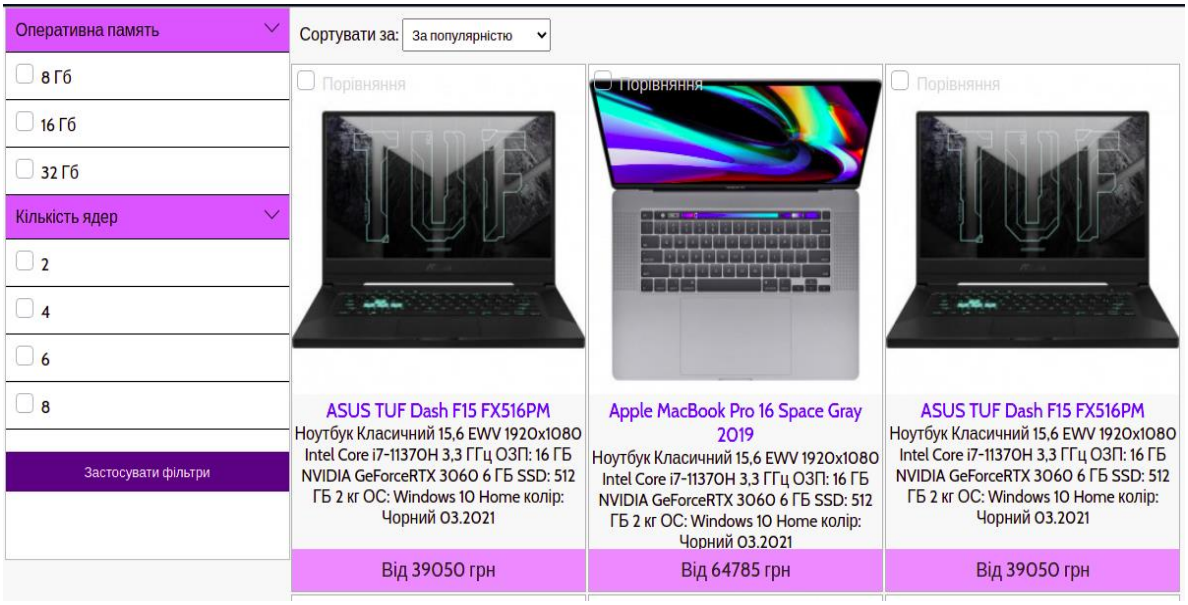

Рис. 2.12. - Вікно перегляду товарів обраної категорії

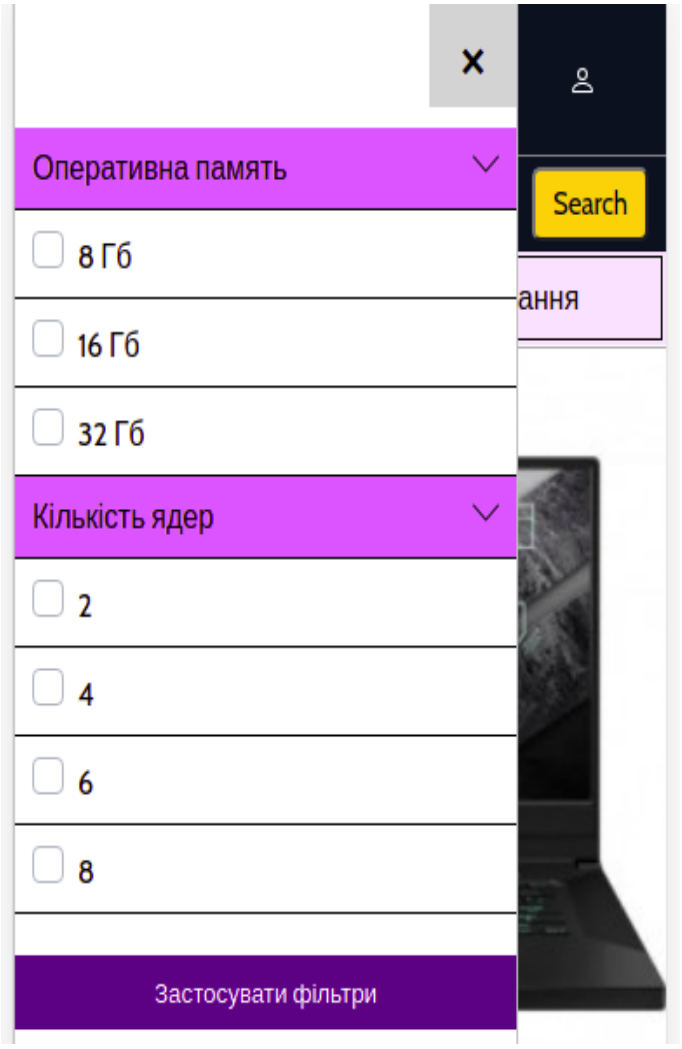

Рис. 2.13. - Фільтрація товарів на iPhone 6/7/8

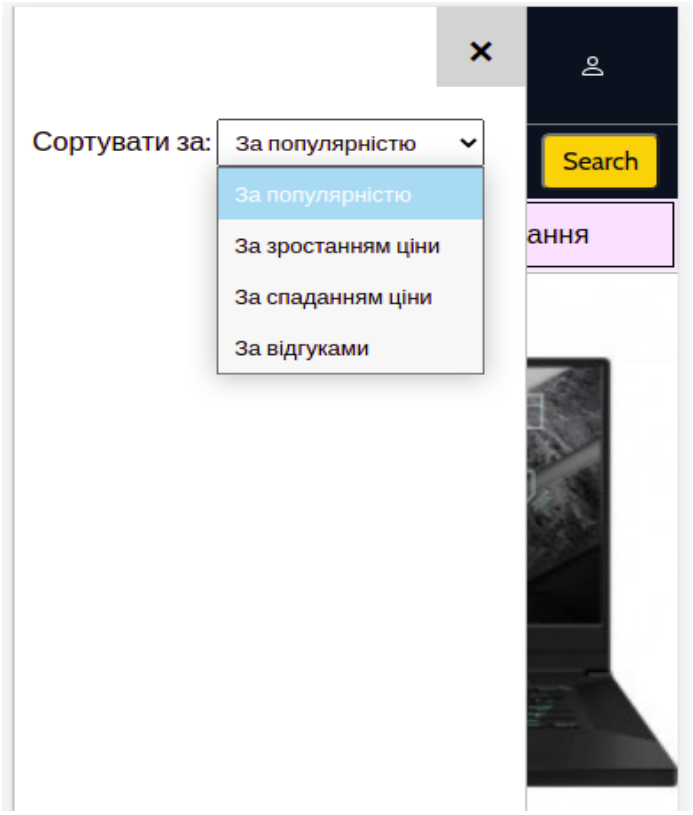

Рис. 2.13. - Фільтрація товарів на iPhone 6/7/8

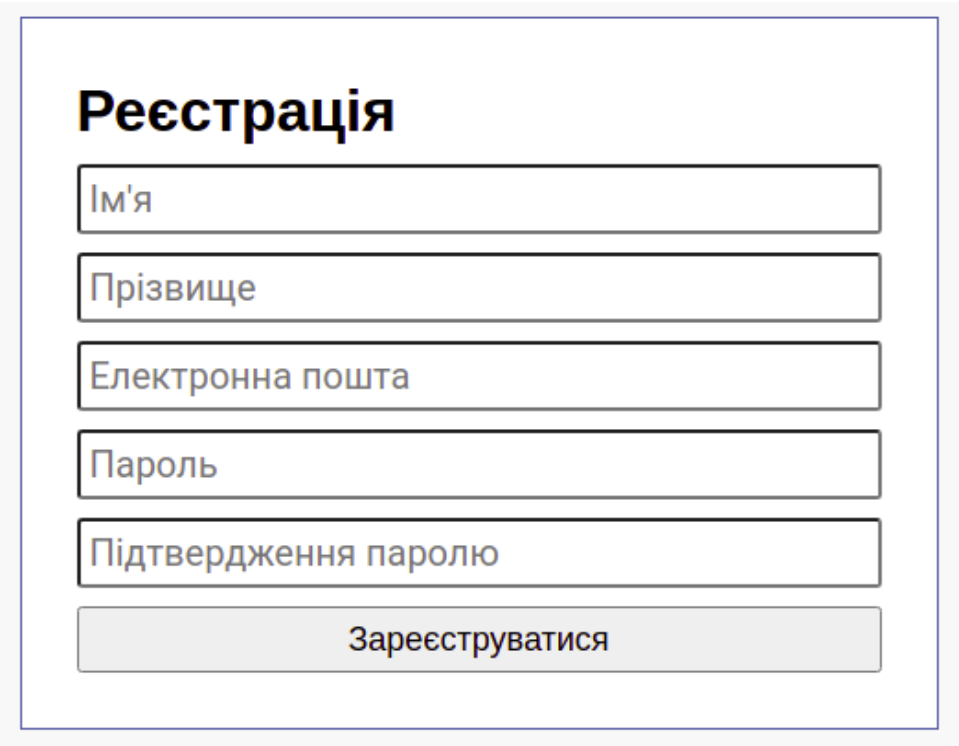

Рис. 2.14. - Форма реєстрації

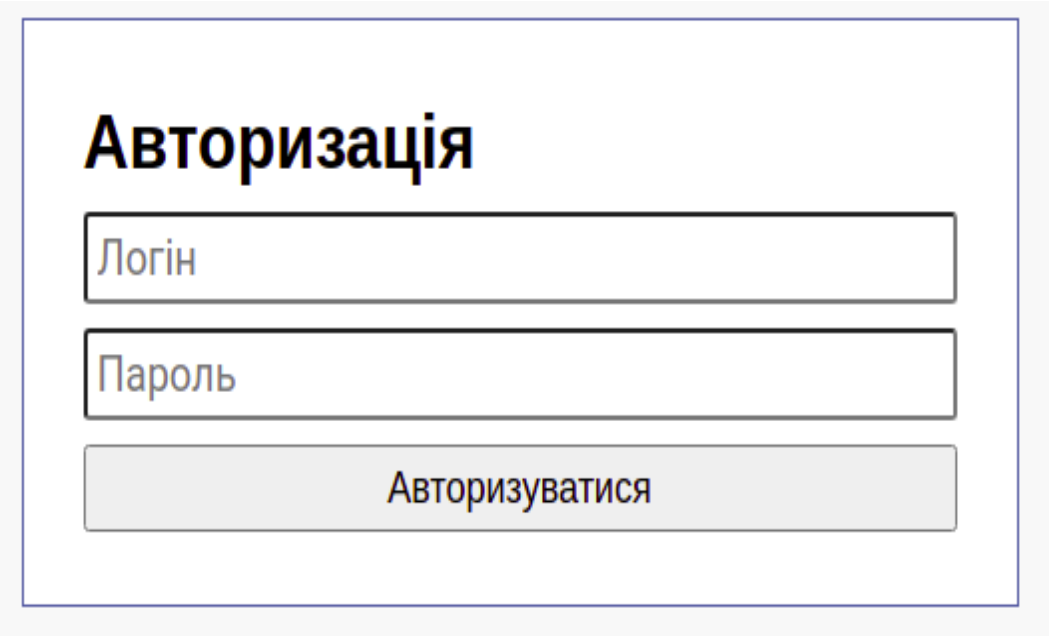

Рис. 2.15. - Форма входу в систему

Для навігації між сторінками є перемикач сторінок (рис. 2.16.).

