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 *Міщенка Юрія Юрійовича*
(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 бібліотеки React*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*проф. Вагонова А.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) *Міщенка Юрія Юрійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого програмного забезпечення платформи Market Dynamics Analyzer для моніторингу динаміки цін з використанням бібліотеки React*

затверджена наказом ректора НТУ «ДП» від *07.06.2021 р.* № *317-с*

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) *Міщенко Ю.Ю.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

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

РЕФЕРАТ

Пояснювальна записка: 85 с., 22 рис., 3 дод., 26 джерел.

Об'єкт розробки: платформа Market Dynamics Analyzer для моніторингу динаміки цін.

Мета кваліфікаційної роботи: створення платформи Market Dynamics Analyzer для моніторингу динаміки цін необхідних товарів. Веб-застосунок має бути, сумісний з десктопними та мобільними пристроями, який буде надавати доступ до платформи, адаптованої для користувачів, бажаючих отримати інформацію про минулі та поточні ціни на необхідний товар.

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

У першому розділі проводиться дослідження предметної галузі та існуючих рішень, визначається актуальність завдання та призначення розробки, виконується постановка завдання, задаються вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі обирається платформа для розробки, виконується проектування додатку і його розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічного обладнання, описується виклик та завантаження програми, 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 програмного додатку який дасть змогу користувачам, переглядати динаміку змін цін необхідного товару у зручному форматі. Для звичайних користувачів це може стати корисним тим, що допоможе знайти найкращу пропозиція серед потрібних товарів. А для деяких комерційних організацій може стати в нагоді для отримання необхідної інформації яка допоможе в прийняті правильного рішень щодо формування власної цінової політики.

Актуальність програмного продукту визначається тим, що у сучасному світі широко використовується моніторинг цін конкурентів. Він є невід’ємною частиною розвитку комерційних організацій.

Список ключових слів: МОНІТОРИНГ ЦІН, ПРОГРАМНИЙ ПРОДУКТ, БРАУЗЕР, КЛІЄНТ, ІНФОРМАЦІЙНА СИСТЕМА, ЕОМ, ПРИСТРІЙ, ТОВАР, КОРИСТУВАЧ, ШАБЛОН ПРОЕКТУВАННЯ, ФРЕЙМВОРК.

ABSTRACT

Explanatory note: 85 pages, 22 figures, 3 appendix, 26 sources.

Object of development: Market Dynamics Analyzer platform for monitoring price dynamics.

The purpose of the qualification work: to create a platform Market Dynamics Analyzer to monitor the dynamics of prices of necessary goods. The web application should be, compatible with desktop and mobile devices, which will provide access to a platform adapted for users who want to get information about past and current prices of the required goods.

In the introduction, we analyze the current state of the problem, specify the purpose of the qualification work and its scope, given, justified the relevance of the topic and clarifies the problem statement.

The first chapter conducts a study of the subject area and existing solutions, determines the relevance of the problem and the purpose of development, you perform a problem statement, you specify the requirements for software implementation, technology and software tools.

In the second section, the platform for the development is selected, the design of the application and its development is performed, the description of the algorithm and structure of the system operation, the input and output data are determined, the characteristics of the composition of the technical equipment, the call and download the program is described, the program operation is described.

In the economic section, the labor intensity of the developed software product is determined, the cost of the work to create the application is calculated and the time for its creation is calculated.

The practical value lies in the development of software application, which will allow users to view the changes in price of the desired goods in a convenient format. For ordinary users it can be useful in that it will help to find the best offer among the necessary goods. And for some commercial organizations may be useful for obtaining the necessary information, which will help in making the right decision on the formation of its own pricing policy.

Relevance of the program product is determined by the fact that in today's world is widely used to monitor the prices of competitors. It is an integral part of development of commercial organizations.

Keywords: PRICE MONITORING, SOFTWARE, BROWSER, CLIENT, INFORMATION SYSTEM, COMPUTER, DEVICE, PRODUCT, USER, DESIGN PATTERNS, FRAMEWORK.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс для програмування застосунків;

ЕОМ – електронно-обчислювальна машина;

RRP – рекомендована роздрібна ціна;

SaaS – програмне забезпечення як послуга;

ІС – інформаційна система;

ООП – об'єктно-орієнтоване програмування;

ОС – операційна система;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

MVC – модель-уявлення-контролер;

CMS – система керування вмістом;

BSD – дистрибутив програм Берклі;

DOM – об'єктна модель документа;

JSX – синтаксис XML в JavaScript;

FP – функціональне програмування;

PWA – прогресивна веб-програма;

DVCS – системи управління розподіленими версіями;

URL – уніфікований покажчик ресурсу;

IDE – інтегроване середовище розробки;

ІТ – інформаційні технології.

ВСТУП

В сучасному світі широко використовується моніторинг цін конкурентів. Він є найважливішою частиною розвитку Інтернет-магазину. Моніторинг цін - це бізнес-процес, відомий всім роздрібним торговцям з давніх часів. До активного розвитку онлайн-торгівлі, працівники роздрібних компаній відвідували торгові точки та запам'ятовували або записували ціни на товари на полицях. Повернувшись до свого магазину або торгової точки, вони об'єднали моніторинг цін у звіти та проаналізували результати наступного зниження цін разом із керівниками своїх компаній.

Цей метод мав і свої недоліки. Сам моніторинг цін на товари зайняв багато часу, а також потрібен час, щоб перевести результати моніторингу роздрібних цін у формати, які було легко проаналізувати. Потрібні були години, щоб приймати рішення щодо скидок цін і міняти їх у своїх магазинах. Із часом конкуренти вносили нові зміни у свої ціни, і все повторювалося б по колу.

Усе це змінилося з розвитком Інтернет-торгівлі та переміщенням моніторингу цін конкурентів в Інтернеті. Зараз працівники різних компаній здійснюють відстеження цін в мережі Інтернет. Вони отримують дані про ціни та акції конкурентів у режимі реального часу і більше не можуть обійтися без автоматичних інструментів моніторингу цін у мережі. Це дає змогу за кілька хвилин визначити цінову політику конкурентів, прийняти рішення щодо власного ціноутворення на основі реального перерізу ринкових цін, динамічно змінювати ціни на свою продукцію на основі моніторингу запасів конкурентів.

Моніторинг цін конкуруючих Інтернет-магазинів є обов’язковою процедурою для кожного учасника електронної комерції. Він складається з вибору сайтів-конкурентів у різних товарних категоріях та систематичного порівняння цін власного сайту з цінами конкурентів. Метою цієї процедури є коригування власних цін магазину з метою максимізації прибутку від продажу товарів.

Програми аналізу були першими рішеннями проблем у цій галузі, і їх можливості не відповідають сучасним потребам представників електронної комерції. Ось чому їх замінили сучасні хмарні сервіси SaaS. Вони характеризуються більшою швидкістю, точністю порівняння продуктів, а головне, наявністю потужних функціональних можливостей для аналізу зібраних даних [3]. Використання таких послуг дає змогу збільшити продажі та прибуток. Кількість інтернет-магазинів зростає, як гриби після дощу. Тому моніторинг цін конкурентів є одним з найактуальніших завдань проектів електронної комерції.

Автоматичний збір цін (парсинг цін) вважається найпростішим та найзручнішим способом збирання маркетингової інформації про ціни партнерів та конкурентів. Щоб налаштувати сам синтаксичний аналізатор та обробити дані аналізу ціни, вам точно знадобиться відданий працівник. Він витратить менше часу, ніж ви витратили б на збір цін вручну, але йому доведеться регулярно перевіряти дані аналізу цін та керувати алгоритмами їх збору.

Парсинг цін на веб-сайтах - це автоматизований процес збору цін, рекламних акцій та доступності продуктів з різних веб-сайтів партнерів та конкурентів в Інтернеті. Цю роботу може виконати кваліфікований менеджер, здатний поєднувати цінові рішення та вміння самостійно налаштовувати та аналізатор цін. Процес буде ефективним, коли ваш асортимент буде відносно стабільним, а Інтернет-ресурси ваших клієнтів та конкурентів не застосовують ефективний захист від автоматичного розбору цін.

Виходячи з вищеописаної інформації, було прийнято рішення розробити платформу Market Dynamics Analyzer для моніторингу динаміки цін в Інтернеті. Дане програмне забезпечення дасть змогу користувачам, переглянути зміну цін необхідного товару у зручному вигляді.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної області

Моніторинг цін (monitoring price) конкурентів в Інтернеті - це регулярне та систематичне спостереження за станом актуальних цін з метою їх оцінки, аналізу тенденцій та розуміння конкурентного середовища. Збір цін необхідний для ефективного ціноутворення. Це призводить до швидкого коригування оптових та роздрібних цін для компаній, партнерів та клієнтів[1].