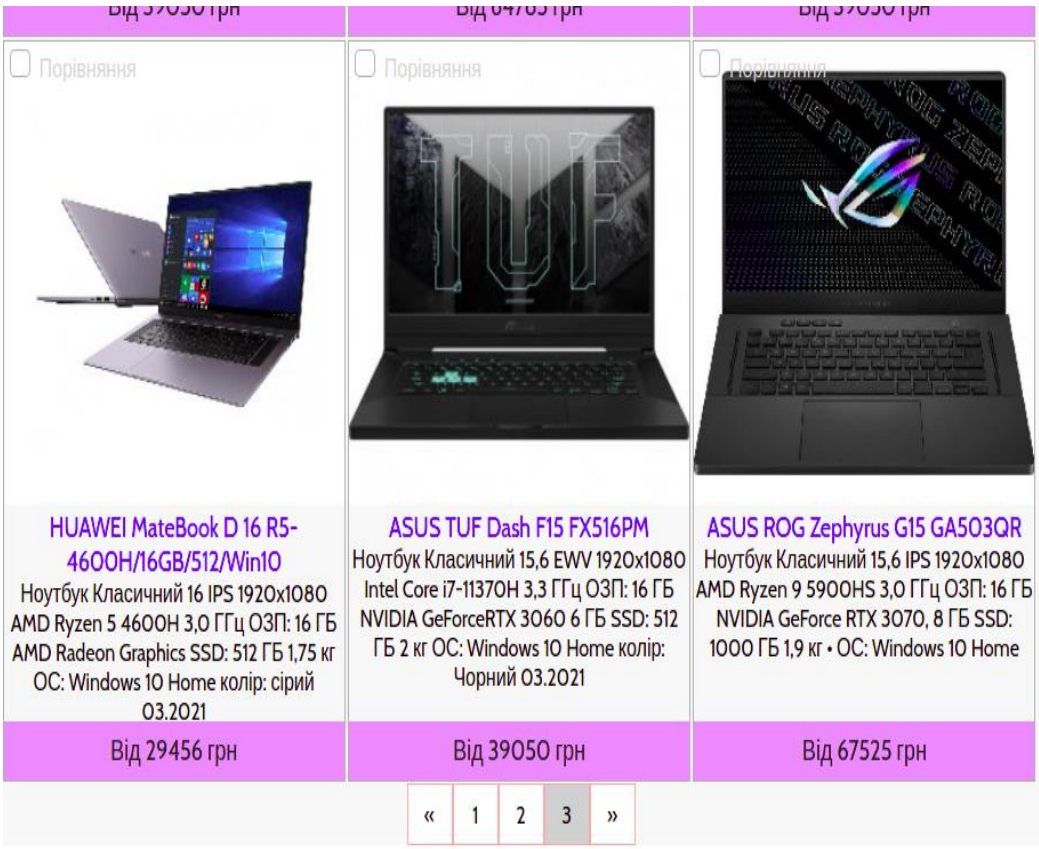

Рис. 2.16. - Перемикання сторінок

Якщо користувач вибере продукт, його перенаправить на сторінку, на якій буде представлено детальну інформацію (рис. 2.17), яка може бути необхідна.