Аналіз моніторингу цін - це процес прийняття рішення щодо ціноутворення компанії на основі даних про цінову ситуацію конкурентів, пропозицій постачальників, розуміння вартості та норми прибутку внутрішніх процесів та ринкових цілей самої компанії. Для управління цим процесом використовуються складні математичні моделі та машинне навчання на основі штучного інтелекту [2].

Для розбору цін на веб-сайті розроблені спеціальні програми - аналізатори цін, які написані різними мовами програмування. Вони автоматично обходять веб-сайти ваших партнерів та конкурентів за розкладом та збирають інформацію про ціни, знижки, акції та наявність там продуктів. Синтаксичні аналізатори можуть сканувати сотні веб-сторінок, тонко налаштовувати дані та упаковувати їх у правильний формат презентації для ваших цінових рішень. У той час, коли сайти конкурентів та партнерів розмножуються дуже швидко, аналізатори цін можуть замінити роботу цілого відділу моніторингового персоналу.

Деякі з найвідоміших ресурсів для моніторингу цін: Marketparser, uXprice, Zoomos, Catalogloader та Pricecop. Але існують десятки, якщо не сотні інших варіантів. Більшість ресурсів мають такі характеристики:

- звіти з можливостями фільтрації та сортування даних;
- збереження історії зміни цін;

- можливість побудови діаграм та графіків;
- інтеграція API;
- надсилання повідомлень про необхідність зміни цін;
- обробка прайс-листів в різних валютах;
- конкурентна аналітика;
- автоматичне оновлення даних ;
- аналіз отриманих даних про ціни та утворення рекомендованої ціни.

Для аналізу можливостей сучасних інструментів моніторингу цін варто розглянути існуючі аналоги та виявити їх переваги та недоліки. Для дослідження були обрані такі відомі проекті як uXprice, Marketparser та Zoomos.

uXprice - це хмарна послуга SaaS для моніторингу цін конкурентів в Інтернеті (рис. 1.1), тому з нею легко і зручно працювати. Щоб почати відстежувати ціни, вам потрібно зареєструватися та завантажити свою продукцію. Є 2 способи зробити це:

1) Вказуйте посилання на товари по одному - скільки товарів, стільки посилань. Це не швидко, але кожен може це зробити;

2) Вкажіть посилання на товарний фід. У цьому випадку завантаження буде масовим і швидким [5].

Товарний фід - це спеціальний файл із даними про всі товари та їх атрибути. Як правило, він створюється для розміщення реклами на різних рекламних платформах. Це може бути фід Google Merchant, Facebook або Яндекса. Підтримувані формати файлів: XML, CSV, YML, TSV, XLS, XLSX.

Пошук конкурентів та порівняння вашої продукції з продуктами конкурентів відбувається автоматично. Але існує також ручне управління на етапі додавання конкурентів до списку для їх відстеження або видалення.

Послуга моніторить лише рекламовані товари, що виключає можливість потрапляння в статистику не поточних цін на товари (не оновлені ціни або ціни з і без того мало відвідуваних, "занедбаних" інтернет-магазинів). За

результатами моніторингу вся статистика структурована та доступна у вашому особистому кабінеті.

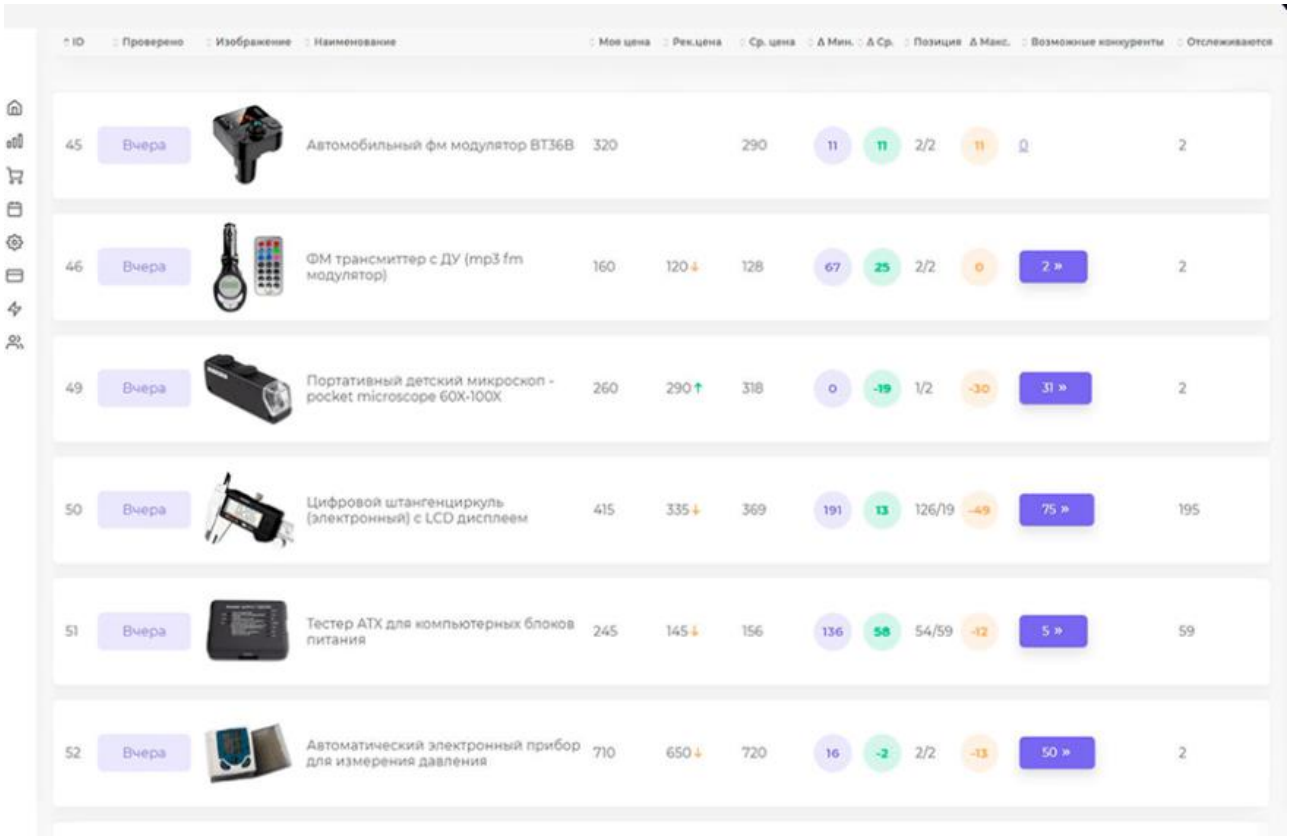

Рис. 1.1. Сторінка платформи uXprice

Зокрема, для кожного товару будуть доступні такі показники:

– кількість конкурентів, що рекламують один і той же товар в Інтернеті;
– ваша цінова позиція;
– середня ціна на ринку;
– мінімальні та максимальні ціни на ринку;
– різниця від середньої ціни у відсотках;
– рекомендована ціна.

Для розрахунку рекомендованої ціни ви можете скористатися стандартною формулою розробників послуг або змінити її відповідно до власних потреб. Існує також сортування за категоріями та загальна статистика усього асортименту у вигляді графіків.

Переваги:

– автоматичне порівняння товарів;

– актуальність даних (відстежуються лише рекламовані продукти);

– Фіксована вартість 1 чека;

– наявність аналітики та API;

– є рекомендації щодо ціни.

Недоліки. Щоб скористатися послугою, потрібно надати посилання на фід, який є не у всіх.

Marketparser - Це проект аналізу інформації з різних сайтів (рис. 1.2), включаючи ринки та агрегатори цін. Синтаксичний аналізатор регулярно контролює постійний список сайтів, а потім надає своїм клієнтам дані з цієї бази даних. За необхідності сайти можна додати до цього списку, залишивши запит розробникам. Час налаштування становить в середньому два дні.