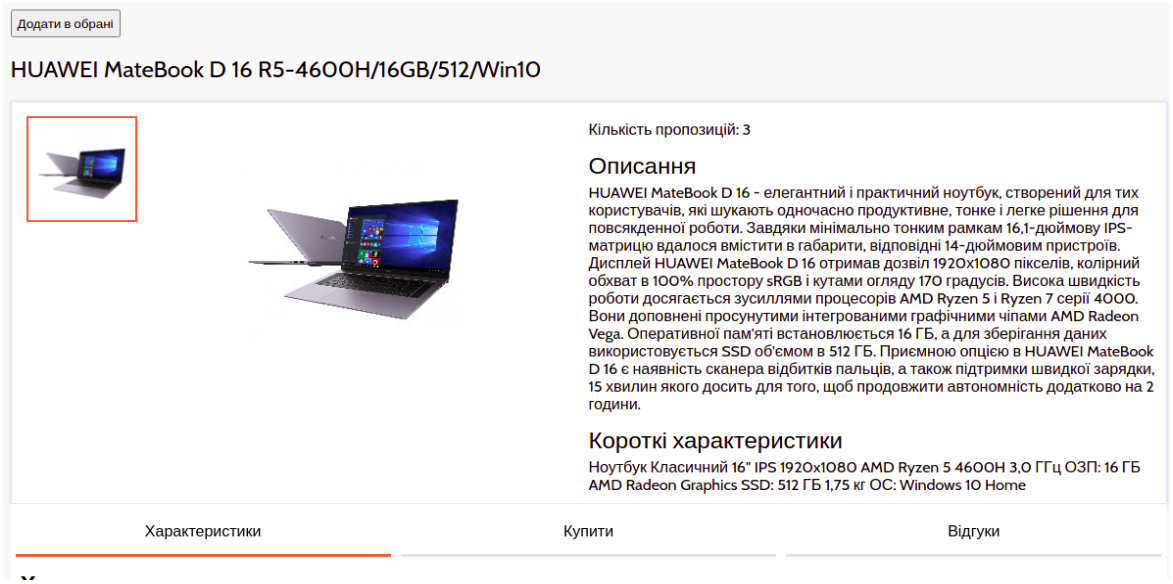

Рис. 2.17. - Сторінка товару з детальним описом

Рис. 2.18. - Сторінка товару з детальним описом на мобільних пристроях

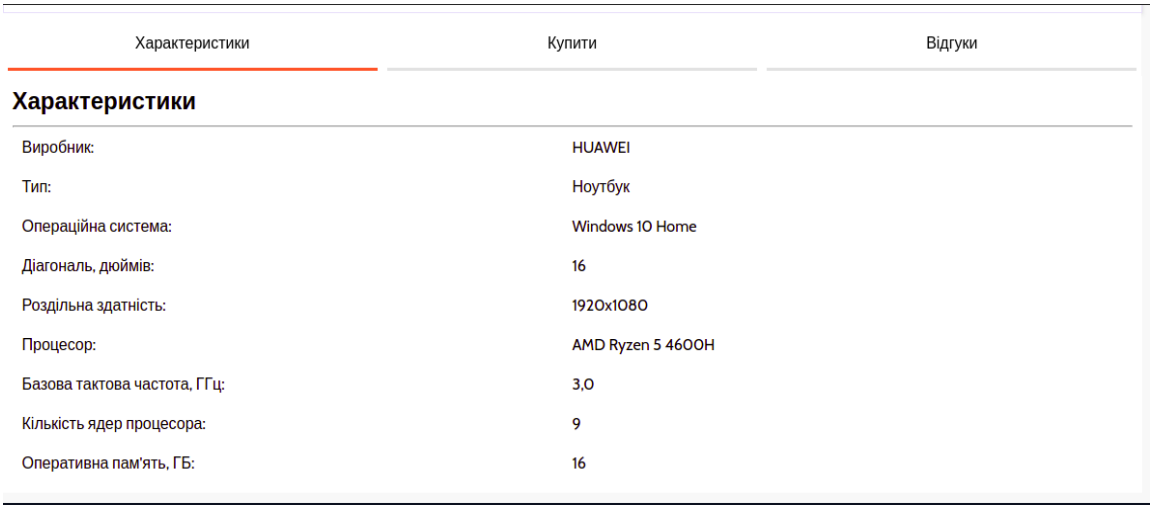

Рис. 2.19. - Сторінка продукту, характеристики

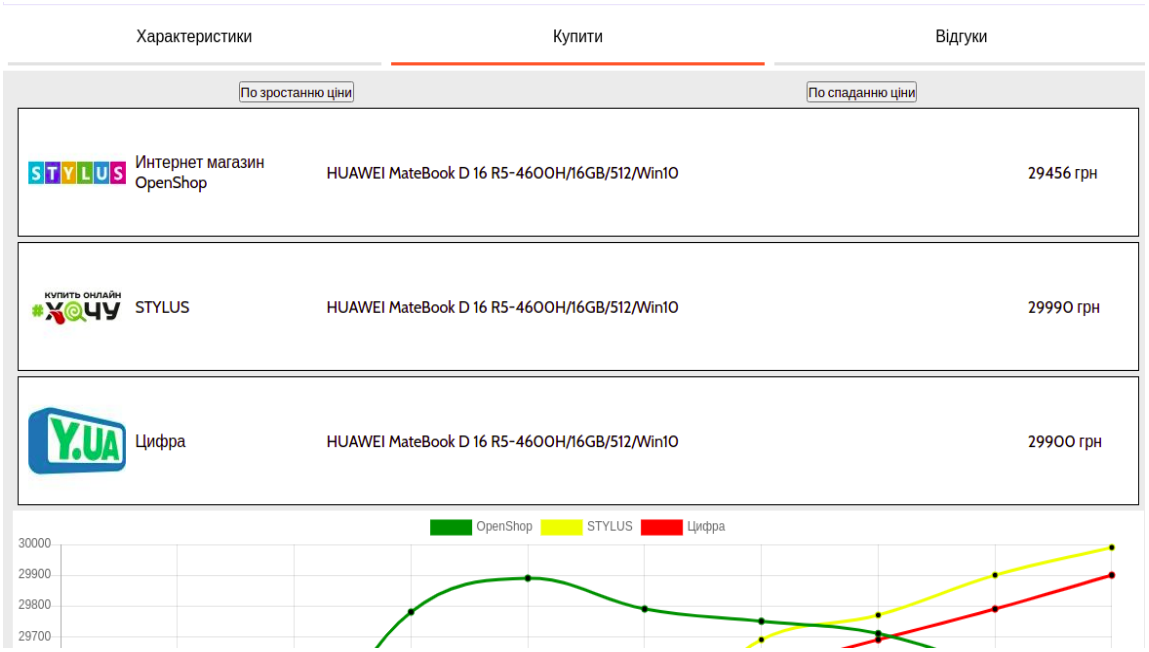

Рис. 2.20. - Сторінка товару, інформація про продавців

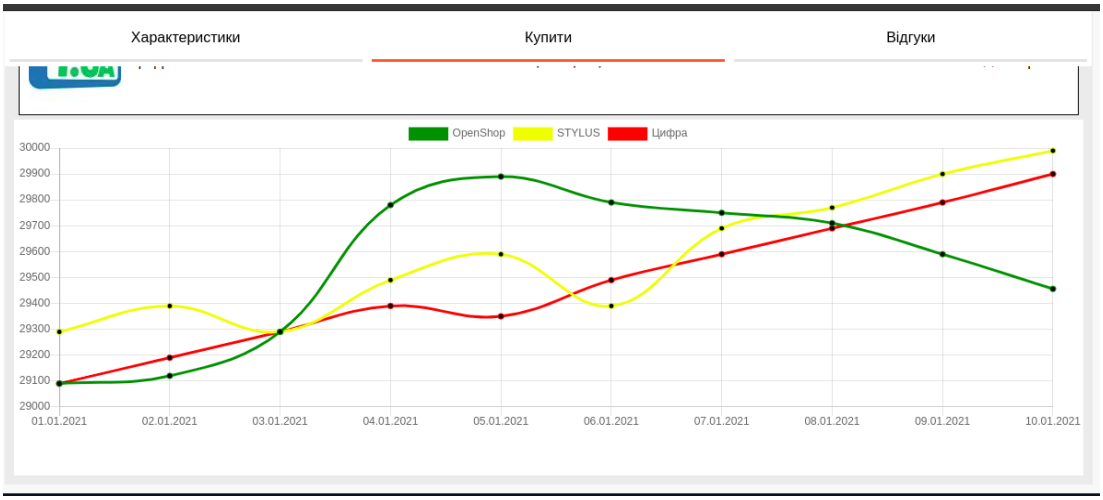

Рис. 2.21. - Сторінка товару, графік динаміки цін по магазинам

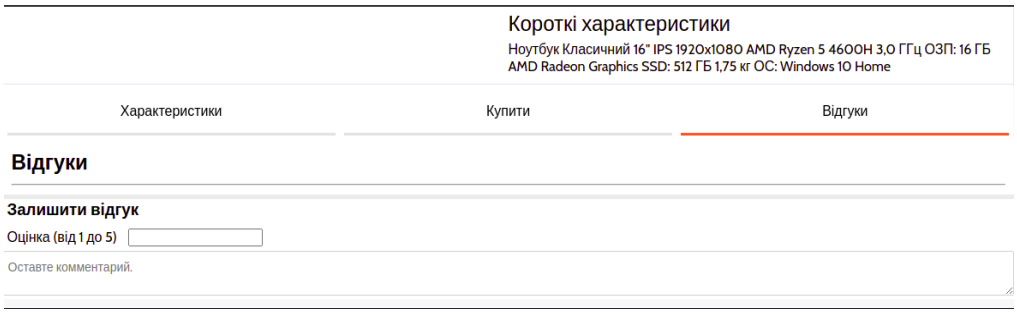

Рис. 2.22. - Сторінка товару, відгуки та запитання

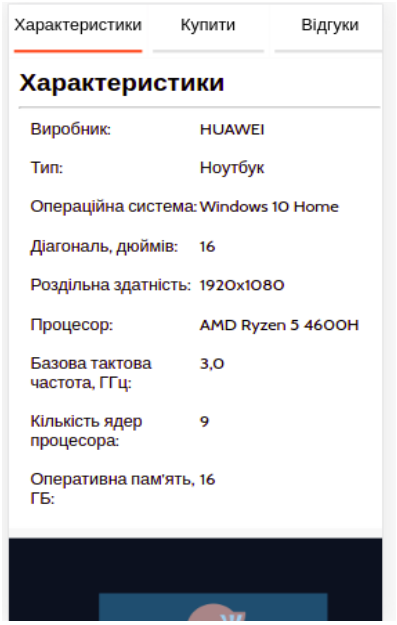

Рис. 2.23. - Сторінка товару, характеристики на мобільних пристрях

Рис. 2.24. - Сторінка товару, інформація про магазини на мобільних пристроях

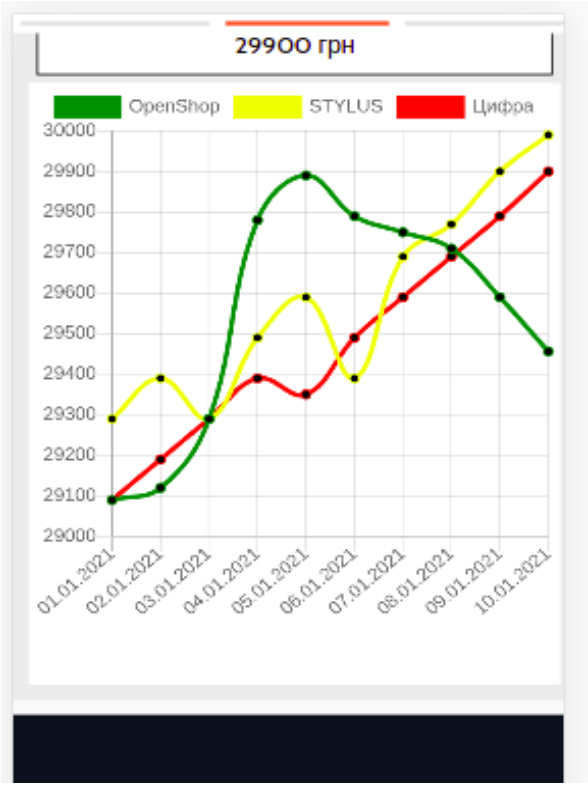

Рис. 2.25. - Сторінка товару, графік динаміки цін на мобільних пристроях

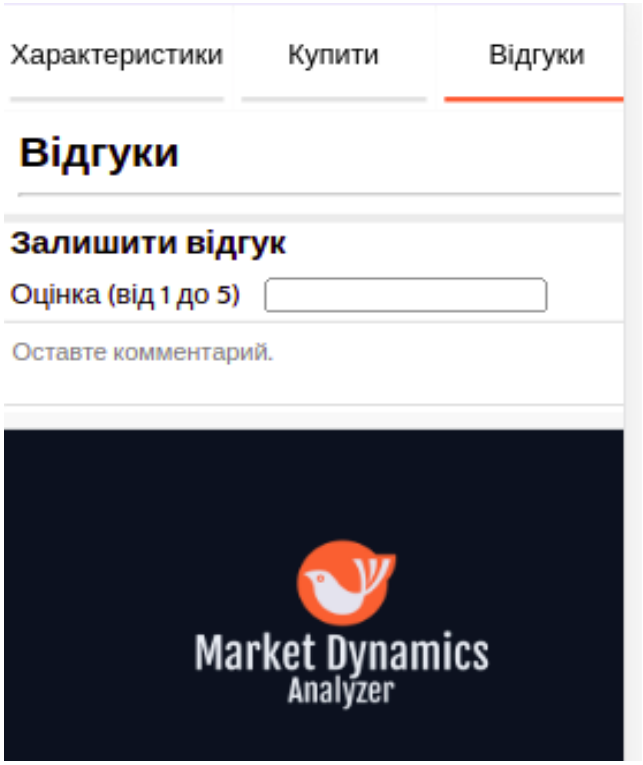

Рис. 2.26. - Сторінка товару, відгуки та питання, вигляд на мобільних пристроях

У профілі користувача можуть відображатися збережуні товари та порівняння товарів по катерогіям.

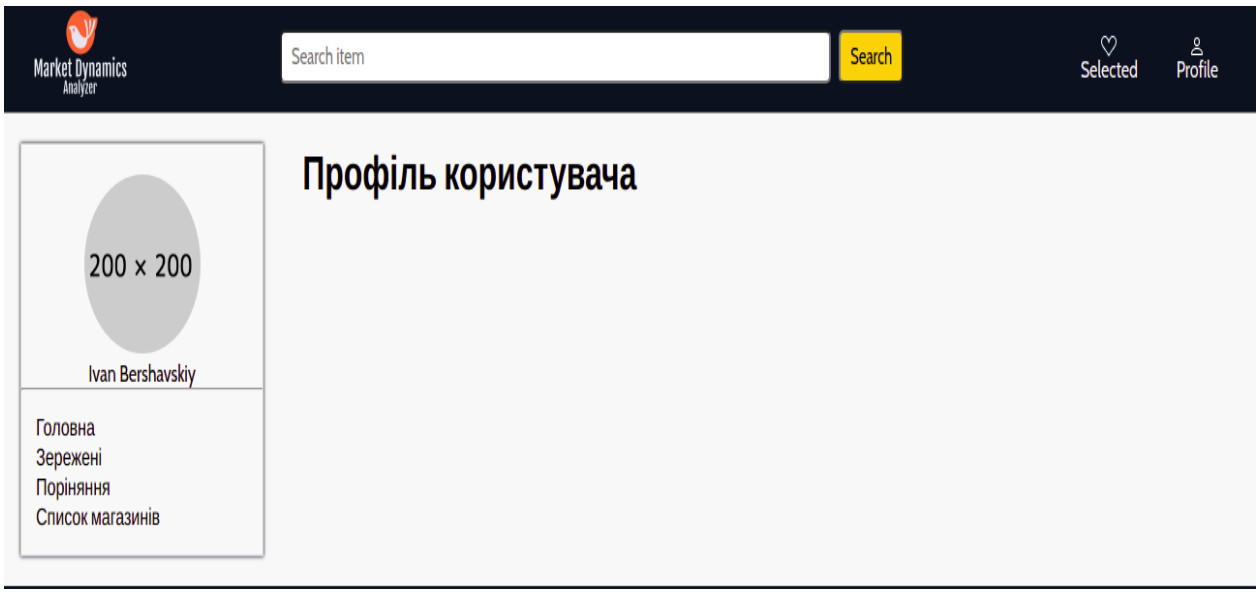

Рис. 2.27. - Профіль користувача

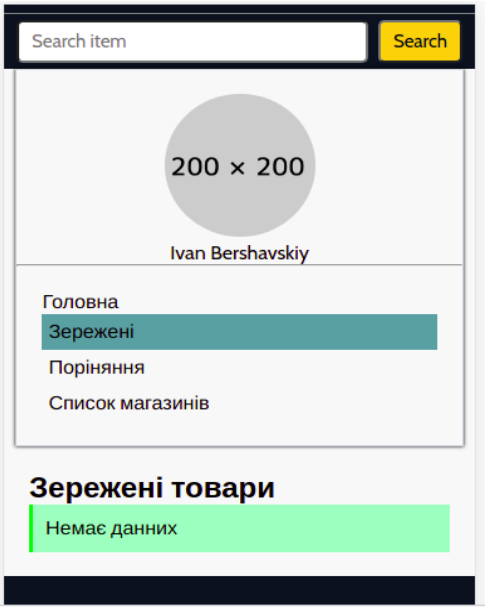

Рис. 2.28. - Профіль користувача, вигляд на мобільних пристроях

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1500;
2. коефіцієнт складності програми – 1,4;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 110 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 15 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p) Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 1{,}2 \cdot (1 + 0{,}05) = 1890$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K} t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_u = \frac{1890 \cdot 1{,}2}{85 \cdot 1{,}2} = 22{,}23 \, t_u = \frac{1890 \cdot 1{,}2}{85 \cdot 1{,}2} = 22{,}23$$ людино-годин.