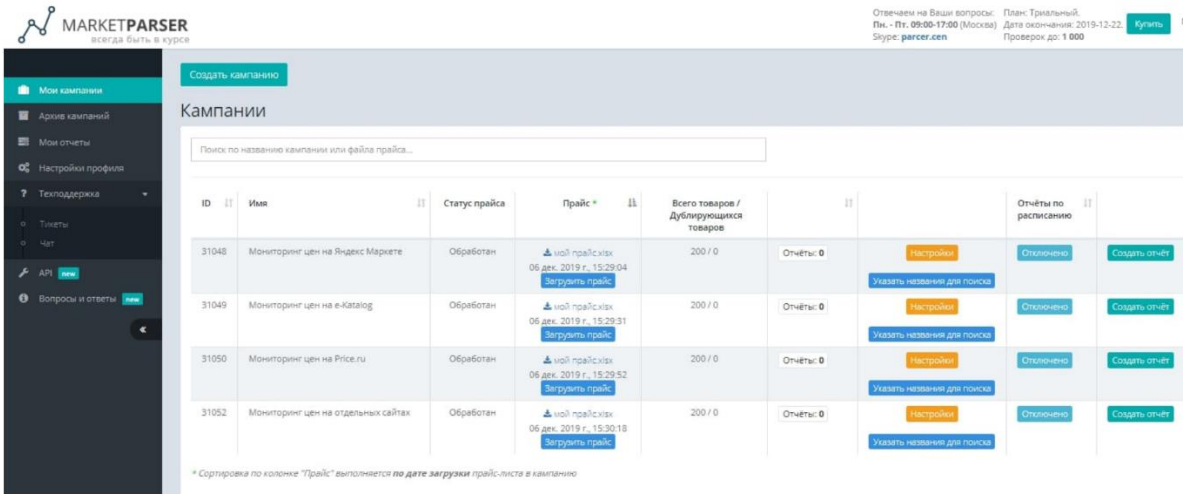

Рис. 1.2. Сторінка платформи Marketparser

Крім того, творці платформи обіцяють функціональність для автоматичного оновлення ціни та доступності. Marketparser також допоможе вам зібрати інформацію з описів товарів на сайтах конкурентів.

Гордістю цієї послуги є висока швидкість пошуку та обробки даних. І це насправді великий плюс, оскільки багато служб з моніторингу цін конкурентів дуже повільні.

Для моніторингу цін за допомогою цієї послуги потрібно надати перелік своїх товарів, який буде використовуватися для пошуку конкурентів. На основі результатів та порівняння із закупівельними цінами ви отримуєте оптимальну роздрібну ціну для продажу. За бажанням його можна негайно завантажити на ваш сайт або в агрегатор цін.

Особливість: моніторинг проводиться окремими містами (кількість міст буде залежати від ціни).

Zoomos - це хмарна послуга SaaS, яка пропонує декілька видів послуг: наповнення вашого сайту продуктами (фотографії, опис, ціни), відстеження RRP для дистриб'юторів та моніторинг цін конкурентів в інтернет-магазинах (рис. 1.3). Послуга допоможе інтернет-магазинам автоматизувати націнку своїх товарів залежно від оптових постачальників та цін конкурентів.

Щоб почати працювати в сервісі Zoomos, потрібно завантажити прайс-лист, а потім прайс-листи своїх постачальників. Мерчендайзинг здійснюється автоматично шляхом порівняння двох прейскурантів. Потім сервіс генерує файл цін та оновлення доступності для магазину та налаштовує правила формування цін.

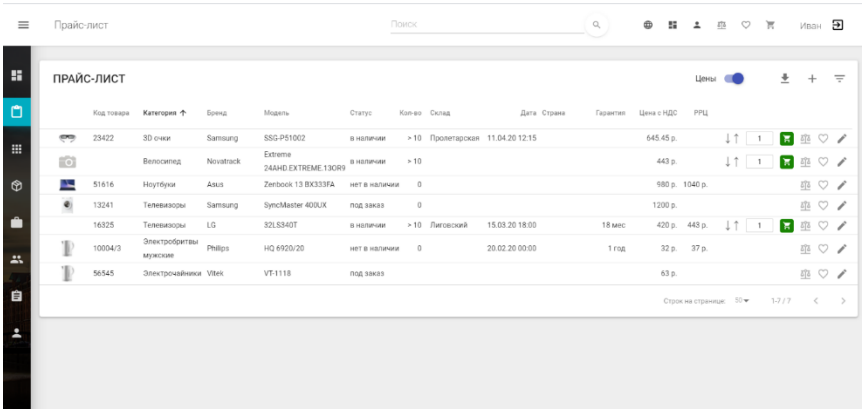

Рис. 1.3. Сторінка платформи Zoomos

Переваги:

- автоматичне оновлення;
- автоматичне вивантаження товарів на ринки;
- необмежена кількість продуктів для аналізу;
- обробка будь-яких типів прайс-листів в різних валютах;
- доступність програми скрізь, де є Інтернет.

З недоліків. Часто постачальників більше одного, тому існує дуже багато прайс-листів, і не всіх товарів, які їм потрібні. Тоді необхідно сформувати загальний прайс-лист для всіх постачальників. У цьому випадку ціни у вашому прейскуранті та прейскуранті постачальників повинні кожного разу оновлюватися вручну).

Після розгляду вище представлених проектів можна сказати, що програмні продукти зі схожими властивостями в час активної діяльності інтернет-магазинів є дуже потрібними. Ефективність онлайн-продажів безпосередньо залежить від конкурентоспроможних цін, тому відстеження цін інших продавців є завданням першочергового значення. Найкращий спосіб зробити це оперативно і отримати точні дані - скористатися послугами аналізу чи послугами SaaS.

1.2. Призначення розробки та галузь застосування

Інформаційна система, що виконана для кваліфікаційної роботи, має назву «Платформа для моніторингу динаміки цін в Інтернеті яка базується на використані парсингу».

Моніторинг та аналіз цін використовується в тих випадках, коли в рішенні про покупку є декілька відповідних за потрібними критеріями та рівних за ціною варіантів.

Моніторинг цін був і завжди буде невід'ємною і важливою частиною будь-якого бізнесу. Це не залежить від галузі, в якій працює ваша компанія - будь то роздрібна торгівля, виробництво, дистрибуція або навіть індустрія подорожей.

Безліч компаній збільшили виручку і прибуток завдяки використанню інструменті сервісів моніторингу цін.

В даний час ціновий аналіз зазвичай є найкращим методом аналізу для вибору варіантів цін на продукт. В рамках цієї концепції ціна продуктів або послуг однієї компанії порівнюється з іншими продуктами, які можуть бути альтернативою. Наприклад, якщо на певний проект подали заявки або пропозиції 7 конкурентів, аналіз цін включатиме детальний огляд переваг, які може забезпечити кожен продукт / послуга, виходячи з їх запропонованої ціни [4].

Призначення розробки кваліфікаційної роботи є реалізація програмного додатку Market Dynamics Analyzer для моніторингу динаміки цін в Інтернеті. Дане програмне забезпечення дасть змогу користувачам, перегляду зміни ціни необхідного товару у зручному форматі. Для звичайних користувачів це може стати корисним тим, що допоможе знайти найбільш вигіднішу пропозиція серед цікавивших його товарів. А для деяких інтернет-магазинів може стати в нагоді для отримання необхідної інформації яка допоможе в прийняті правильного рішень щодо формування власної цінової політики.

1.3. Підстава для розробки

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

Тема роботи погоджується з головою проекту, випускаючою кафедрою, та затверджується наказом ректора.

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

– освітня програма 121 «Інженерія програмного забезпечення»;

– освітня програма та план навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;

– завдання на кваліфікаційну роботу на тему «Проектування та розробка платформи Market Dynamics Analyzer для моніторингу динаміки цін. Розробка веб-додатку з використанням бібліотеки React».

1.4. Постановка завдання

Метою проекту є створення платформи Market Dynamics Analyzer для моніторингу динаміки цін необхідних товарів. Веб-застосунок має бути, сумісний з десктопними та мобільними пристроями, який буде надавати доступ до платформи, адаптованої для користувачів, бажаючих отримати інформацію про минулі та поточні ціни на необхідний товар.

Поставлена мета може бути досягнута при виконанні наступних вимог:

– вивчення предметної області розв’язуваного завдання;

– проведення порівняльної характеристики можливостей аналогічних застосунків;

– вибір доречної архітектури програми;

– розробка структур вхідних і вихідних даних для проектованого програмного забезпечення;

– написання програмного коду проекту;

– розробка довідника для пояснення користування додатком.

Кінцева програма має представляти собою веб-застосунок, який буде працювати у веб-браузері як десктопному, так і мобільному варіанті.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт має містити наступних функціональні характеристики.

Функціональні можливості «Гостя» повинні містити наступні функції:

– фільтрація та сортування товарів;

– пошук за необхідними критеріями;

– можливість перегляду графіку цін товарів та їхньої середньої вартості;

– перегляд каталогу товарів.

Функціональні можливості «Користувача» повинні містити наступні функції, а також всі функціональні можливості які представлені Гостю:

– перегляд збережених товарів;

– додання категорій які зацікавили користувача.

1.5.2.Вимоги до інформаційної безпеки

Для забезпечення надійного функціонування системи необхідно реалізувати наступні вимоги:

– валідація введених даних: перевірка на відповідність типів, на введення обов’язкових полів даних;

– обробка рідкісних ситуацій;

– виведення сповіщень при виникненні помилок;

– цілісність даних;

– конфіденційність інформації;

– доступність інформації для всіх авторизованих користувачів.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для користувача є важливим мати систему, спроможну запускати та працювати з сучасними браузерами. Тож необхідно мати такі мінімальні параметри для комп’ютера чи ноутбука :

– операційна система [OS]: Windows XP з пакетом оновлень 2 + Windows Vista Windows 7 Windows 8 Windows 10;

– ЦП [CPU]: Intel Pentium 4 / Athlon 64 або більш пізньої версії з підтримкою SSE2;

– відеоадаптер [GPU]: 3D адаптер nVidia, Intel, AMD/ATI;

– звукова карта DirectX Compatible;

– відеопам’ять [VRAM]: 64 МБ;

– накопичувач [HDD]: 350 МБ;

– оперативна пам’ять [RAM]: 512 МБ.