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot K} t_a = \frac{Q}{(20 \ldots 25) \cdot K}, \quad (3.4)$$

$$t_a = \frac{1890}{23 \cdot 1{,}2} = 68{,}48 t_a = \frac{1890}{23 \cdot 1{,}2} = 68{,}48$$ людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25) \cdot K} t_n = \frac{Q}{(20 \ldots 25) \cdot K}, \quad (3.5)$$

$$t_n = \frac{1890}{21 \cdot 1{,}2} = 75 t_n = \frac{1890}{21 \cdot 1{,}2} = 75$$ людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4 \ldots 5) \cdot K} t_{отл} = \frac{Q}{(4 \ldots 5) \cdot K}, \quad (3.6)$$

$$t_n = \frac{1890}{5 \cdot 1{,}2} = 315 t_n = \frac{1890}{5 \cdot 1{,}2} = 315$$ людино-годин,

– за умови комплексного налагодження завдання:

$$t^к_{отл} = 1{,}2 \cdot t_{отл} t^к_{отл} = 1{,}2 \cdot t_{отл}, \quad (3.7)$$

$$t^к_{отл} = 1{,}2 \cdot 315 = 378 t^к_{отл} = 1{,}2 \cdot 315 = 378$$ людино-годин

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до} t_д = t_{др} + t_{до}, \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_д = \frac{Q}{(15...20)\cdot K} t_д = \frac{Q}{(15...20)\cdot K}, \quad (3.9)$$

$$t_д = \frac{1890}{20 \cdot 1,2} = 78,75 t_д = \frac{1890}{20 \cdot 1,2} = 78,75$$ людино-годин.

$t_{до}$– трудомісткість редагування, печатки й оформлення документації

$$t_{до}=0,75 \cdot t_{др}, \quad (3.10)$$

$$t_{до} = 0,75 \cdot 68,48 = 51,36 t_{до} = 0,75 \cdot 68,48 = 51,36$$ людино-годин.

$$t_д = 68,48 + 51,36 = 119,84 t_д = 68,48 + 51,36 = 119,84$$ людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +22,23+ 68,48 + 75 + 378 + 119,84 = 713,55 людино-годин.

У результаті ми розрахували, що в загальній складності необхідно близько 713,55 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ включають витрати на заробітну плату виконавця програми З/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{по} = З_{зп} + З_{мв} \text{ грн.}, \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{зп} = t \cdot С_{пр} \text{ грн.}, \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{зп} = 713.55 \cdot 110 = 78490.5 \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М} \text{ грн.}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 378 \cdot 15 = 56703_{МВ} = 378 \cdot 15 = 5670 \text{ грн.}$$

$$К_{по} = 78490.5 + 5680 = 84160{,}5 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.}, \quad (3.14)$$

де $В_к$- число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{713,55}{1 \cdot 176} = 4,05 T = \frac{713,55}{1 \cdot 176} = 4,05 \text{ міс.}$$

Висновки. На розробку даного програмного забезпечення піде близько 713,55 людино-години. Тобто, ймовірна очікувана тривалість розробки складатиме 4,05 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 84160,5 грн.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи був розроблений практичний та надійний веб-застосунок, використовуючи сучасні підходи та інструменти для створення клієнтської частини веб застосунку.

Перевагами використання сучасних підходів до створення програми є подальша легкість впровадження, за необхідності, додаткових функцій, за рахунок модульної архітектури.

Додаток призначений для того, щоб надати людям, які перебувають в пошуку необхідного товару зручний інструмент для відстеження цін на товар в окремому магазині, на відповідному проміжку часу. Для наглядності дані наводяться у вигляді графіка.

Веб-застосунок має два режими роботи, для авторизованого користувача та гостя. Гість має змогу знаходити товар за допомогою пошуку та фільтрів, і переглядати дані про нього.

Авторизований користувач має розширені можливості. Він може додавати товар до обраних та додавати товар до порівняння.

Обрані товари користувач має змогу переглядати щоразу після авторизації. Порівняння і збережені товари відображаються в профі користувача на відповідній вкладці.

В результаті отримано застосунок, який має практичну цінність, тому, що вирішує реальні проблеми такі як заощадження часу та коштів.

Під час виконання даної кваліфікаційної роботи були виконані пройдені наступні етапи створення програмного продукту:

- аналіз предметної галузі задачі, що розв’язується;
- обрання раціональної архітектури та технології створення додатку;
- написання програмного коду веб-додатку;
- розробка рекомендацій щодо використання застосунку;

Під час виконання кваліфікаційної роботи також було визначено трудомісткість розробленого програмного продукту 713,55 людино-години,

проведений підрахунок вартості роботи по створенню програми 84160,5 грн та розраховано час на його створення 4.05 міс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Vue3. URL: https://v3.vuejs.org/.

2. Офіційна документація Vue Rourer. URL: https://router.vuejs.org/.

3. Офіційна документація Vuex. URL:https://next.vuex.vuejs.org/.

4. Офіційна документація Axios. URL: https://axios-http.com/docs/intro.

5. Vue Style Guide. URL: https://vuejs.org/v2/style-guide/.

6. Эрик Хэнчетт, Бенджамин Листуон. Vue.js в действиию. - СПб.: Питер, 2019. 306 с.

7. Gus Khwaja. Practical Web Penetration Testing. – ISBN: 1788624033, 2018. 429 с.

8. Callum Macrae. Vue.js: Up and Running, - O'REILLY Media. 2018. 174 с.

9. Безпека JSON Web Tokens. URL:bhttps://cyberpolygon.com/ru/materials/security-of-json-web-tokens-jwt/.

10. Scott Chacon, Ben Straub. Pro Git. - APRESS, 2020 543 с.

11. Солодушкин С. И., Юманова И. Ф. Разработка программных комплексов на языке Javascript. - Издательство Уральского университета. 2020. 136 с.

12. Edd Yerburgh, Testing Vue.js Applications, – MANING. 2018. 272 с.

13. Слободан Стоянович, Александар Симович. Бессерверные приложения на JavaScript, - Москва : ДМК Пресс. 2020. 394 с.

14. What is Vuex. URL: https://riptutorial.com/vue-js/example/11859/what-is-vuex-

15. Luis Atencio. The Joy of JavaScript, - MANING. 2021. 362 с.

16. Waweru Mwaura. End-to-end Web Testing With Cypress. Explore techniques for automated frontend web testing with Cypress and JavaScript, Packt Publishing. 2021. 241 с.

17. Дэвид Флэнаган. JavaScript. Полное руководство. 7-е издание, – O'REILLY Media. 2021. 720 с.

18. John Resig, Bear Bibeault, Josip Maras. Secrets of the JavaScript Ninja, - MANING. 2016. 464 с.

19. Алексей Крючков. Создание приложений для браузера Google Chrome, - SelfPub, 2018. 23 с.

20. Adam D. Scott, Matthew MacDonald, Shelley Powers. JavaScript Cookbook: Programming the Web, - O'REILLY Media. 2021. 527 с.

21. Кириченко А. В. JavaScript для FrontEnd-разработчиков. Написание. Тестирование. Развертывание, - Наука и Техника. 2020. 322с.

22. A Closer Look at Vue Router. URL: https://vuejsdevelopers.com/2020/01-/27/closer-look-at-vue-router/

23. John Dean. Web Programming with HTML5, CSS, and JavaScript, - Jones & Bartlett Learning, 2018. – 700 с.

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг store index.js:

```
import { createStore } from "vuex";
import { actions } from "./actions";
import { mutations } from "./mutations";

export default createStore({
  state: {
    token: localStorage.getItem("token") || "",// jwt токен пористувача для ыдентицыкації
    status: "",
    user: {},
    compareProducts: JSON.parse(sessionStorage.getItem("compare-product")) || {}// списак
продуктів для порівняння
  },
  getters: {
    isAuthenticated: state => !!state.token, //перевірка чи користувач увійшов в систему
    authStatus: state => state.status // статус авторизіції
  },
  mutations,
  actions,
  modules: {}
});
```

Лістинг mutations.js:

```
export const mutations = {
  authenticated(state, user) {
    state.user = user;
  },
  settoken(state, token) {// установка точена
    localStorage.setItem("token", token);
    state.token = token;
  },
  auth_request(state) {
    state.status = "loading";
  },
  auth_success(state, token, user) {// успішна авторизація
    state.status = "success";
    state.token = token;
    state.user = user;
  },
  auth_error(state) { // пмилка при авторизації
    state.status = "error";
  },
  logout(state) { // вихід з системи
    state.status = "";
    state.token = "";
    state.user = {};
```

```
  },
  COMPARE_UPDATE: state => {// оновлення спискупорівнянь
    state.compareProducts = JSON.parse(sessionStorage.getItem("compare-product")) || {};
  },
  ADD_ITEM_TO_COMPARE(state, product) { // додавання елементу до списку порівнянь
    state.compareProducts.notebook.push(product);
  }
};
```

Лістинг main.js:

```
 import { createApp } from "vue";
// import Axios from "axios";
import App from "./App.vue";
import router from "./router";
import store from "./store";

createApp(App)
  .use(store)
  .use(router)
  .mount("#app");
```

Лістинг App.vue:

```
<template>
  <the-header />
  <router-view />
  <the-footer />
</template>

<script>
import TheFooter from "./components/TheFooter.vue";
import TheHeader from "./components/TheHeader.vue";

export default {
  components: { TheHeader, TheFooter },
  data() {
    return {
      mobileScreen: null,
      sizeScreen: null
    };
  },
  created() {
    window.addEventListener("resize", this.handlerResize);. // для отримання розміру вікна
користувача
    this.handlerResize();
  },
  unmounted() {
    window.removeEventListener("resize", this.handlerResize);
  },
  methods: {
    handlerResize() {
      this.sizeScreen = window.innerWidth;
```

```
      if (window.innerWidth < 750) {
        this.mobileScreen = true;
      } else this.mobileScreen = false;
    }
  },
  mounted() {
    const x = localStorage.getItem("token");
    this.$store.dispatch("AUTH", x);
  }
};
</script>
```

Лістинг roter/index.js:

```
import { createRouter, createWebHistory } from "vue-router";
import Product from "../views/product/Product.vue";
import About from "../views/About.vue";
import store from "../store";

const ifAuthenticated = (to, from, next) => {
  if (store.getters.isAuthenticated) {
    next();
    return;
  }
  next("/login");
};
const ifNotAuthenticated = (to, from, next) => {
  if (!store.getters.isAuthenticated) {
    next();
    return;
  }
  next("/");
};

const routes = [
  {
    path: "/",
    name: "Home",
    component: () => import(/* webpackChunkName: "home" */ "../views/Home.vue")
  },
  {
    path: "/comparecategoty/:category",
    name: "CompareCategory",
    component: () =>
      import(/* webpackChunkName: "compareproducts" */ "../views/CompareProductView.vue")
  },
  {
    path: "/profile",
    name: "Profile",
    component: () => import(/* webpackChunkName: "profile" */ "../views/profile/Profile.vue"),
    beforeEnter: ifAuthenticated,
```

```
    children: [
      {
        path: "",
        name: "About",
        component: About
      },
      {
        path: "bookmarks",
        name: "BokmarksItem",
        component: () =>
          import(
            /* webpackChunkName: "bookmarks" */ "../views/profile/profile-
subroutes/BookmarksItem.vue"
          )
      },
      {
        path: "compare",
        name: "CompareItem",
        component: () =>
          import(
            /* webpackChunkName: "compare" */ "../views/profile/profile-
subroutes/CompareProduct.vue"
          )
      },

      {
        path: "shops",
        name: "ShopsItem",
        component: () =>
          import(/* webpackChunkName: "shops" */ "../views/profile/profile-
subroutes/ShopsItem.vue")
      }
    ]
  },

  {
    path: "/:category",
    name: "Products",
    component: () => import(/* webpackChunkName: "products" */ "../views/Products.vue")
  },
  {
    path: "/:category/:filterOption",
    name: "ProductsFiltration",
    component: () => import(/* webpackChunkName: "products" */ "../views/Products.vue")
  },
  {
    path: "/product/:id",
    name: "Product",
    component: Product,
    children: [
      {
        path: "description",
```

```
        name: "Description",
        component: () =>
          import(/* webpackChunkName: "description" */ "../views/product/Description.vue")
      },
      {
        path: "bue",
        name: "Bue",
        component: () => import(/* webpackChunkName: "bue" */ "../views/product/Bue.vue")
      },
      {
        path: "reviews",
        name: "Reviews",
        component: () => import(/* webpackChunkName: "reviews" */
"../views/product/Reviews.vue")
      }
    ]
  },
  // { path: "/product/:name/prod/:id", name: "About", component: About },
  {
    path: "/login",
    name: "Login",
    // component: Login,
    component: () => import(/* webpackChunkName: "logins" */ "../views/Login.vue"),

    beforeEnter: ifNotAuthenticated
  },
  {
    path: "/registration",
    name: "Registration",
    component: () => import(/* webpackChunkName: "registration" */ "../views/Registration.vue")
    // component: Registration
  }
];

const router = createRouter({
  scrollBehavior(to) {
    if (to.name === "Products") window.scrollTo(0, 0);
  },
  history: createWebHistory(process.env.BASE_URL),
  routes
});

export default router;
```

Лістинг validations/validations.js:

```
const validation = {
  isEmail(login) {
    const re = /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
    return re.test(login);
  },
```

```
  minLength(value, length) {
    return value.length >= length;
  },
  maxLength(value, length) {
    return value.length <= length;
  },
  required(value) {
    return value.length > 1;
  }
};

export { validation };
```

Лістинг views/Home.vue:

```
 <template>
  <div class="view-main">
    <div class="aside-category-menu">
      <app-categories />
    </div>
    <div class="main">
      <div>
        <h1>Останні новини і анонси:</h1>
      </div>
      <div class="cards-conteiner">
        <home-card
          v-for="(item, idx) in items"
          :key="idx"
          :img="item.sourceName"
          :title="item.title"
          :body="item.body"
        />
      </div>
    </div>
  </div>
  <div class="home_about">
    <h3>Market Dinamics Analizer – сервіс вибору товарів та пропозицій інтернет-
магазинів</h3>
    Тут ви підберете потрібний товар, порівняєте ціни на нього та вирішите, в якому
    інтернет-магазині здійснити покупку. Вашими головними помічниками у виборі товару
будуть розумні
    фільтри та гіди покупця. Також вам допоможуть відгуки реальних покупців про товари та
магазини і
    різноманітні можливості сортування та відбору пропозицій.
  </div>
  <app-history-carousel />
</template>

<script>
import AppCategories from "../components/AppCategories.vue";
import AppHistoryCarousel from "../components/AppHistoryCarousel.vue";
import HomeCard from "../components/HomeCard.vue";
```

```
export default {
  name: "Home",
  components: { AppHistoryCarousel, AppCategories, HomeCard },
  data() {
    return {
      items: [
        {
          sourceName: "1",
          title: "Складно  знайти товар?",
          body: "Скористайтеся зручними фільтрами для пошуку товару з необхідними
характеристиками"
        },
        {
          sourceName: "2",
          title: "Обираєш шолом?",
          body: "У нас є безліч шоломів для будь якого користувача"
        },
        {
          sourceName: "3",
          title: "У пошуках ігрового ноутбуку?",
          body: "У ми для тебе зібрали найкращі пропозиції в Україні"
        },
        {
          sourceName: "4",
          title: "Не знаєш де обрати відеокарту без шалених націнок?",
          body: "Це було складно, але нам вдалось знайти дещо цікаве для тебе"
        },
        {
          sourceName: "5",
          title: "Вирішив перекваліфікуватись у лісоруба?",
          body: "У нас для вас є дещо цікаве. Думаємо вам сподобається"
        }
      ]
    };
  },
  computed: {
    isPc() {
      return this.$root.sizeScreen > 750;
    }
  },

  methods: {
    closeCategory() {
      this.$refs.asideCategory.style.width = "0";
    },
    openCategory() {
      this.$refs.asideCategory.style.width = "100%";
    }
  }
};
</script>
```

Лістинг views/Registration.vue:

```
<template>
  <div class="view-registration">
    <div class="form-container">
      <h1>Реєстрація</h1>
      <div class="registration-form">
        <form @submit.prevent="handleForm">
          <form-input
            v-model="user.firstName.value"
            @blur="user.firstName.isTouched = true"
            @focus="user.firstName.isTouched = false"
            :isValide="user.firstName.isValide"
            :isInvalide="!user.firstName.required && user.firstName.isTouched"
            :plaseHolder="'Ім\'я'"
          />
          <form-input-error
            v-if="!user.firstName.required && user.firstName.isTouched"
            :errorMessage="'Введіть ім\'я'"
          />
          <form-input
            v-model="user.lastName.value"
            @blur="user.lastName.isTouched = true"
            @focus="user.lastName.isTouched = false"
            :isValide="user.lastName.isValide"
            :isInvalide="!user.lastName.required && user.lastName.isTouched"
            :plaseHolder="'Прізвище'"
          />
          <form-input-error
            v-if="!user.lastName.required && user.lastName.isTouched"
            :errorMessage="'Введіть прізвище'"
          />
          <form-input
            v-model="user.email.value"
            @blur="user.email.isTouched = true"
            @focus="user.email.isTouched = false"
            :isValide="user.email.isValide"
            :isInvalide="(!user.email.required || !user.email.isEmail) && user.email.isTouched"
            :plaseHolder="'Електронна пошта'"
          />
          <form-input-error
            v-if="!user.email.required && user.email.isTouched"
            :errorMessage="'Введіть електронну пошту'"
          />
          <form-input-error
            v-else-if="!user.email.isEmail && user.email.isTouched"
            :errorMessage="'Невірний формат (приклад: email.email.com)'"
          />
          <form-input
            v-model="user.password.value"
```

```
          @blur="user.password.isTouched = true"
          @focus="user.password.isTouched = false"
          :isValide="user.password.isValide"
          :isInvalide="
            (!user.password.minLength && user.password.isTouched) ||
              (!user.password.maxLength && user.password.isTouched)
          "
          :plaseHolder="'Пароль'"
        />
        <form-input-error
          v-if="!user.password.required && user.password.isTouched"
          :errorMessage="'Введіть пароль'"
        />
        <form-input-error
          v-else-if="!user.password.minLength && user.password.isTouched"
          :errorMessage="'Короткий пароль (мінімум 6 символів)'"
        />
        <form-input-error
          v-else-if="!user.password.maxLength && user.password.isTouched"
          :errorMessage="'Занадто довгий пароль'"
        />

        <form-input
          v-model="user.confirmPassword.value"
          @blur="user.confirmPassword.isTouched = true"
          @focus="user.confirmPassword.isTouched = false"
          :isValide="user.confirmPassword.isValide"
          :isInvalide="
            (!user.confirmPassword.minLength && user.confirmPassword.isTouched) ||
              (!user.confirmPassword.maxLength && user.confirmPassword.isTouched) ||
              (!user.confirmPassword.equalPassword && user.confirmPassword.isTouched)
          "
          :plaseHolder="'Підтвердження паролю'"
        />
        <form-input-error
          v-if="!user.confirmPassword.required && user.confirmPassword.isTouched"
          :errorMessage="'Введіть пароль'"
        />
        <form-input-error
          v-else-if="!user.confirmPassword.equalPassword && user.confirmPassword.isTouched"
          :errorMessage="'Поля не співпадають'"
        />
        <input type="submit" value="Зареєструватися" class="submitt-button" />
      </form>
    </div>
  </div>
 </div>
</template>

<script>
import { computed } from "vue";
import FormInput from "../components/FormInput.vue";
```

```
import { validation } from "../validations/validations";
import FormInputError from "../components/FormInputError.vue";

export default {
  components: { FormInput, FormInputError },
  data() {
    return {
      formValid: computed(() => {
        return Object.keys(this.user).every(key => this.user[key].isValide === true);
      }),
// валідація полів у реальному часі
      user: {
        firstName: {
          value: "",
          isTouched: false,
          required: computed(() => validation.required(this.user.firstName.value)),
          isValide: computed(() => this.user.firstName.required && this.user.firstName.isTouched)
        },
        lastName: {
          value: "",
          isTouched: false,
          required: computed(() => validation.required(this.user.lastName.value)),
          isValide: computed(() => this.user.lastName.required && this.user.lastName.isTouched)
        },
        email: {
          value: "",
          isTouched: false,
          required: computed(() => validation.required(this.user.email.value)),
          isEmail: computed(() => validation.isEmail(this.user.email.value)),
          isValide: computed(
            () => this.user.email.required && this.user.email.isEmail && this.user.email.isTouched
          )
        },
        password: {
          value: "",
          isTouched: false,
          required: computed(() => validation.required(this.user.password.value)),
          minLength: computed(() => validation.minLength(this.user.password.value, 6)),
          maxLength: computed(() => validation.maxLength(this.user.password.value, 10)),
          isValide: computed(
            () =>
              this.user.password.required &&
              this.user.password.minLength &&
              this.user.password.maxLength
          )
        },
        confirmPassword: {
          value: "",
          isTouched: false,
          required: computed(() => validation.required(this.user.confirmPassword.value)),
          minLength: computed(() => validation.minLength(this.user.confirmPassword.value, 6)),
          maxLength: computed(() => validation.maxLength(this.user.confirmPassword.value, 10)),
```

```
        equalPassword: computed(
          () => this.user.password.value === this.user.confirmPassword.value
        ),
        isValide: computed(
          () =>
            this.user.confirmPassword.required &&
            this.user.confirmPassword.minLength &&
            this.user.confirmPassword.maxLength &&
            this.user.confirmPassword.equalPassword
        )
      }
    }
  };
 },
 methods: {
  handleForm() {// запит на реєстраціє користувача
    const sendData = {
      firstName: this.user.firstName.value,
      lastName: this.user.lastName.value,
      emailAddress: this.user.email.value,
      password: this.user.password.value
    };
    console.log(sendData);
    if (this.formValid) {
      this.$store
        .dispatch("REGISTER", sendData)
        .then(() => this.$router.push("/"))
        .catch(err => console.log(err));
    }
  }
 }
};
</script>
```