Для користувача на платформі Android, слід дотримуватися таким технічним вимогам:

- операційна система Android 5.0 (Lollipop) або вище.

1.5.4. Вимоги до інформаційної та програмної сумісності

Інформаційна система вимагатиме від користувача середовища з такими компонентами:

– одна з операційних систем: Windows, Linux, Mac OS, Android, IOS;

– для користування на десктопній системі – один з існуючих браузерів (Google Chrome, Opera, Mozilla Firefox) для Android чи IOS - мобільний браузер;

– постійний доступ до мережі Інтернет.

Користувацький інтерфейс має бути створений за допомогою бібліотеки ReactJS. React - це бібліотека JavaScript, розроблена Facebook в 2013 році, яка відмінно підходить для створення сучасних односторінкових додатків будь-якого розміру і масштабу [6].

Дана бібліотека була обрана для розробки так як вона має компонентно-орієнтований підхід, можливість легко модифікувати існуючі компоненти та повторно використовувати код перетворюють розробку React на постійний процес вдосконалення. Компоненти, створені під час роботи над певним проектом, не мають додаткових залежностей. Таким чином, ніщо не заважає використовувати їх знову і знову в різних типах проектів. Весь попередній досвід

можна легко застосувати під час роботи на новому веб-сайті або навіть під час створення мобільного додатка.

Для розробки програмного продукта необхідна наявність наступних програм та систем:

- Node JS (v8);
- один з існуючих браузерів (Google Chrome, Opera, Mozilla Firefox);
- Visual studio code.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути веб-застосунок, сумісний з десктопними та мобільними пристроями, який буде надавати доступ до платформи для моніторингу динаміки цін, адаптованої для користувачів, бажаючих отримати інформацію про минулі та поточні ціна на необхідний товар.

Основне призначення додатку:

– надання можливості перегляду користувачам графіків цін на товари з різних сайтів за вказаний період та перегляд їхньої середньої вартості;

– надання можливості для зареєстрованих користувачів зберігати в профіль вподобані ним товари та категорії товарів які цікавлять користувача в даний час.

Розроблювана програма має дотримуватися наступних функціональних характеристик:

– фільтрація та сортування товарів;

– пошук за необхідними критеріями;

– можливість перегляду графіків цін на товари;

– перегляд каталогу товарів.

– перегляд збережених товарів;

– додання категорій які зацікавили користувача.

Для досягнення поставленої задачі розроблюваний продукт має підтримувати такі операції:

– внесення і редагування інформації, щодо збережених категорій та товарів користувача;

– авторизація і аутентифікація користувача задопомогою форми входу та використання токену автентифікації.

2.2. Опис застосованих математичних методів

Під час проектування та розробки данного програмного продукту використовувалися лише прості арифметичні дії для розрахунку середньої вартості товару серед обраних магазинів. Інші математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Шаблон проектування при розробці програмного забезпечення - це повторювана архітектурна конструкція, яка представляє вирішення проблеми архітектури в якомусь часто вживаному контексті.

Зазвичай шаблон - це не повністю готовий продукт, який можна безпосередньо перетворити в код; це лише приклад рішення проблеми, який можна використовувати в різних ситуаціях. Об'єктно-орієнтовані шаблони показують взаємозв'язки та взаємодії між класами або об'єктами, не визначаючи, які кінцеві класи або об'єкти програми будуть використовуватися.

Для даного програмного додатку було обрано архітектуру Model-View-Controller (MVC).

Шаблон проектування MVC (рис. 2.1). став основою для архітектурного рішення першого середовища програмування з графічним інтерфейсом користувача - Smalltalk-80. Вперше MVC був описаний норвежцем Тригве Рінскаугом, який деякий час працював у лабораторії Xerox PARC. Впровадження шаблону в Smalltalk-80 було описано дещо пізніше Стівом Бербеком [10].

Шаблон проектування MVC передбачає поділ даних програми, користувальницького інтерфейсу та логіки управління на три окремі компоненти: модель, вигляд та контролер - так що кожен компонент може бути змінений незалежно. Модель (Model) надає дані перегляду та реагує на команди контролера, змінюючи свій стан. View відповідає за відображення моделі

користувачеві, реагування на зміни в моделі. Контролер інтерпретує дії користувача, попереджаючи модель про зміну.