Лістинг views/Login.vue:

```
<template>
 <div class="view-login">
  <div class="form-container">
   <h1>Авторизація</h1>
   <form @submit.prevent="handleForm">
    <form-input
     v-model="user.login.value"
     :isValide="user.login.isEmail && user.login.isTouched"
     :isInvalide="!user.login.isEmail && user.login.isTouched"
     @blur="user.login.isTouched = true"
     @focus="user.login.isTouched = false"
     :plaseHolder="'Логін'"
    />
    <form-input-error
     v-if="!user.login.isEmail && user.login.isTouched"
     :errorMessage="'Введіть електронну пошту'"
```

```
      />
      <form-input
        v-model="user.password.value"
        :isValide="user.password.minLength && user.password.isTouched"
        :isInvalide="
          (!user.password.minLength && user.password.isTouched) ||
            (!user.password.maxLength && user.password.isTouched)
        "
        @blur="user.password.isTouched = true"
        @focus="user.password.isTouched = false"
        :plaseHolder="'Пароль'"
      />
      <form-input-error
        v-if="
          (!user.password.minLength && user.password.isTouched) ||
            (!user.password.maxLength && user.password.isTouched)
        "
        :errorMessage="'Невірний формат'"
      />
      <input type="submit" value="Авторизуватися" class="submitt-button" />
    </form>
  </div>
 </div>
</template>

<script>
import { computed } from "vue";
import { validation } from "../validations/validations";
import FormInput from "../components/FormInput.vue";
import FormInputError from "../components/FormInputError.vue";

export default {
 components: { FormInput, FormInputError },
 data() {
  return {
   user: {
// поля  та їх валідація
      login: {
        value: "",
        isTouched: false,
        validators: { isEmail: validation.isEmail, required: validation.required },
        isEmail: computed(() => validation.isEmail(this.user.login.value)),
        isValide: computed(() => this.user.login.isEmail)
      },
      password: {
        value: "",
        isTouched: false,
        minLength: computed(() => validation.minLength(this.user.password.value, 6)),
        maxLength: computed(() => validation.maxLength(this.user.password.value, 10)),
        isValide: computed(() => this.user.password.minLength && this.user.password.maxLength)
      }
    },
```

```
    valide: computed(() => Object.keys(this.user).every(key => this.user[key].isValide))
   };
  },
  computed: {},
  methods: {
   handleForm() {
    console.log(this.user);
// вхід користувача в систему
    this.$store
     .dispatch("LOGIN", {
      email_address: this.user.login.value,
      password: this.user.password.value
     })
     .then(() => this.$router.push("/"))
     .catch(err => console.log(err));
   }
  }
};
</script>
```

Лістинг views/product/Product.vue:

```
<template>
  <div class="product-view">
   <button @click="bookmarkProduct">Додати в обрані</button>
   <!-- <button @click="compareProduct">Add to compare</button> -->
   <div class="product-title">
    <h3>HUAWEI MateBook D 16 R5-4600H/16GB/512/Win10</h3>
   </div>

   <div class="product-info">
    <product-slider :image="images" />
    <!-- Carousell -->
    <div class="product-description">
     <p class="offers-count">Кількість пропозицій: 3</p>
     <div class="title">
      <h3>Описання</h3>
     </div>
     <p>
      HUAWEI MateBook D 16 - елегантний і практичний ноутбук, створений для тих
користувачів,
      які шукають одночасно продуктивне, тонке і легке рішення для повсякденної роботи.
Завдяки
      мінімально тонким рамкам 16,1-дюймову IPS-матрицю вдалося вмістити в габарити,
відповідні
      14-дюймовим пристроїв. Дисплей HUAWEI MateBook D 16 отримав дозвіл 1920x1080
пікселів,
      колірний обхват в 100% простору sRGB і кутами огляду 170 градусів. Висока
швидкість роботи
      досягається зусиллями процесорів AMD Ryzen 5 і Ryzen 7 серії 4000. Вони доповнені
      просунутими інтегрованими графічними чіпами AMD Radeon Vega. Оперативної
пам'яті
```

```
        встановлюється 16 ГБ, а для зберігання даних використовується SSD об'ємом в 512 ГБ.
        Приємною опцією в HUAWEI MateBook D 16 є наявність сканера відбитків пальців, а
також
        підтримки швидкої зарядки, 15 хвилин якого досить для того, щоб продовжити
автономність
        додатково на 2 години.
       </p>
       <div class="title">
        <h3>Короткі характеристики</h3>
       </div>
       <h3></h3>
       <p>
        Ноутбук Класичний 16" IPS 1920x1080 AMD Ryzen 5 4600H 3,0 ГГц ОЗП: 16 ГБ AMD
Radeon
        Graphics SSD: 512 ГБ 1,75 кг ОС: Windows 10 Home
       </p>
     </div>
    </div>
    <div id="container"></div>
    <div class="tab">
     <ul class="tab-list">
      <li class="tab-item" @click="scrollToElement">
       <router-link :to="`/product/${this.$route.params.id}/${'description'}`" class="link"
        >Характеристики</router-link
       >
      </li>
      <li class="tab-item" @click="scrollToElement">
       <router-link :to="`/product/${this.$route.params.id}/${'bue'}`" class="link"
        >Купити</router-link
       >
      </li>

      <li class="tab-item" @click="scrollToElement">
       <router-link :to="`/product/${this.$route.params.id}/${'reviews'}`" class="link"
        >Відгуки</router-link
       >
      </li>
     </ul>
    </div>
    <div class="viev-box-characteristics">
     <router-view />
    </div>
   </div>
</template>

<script>
// import store from "../../store";
import axios from "axios";
import ProductSlider from "../../components/ProductSlider.vue";

export default {
  components: { ProductSlider },
```

```
data() {
  return {
    active: true,
    images: [],
    productInfo: {},
    tabNameList: [
      { name: "Description", param: "description" },
      { name: "Bue", param: "bue" },
      { name: "Reviews", param: "reviews" }
    ]
  };
},
methods: {
  scrollToElement() {
    const container = this.$el.querySelector("#container");
    container.scrollIntoView({ behavior: "smooth" });
  },
  bookmarkProduct() {
    if (localStorage.getItem("bookmarks-item") !== null) {
      const array = JSON.parse(localStorage.getItem("bookmarks-item"));

      if (!array.some(e => e.userId === this.$route.params.id)) {
        array.push({
          userId: null || this.$route.params.id,
          categoryId: 2,
          productId: this.productInfo.id
        });
        localStorage.setItem("bookmarks-item", JSON.stringify(array));
      }
    } else {
      const array = [];
      array.push({
        userId: 2,
        categoryId: this.productInfo.categoryId,
        productId: this.productInfo.id
      });
      localStorage.setItem("bookmarks-item", JSON.stringify(array));
    }
  },
  compareProduct() {
    const product = {
      id: this.$route.params.id,
      img: "https://hotline.ua/img/yp/287_.jpg",
      title: "Notebook lenovo",
      price: "28123,2"
    };
    this.$store.dispatch("ADD_PRODUCT_TO_COMPARE", product);
  },
  addHistory() {
    if (localStorage.getItem("watch-history") !== null) {
      const array = JSON.parse(localStorage.getItem("watch-history"));
      if (!array.some(e => e.id === this.$route.params.id)) {
```

```
        // добавление елементов в масив в lockalStorage
        array.push({
          id: this.$route.params.id,
          img: this.images[0],
          title: this.productInfo.name,
          price: Number(this.productInfo.currentPrice)
        });
        localStorage.setItem("watch-history", JSON.stringify(array));
      }
    } else {
      const array = [];
      array.push({
        id: this.$route.params.id,
        img: this.images[0],
        title: this.productInfo.name,
        price: this.productInfo.currentPrice
      });
      localStorage.setItem("watch-history", JSON.stringify(array));
    }
  }
 },
 mounted() {
  axios({
    url: `http://localhost:3000/products/product/${this.$route.params.id}`,
    method: "GET"
  })
    .then(resp => {
      this.productInfo = resp.data;
      const arr = this.productInfo.image.split(" ");
      this.images = arr;
    })
    .then(() => {
      this.addHistory();
    });
 }
};
</script>
```

Лістинг views/product/Bue.vue:

```
<template>
 <div class="view-bue">
  <div class="filter-options">
    <button @click="sortEncrease">По зростанню ціни</button>
    <button @click="sortDecrease">По спаданню ціни</button>
  </div>
  <div v-for="shop in shops" :key="shop.id" class="shop-card">
    <img :src="shop.logo" height="100" class="shop-card__image" />
    <div class="shop-card__title">{{ shop.title }}</div>
    <p class="shop-card__description">{{ shop.description }}</p>
    <div class="shop-card__price">{{ shop.price }} грн</div>
  </div>
```

```
    <div ref="chart" class="chart">
      <monthly-chart
        v-bind:chartData="chartData"
        v-bind:chartOptions="chartOptions"
        :width="$root.sizeScreen"
        :height="400"
      />
    </div>
  </div>
</template>

<script>
import shopsData from "@/db/shops.json";
import MonthlyChart from "@/components/MonthlyChart.vue";

export default {
  components: { MonthlyChart },
  data() {
    return {
      shops: [],
      chartData: {
        datasets: [
          {
            label: "OpenShop",
            data: [29090, 29120, 29290, 29780, 29890, 29790, 29750, 29710, 29590, 29456],
            borderColor: "Green",
            fill: false,
            backgroundColor: "Black"
          },
          {
            label: "STYLUS",
            data: [29290, 29390, 29290, 29490, 29590, 29390, 29690, 29770, 29900, 29990],
            fill: false,
            borderColor: "Yellow ",
            backgroundColor: "Black"
          },
          {
            label: "Цифра",
            data: [29090, 29190, 29290, 29390, 29350, 29490, 29590, 29690, 29790, 29900],
            fill: false,
            borderColor: "Red",
            backgroundColor: "Black"
          }
        ],
        // These labels appear in the legend and in the tooltips when hovering different arcs
        labels: [
          "01.01.2021",
          "02.01.2021",
          "03.01.2021",
          "04.01.2021",
          "05.01.2021",
          "06.01.2021",
```

```
        "07.01.2021",
        "08.01.2021",
        "09.01.2021",
        "10.01.2021"
      ]
    },
    chartOptions: {
      scales: {
        yAxes: [
          {
            // ticks: {
            //   beginAtZero: true
            // }
          }
        ]
      }
    }
  };
 },
 methods: {
  sortEncrease() {
    this.shops.sort((a, b) => (a.price > b.price ? 1 : -1));
  },
  sortDecrease() {
    this.shops.sort((a, b) => (a.price > b.price ? -1 : 1));
  }
 },
 mounted() {
  this.shops = shopsData.shops;
 }
};
</script>
```

Лістинг views/product/Reviews:

```
<template>
 <div class="view-reviews">
  <h2 class="view-reviews__title">Відгуки</h2>
  <hr />
  <div class="view-reviews__content">
   <feedback-item v-for="(item, idx) in feedbackReqponse" :key="idx" :items="item" />
  </div>
 </div>
 <div class="feedbackForm">
  <h3>Залишити відгук</h3>
  <label for="raiting">Оцінка (від 1 до 5) </label>
  <input type="text" id="raiting" v-model="rating" />
  <br />
  <textarea
   ref="textarea"
   rows="10"
   cols="45"
```

```
      name="text"
      @keydown.enter="sendCommentaty"
      placeholder="Оставте комментарий."
      class="textarea_field"
      v-model="feedbackComment"
      @blur="isTouched = true"
    ></textarea>
    <button
      v-if="
        (isTouched === true && feedbackComment.length > 0) ||
          (feedbackComment && feedbackComment.length > 0)
      "
      @click="sendFeedback"
    >
      Оставить отзыв
    </button>
  </div>
</template>

<script>
import axios from "axios";
import FeedbackItem from "../../components/FeedbackItem.vue";

export default {
  components: { FeedbackItem },
  data() {
    return {
      feedbackReqponse: [],
      rating: "",
      feedbackComment: "",
      isTouched: false,
      timerId: undefined
    };
  },
  mounted() {
    const now = new Date().toLocaleDateString();
    console.log(now);
    this.reload();
    this.timerId = setInterval(() => {
      this.reload();
    }, 5000);
  },
  beforeUnmount() {
    clearInterval(this.timerId);
  },
  methods: {
    reload() {
      axios({
        url: `http://localhost:3000/reviews/${this.$route.params.id}`,
        method: "GET"
      }).then(resp => {
        this.feedbackReqponse = resp.data;
```

```
      });
    },
    sendFeedback() {
      const now = new Date().toLocaleDateString();
      if (
        this.rating === "1" ||
        this.rating === "2" ||
        this.rating === "3" ||
        this.rating === "4" ||
        this.rating === "5"
      ) {
        axios({
          url: `http://localhost:3000/reviews`,
          data: {
            idCommented: null,
            idProduct: this.$route.params.id,
            comment: this.feedbackComment,
            date: now,
            idUser: 2
          },
          method: "POST"
        }).then(() => {
          this.feedbackComment = "";
          this.rating = "";
        });
      }
    }
  }
};
</script>
```

Лістинг components/BaseChart.js:

```
import Chart from "chart.js";
import { defineComponent, h, reactive } from "vue";

function useChartInfo() {
  const state = reactive({
    myName: "",
    userData: {},
    userOptions: {}
  });

  function setChartData(payload) {
    state.userData = payload;
  }

  function setChartOption(payload) {
    state.userOptions = payload;
  }

  return {
```

```
    state,
    setChartData,
    setChartOption
  };
}

/**
 *
 * @param chartsId string
 * @param chartsType string
 */
function generateChart(chartsId, chartsType) {
  const { state, setChartData, setChartOption } = useChartInfo();

  return defineComponent({
    name: "BaseChart",
    props: {
      chartId: {
        type: String,
        required: false
      },
      chartType: {
        type: String,
        required: false
      },
      width: {
        type: Number,
        required: false,
        default: 400
      },
      height: {
        type: Number,
        required: false,
        default: 400
      },
      cssClasses: {
        type: String,
        required: false,
        default: ""
      },
      styles: {
        type: Object,
        required: false
      }
    },
    data() {
      return {
        state: {
          chartObj: null
        }
      };
    },
```

```
// emits: ['chart:update'],

beforeUnmount() {
  if (this.state.chartObj) {
    this.state.chartObj.destroy();
  }
},
methods: {
  renderChart(userData, userOptions) {
    setChartData(userData);
    setChartOption(userOptions);


    if (this.state.charObj != null && this.state.chartObj.data != null) {
      this.state.chartObj.data.datasets.pop();
    }

    const ctx = this.$refs.canvas.getContext("2d");
    this.state.chartObj = new Chart(ctx, {
      type: chartsType,
      data: userData,
      options: userOptions
    });
  }
},
beforeMount() {
  if (document.getElementById(chartsId)) {
    const ctx = document.getElementById(chartsId).getContext("2d");
    this.state.chartObj = new Chart(ctx, {
      type: chartsType,
      data: {
        datasets: [
          {
            data: [1, 2, 3, 4],
            backgroundColor: ["Red", "Yellow", "Blue", "Green"]
          }
        ],
        labels: ["Red", "Yellow", "Blue", "Green"]
      },
      options: {
        responsive: false
      }
    });
  }
},
computed: {
  currentChartData() {
    return state.userData;
  },
  currentChartOption() {
    return state.userOptions;
  }
```

```
    },
    watch: {
      chartData(prevState, newState) {
        if (prevState !== newState) {
          this.renderChart(newState, this.currentChartOption);
        }
      }
    },
    render() {
      return h("div", { style: this.styles, class: this.cssClasses }, [
        h("canvas", {
          ref: "canvas",
          id: this.chartId,
          width: this.width,
          height: this.height
        })
      ]);
    }
  });
}

const Bar = generateChart("bar-chart", "bar");
const Bubble = generateChart("bubble-chart", "bubble");
const Doughnut = generateChart("doughnut-chart", "doughnut");
const HorizontalBar = generateChart("horizontalbar-chart", "horizontalBar");
const Line = generateChart("line-chart", "line");
const Pie = generateChart("pie-chart", "pie");
const PolarArea = generateChart("polar-chart", "polarArea");
const Radar = generateChart("radar-chart", "radar");
const Scatter = generateChart("scatter-chart", "scatter");

export {
  Bar,
  Bubble,
  Doughnut,
  HorizontalBar,
  Line,
  Pie,
  PolarArea,
  Radar,
  Scatter,
  generateChart
};
```

Лістинг components/AppPagination.vue:

```
<template>
  <div class="pagination">
    <div class="pagination-container">
      <ul>
        <li class="pagination-button prev" @click="changePage(prevPage)"></li>
        <li v-if="hasFirst()" @click.prevent="changePage(1)" class="pagination-button">
```

```
        1
      </li>
      <!-- <li class="pagination-button" v-if="hasFirst()" disable>...</li> -->
      <li
        class="pagination-button"
        v-for="page in pages"
        :key="page"
        @click.prevent="changePage(page)"
        :class="{
          'active-page': current == page
        }"
      >
        {{ page }}
      </li>
      <!-- <li class="pagination-button" v-if="hasLast()">...</li> -->
      <li class="pagination-button" v-if="hasLast()" @click.prevent="changePage(totalPages)">
        {{ totalPages }}
      </li>
      <li class="pagination-button next" @click.prevent="changePage(nextPage)"></li>
    </ul>
  </div>
 </div>
</template>
<script>
export default {
 name: "PaginationComponent",
 props: {
  current: {
   type: Number,
   default: 1
  },
  total: {
   type: Number,
   default: 100
  },
  perPage: {
   type: Number,
   default: 9
  },
  pageRange: {
   type: Number,
   default: 2
  }
 },
 data() {
  return {};
 },
 methods: {
  changePage(page) {
   if (page > 0 && page <= this.totalPages) {
    console.log("Next", page);
    this.$emit("change-page", page);
```

```
    }
   },
   hasFirst() {
    return this.rangeStart !== 1;
   },
   hasLast() {
    return this.rangeEnd < this.totalPages;
   },
   hasPrev() {
    return this.current > 1;
   },
   hasNext() {
    return this.current < this.totalPages;
   }
  },
  computed: {
   pages() {
    const pages = [];
    for (let i = this.rangeStart; i <= this.rangeEnd; i += 1) {
     pages.push(i);
    }
    return pages;
   },
   rangeStart() {
    const start = this.current - this.pageRange;
    return start > 0 ? start : 1;
   },
   rangeEnd() {
    const end = this.current + this.pageRange;
    return end < this.totalPages ? end : this.totalPages;
   },
   totalPages() {
    return Math.ceil(this.total / this.perPage);
   },
   prevPage() {
    return this.current - 1;
   },
   nextPage() {
    return this.current + 1;
   },
   isFirst() {
    return this.current > 1;
   }
  }
};
</script>
```

Лістинг components/ProductFiltration.vue:

```
<template>
 <div class="sort">
```

```
    <div class="select">
      Сортувати за:
      <select v-model="selectedSortOption" class="select-item">
        <option value="" hidden selected>{{ selectedSortOption }}</option>
        <option v-for="(option, idx) in sortOptions" :value="option" :key="idx">
          {{ option }}
        </option>
      </select>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      sortOptions: ["За популярністю", "За зростанням ціни", "За спаданням ціни", "За
відгуками"],
      selectedSortOption: "За популярністю"
    };
  }
};
</script>
```

Лістинг components/ProductSlider.vue:

```
<template>
  <div class="slider-wrap">
    <div class="product-slider">
      <div class="slide-nav">
        <img
          v-for="(item, key) in image"
          :src="item"
          :key="key"
          next-button
          @click="selected = key"
          :class="{ activeImg: selected === key }"
        />
      </div>
      <div class="active-slide">
        <button v-if="$root.mobileScreen" @click="goPrev" class="prev-button"></button>
        <button v-if="$root.mobileScreen" @click="goNext" class="next-button"></button>
        <img :src="image[selected]" class="img-view" />
      </div>
    </div>
  </div>
</template>

<script>
export default {
  props: ["image"],
  data() {
```

```
    return {
      selected: 0
     };
  },
  methods: {
    goNext() {
      if (this.selected < this.image.length - 1) this.selected += 1;
    },
    goPrev() {
      if (this.selected > 0) this.selected -= 1;
    }
  }
};
</script>
```

Лістинг components/TheHeader.vue:

```
<template>
  <div class="the-header">
    <header-logo />
    <app-search v-if="!mobileScreen" />
    <header-actions />
  </div>
  <div v-if="mobileScreen" class="mobile-search-form">
    <app-search />
  </div>
</template>

<script>
import AppSearch from "./AppSearch.vue";
import HeaderActions from "./HeaderActions.vue";
import HeaderLogo from "./HeaderLogo.vue";

export default {
  components: { HeaderLogo, AppSearch, HeaderActions },
  data() {
    return {
      mobileScreen: null
    };
  },
  created() {
    window.addEventListener("resize", this.handlerResize);
    this.handlerResize();
  },
  unmounted() {
    window.removeEventListener("resize", this.handlerResize);
  },
  methods: {
    handlerResize() {
      if (window.innerWidth < 750) {
        this.mobileScreen = true;
      } else this.mobileScreen = false;
```

```
    }
  }
};
</script>
```

Лістинг components/AppFilters.vue:

```
<template>
  <div v-for="item in filterOptions" :key="item.code">
    <app-filter-item :item="item" @setCheck="addFilterOption" />
  </div>
  <button @click="this.$emit('filterProduct', selectedFilterOptions)" class="filter-submit">
    Застосувати фільтри
  </button>
</template>

<script>
// import filterItems from "@/db/filterItems.json";
// import axios from "axios";
import AppFilterItem from "./AppFilterItem.vue";

export default {
  components: { AppFilterItem },
  emits: ["filterProduct"],
  props: ["filterOptions"],
  data() {
    return {
      // filterOptions: [],
      selectedFilterOptions: []
    };
  },
  mounted() {
    // this.filterOptions = filterItems.filters;
  },
  methods: {
    showSelected() {
      console.log(this.selectedFilterOptions);
    },
    addFilterOption(data) {
      if (this.selectedFilterOptions.includes(data.code)) {
        const index = this.selectedFilterOptions.indexOf(data.code);
        if (index > -1) {
          this.selectedFilterOptions.splice(index, 1);
        }
      } else this.selectedFilterOptions.push(data.code);
    },
    setCheck() {
      console.log(this.selectedFilterOptions);
    }
  }
};
</script>
```

Лістинг components/AppHistoryCarousel.vue:

```
<template>
  <div class="history-carousel">
    <h2 class="history-carousel__title">Переглянуті товари</h2>
    <div class="history-carousel__slider-container" ref="sliderContainer">
      <div
        class="carousel__container"
        :style="{
          transform: 'translateX(' + swipeWidth + 'px)'
        }"
        ref="carouselContainer"
      >
        <div
          v-for="item in listWatchedProduct"
          :key="item.id"
          ref="historyCard"
          class="history-card"
          :style="{ 'min-width': widthCard + 'px' }"
        >
          <div class="card-content ">
            <div class="card-content__img ">
              <img :src="item.img" width="100" height="100" :alt="item.img" />
            </div>
            <div class="card-content__title">{{ item.title }}</div>
            <div class="card-content__price">{{ item.price }} грн.</div>
          </div>
        </div>
      </div>
    </div>
    <button @click="scrollBack" class="back-button" value="Back"></button>
    <button @click="scrollForward" class="forward-button" value="Forward"></button>
  </div>
</template>

<script>
import { computed } from "vue";

export default {
  data() {
    return {
      touchStart: 0,
      touchEnd: 0,
      swipeWidth: 0,
      firstSlideOnCarousel: 0,
      widthSliderContainer: computed(() => this.$root.sizeScreen),
      listWatchedProduct: [],
      lastSlide: computed(() => this.listWatchedProduct.length - this.countCardInRow),
      lengthShift: computed(
        () => (this.listWatchedProduct.length - this.countCardInRow) * this.widthCard
      ),
```

```
    countCardInRow: computed(() => {
      if (this.widthSliderContainer < 450) return 1;
      if (this.widthSliderContainer < 800 && this.widthSliderContainer > 450) return 2;
      if (this.widthSliderContainer < 1000 && this.widthSliderContainer > 800) return 3;
      if (this.widthSliderContainer < 1200 && this.widthSliderContainer > 1000) return 4;
      if (this.widthSliderContainer >= 1200) return 5;
      return 5;
    })
  };
},
watch: {
  widthSliderContainer() {
    this.swipeWidth = -this.widthCard * this.firstSlideOnCarousel;
    if (this.swipeWidth <= -this.lengthShift) {
      this.swipeWidth = -this.lengthShift;
    }
  }
},
computed: {
  widthCard() {
    return this.widthSliderContainer / this.countCardInRow;
  }
},
methods: {
  scrollBack() {
    if (this.swipeWidth < 0) {
      this.firstSlideOnCarousel -= 1;
      this.swipeWidth += this.widthCard;
    }
    if (this.swipeWidth >= 0) {
      this.swipeWidth = 0;
      this.firstSlideOnCarousel = 0;
    }
  },
  scrollForward() {
    if (this.swipeWidth >= -(this.widthCard * (this.lastSlide - 1))) {
      this.firstSlideOnCarousel += 1;
      this.swipeWidth -= this.widthCard;
    }
  },
  touchEventStart(e) {
    this.touchStart = e.changedTouches[0].screenX;
  },
  touchEventEnd(e) {
    this.touchEnd = e.changedTouches[0].screenX;
    if (this.touchEnd > this.touchStart) this.scrollBack();
    if (this.touchEnd < this.touchStart) this.scrollForward();
    this.touchStart = 0;
    this.touchtouchEnd = 0;
  }
},
mounted() {
```

```
    if (localStorage.getItem("watch-history") !== null) {
      const temp = JSON.parse(localStorage.getItem("watch-history"));
      for (let i = temp.length - 1; i >= 0; i -= 1) {
        this.listWatchedProduct.push(temp[i]);
      }
    }
    this.$refs.sliderContainer.addEventListener("touchstart", this.touchEventStart);
    this.$refs.sliderContainer.addEventListener("touchend", this.touchEventEnd);
  },
  beforeUnmount() {
    this.$refs.sliderContainer.removeEventListener("touchstart", this.touchEventStart);
    this.$refs.sliderContainer.removeEventListener("touchend", this.touchEventEnd);
  }
};
</script>
```

Лістинг components/FormInput.vue:

```
<template>
  <div class="form-input">
    <input
      type="text"
      :value="modelValue"
      @input="$emit('update:modelValue', $event.target.value)"
      @blur="$emit('blur')"
      @focus="$emit('focus')"
      :class="{
        isValide: isValide,
        isInvalide: isInvalide
      }"
      :placeholder="plaseHolder"
    />
  </div>
</template>

<script>
export default {
  props: {
    modelValue: {
      type: String,
      default: "",
      required: true
    },
    isValide: {
      type: Boolean,
      default: false
    },
    isInvalide: {
      type: Boolean,
      default: false
    },
    plaseHolder: {
```

```
      type: String
    }
  },
  data() {
    return {};
  },
  methods: {
    handleInput(event) {
      this.$emit("input", event.target.value);
    }}};
</script>
```

Лістинг package.json:

```
{ "name": "mda3",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "@wyhaya/vue-slide": "^1.0.2",
    "axios": "^0.21.1",
    "chart.js": "^2.9.4",
    "core-js": "^3.6.5",
    "vue": "^3.0.0",
    "vue-router": "^4.0.0-0",
    "vuex": "^4.0.0-0"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~4.5.0",
    "@vue/cli-plugin-eslint": "~4.5.0",
    "@vue/cli-plugin-router": "~4.5.0",
    "@vue/cli-plugin-vuex": "~4.5.0",
    "@vue/cli-service": "~4.5.0",
    "@vue/compiler-sfc": "^3.0.0",
    "@vue/eslint-config-airbnb": "^5.0.2",
    "babel-eslint": "^10.1.0",
    "eslint": "^6.7.2",
    "eslint-config-prettier": "^8.3.0",
    "eslint-plugin-import": "^2.20.2",
    "eslint-plugin-prettier": "^3.4.0",
    "eslint-plugin-vue": "^7.0.0",
    "sass": "^1.26.5",
    "sass-loader": "^8.0.2"
  }}
```

Всі інші файли можна буде переглянути на магнітному носії, на якому буде архів з готовим проектом.

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
"Розробка веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням фреймворку Vue.js."
студента групи 121-18ск-1 Бершавського Івана Олександровича

Керівник економічного розділу
Зав. каф. ПЕП та ПУ, д.е.н **Вагонова О.Г.**

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Бершавський.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Бершавський.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Бершавський.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Бершавський.ppt	Презентація кваліфікаційної роботи