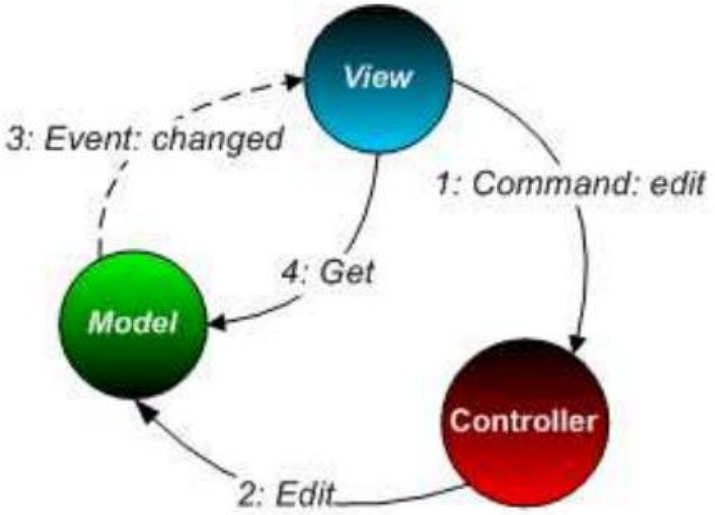

Рис. 2.1. Шаблон проектування MCV

Контролери - це модулі, що містять обробники маршрутів, функції, що приймають стандартні експрес-параметри запиту та відповіді. Контролери мають логіку для перевірки даних та базової обробки запитів, але вони не отримують безпосереднього доступу до бази даних і не виконують рутинних операцій, для цього використовуються сховища та супровідники.

Сховища - тут розташована логіка доступу до бази даних та додаткова бізнес-логіка, частина бізнес-логіки може знаходитися в контролері, але сховища переважніше для цього.

Помічники - помічники виконують певні операції - надсилання електронних листів, обробку помилок, реєстрацію тощо.

Маршрутизатори - зіставлення маршрутів (URL-адрес) з відповідними обробниками з контролерів. У цьому випадку не роблять особливої різниці між маршрутами для методів API та маршрутами для веб-сторінок (переглядів).

Завдання - скрипти службових програм для таких завдань, як створення початкової бази даних або імпорт даних із файлу.

Представлення - подання сервера (HTML-шаблон) підтримує подання макета та часткові подання. У Contoso подання сервера використовуються для сторінок автентифікації, інші подання HTML генеруються на клієнті [11].

Bootstrap - це фреймворк CSS / HTML для побудови веб-сайтів. Іншими словами, це набір інструментів для побудови веб-макета. Він має ряд переваг, що робить його найбільш популярним серед інших подібних фреймворків [12]. Переваги Bootstrap:

– швидкість роботи - створення макетів за допомогою Bootstrap займає менше часу завдяки великому набору готових до використання елементів;

– гнучкість - додавання нових елементів не порушує загальної структури сітки, що динамічно змінюється;

– простота модифікації - стилі редагування досягаються додаванням нових правил CSS, які замінюють уже створені. У цьому випадку вам не потрібно використовувати такі атрибути, як! Important;

– велика кількість шаблонів;

– величезна спільнота прихильників / розробників;

– широкий спектр використання - Bootstrap використовується для створення тем для майже будь-якої CMS (Magento, Joomla, WordPress або будь-якої іншої), включаючи цільові сторінки на одній сторінці;

– зрозуміла офіційна документація.

Якщо ви оберете Bootstrap, цей фреймворк дозволить вам заощадити багато часу на розробку інтерфейсу завдяки великій кількості готових компонентів. Тут слід звернути увагу, що Bootstrap - це, так би мовити, набір із трьох фреймворків: CSS / HTML, компонентів JS та шрифтового знака.

2.4. Опис використаних технологій та мов програмування

Дана інформаційна система була розроблена з використанням таких технологій:

– JavaScript;

– React JS.

Для правильного функціонування розробленого програмного додатку необхідна відповідна операційна система.

Операційна система, або коротше ОС – це основний набір програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини; вона забезпечує управління обчислювальним процесом та організовує взаємодію з юзером. Операційна система зазвичай складається з ядра операційної системи та основного набору прикладних програм.

Сучасні операційні системи мають графічний інтерфейс користувача, який крім клавіатури також використовує вказівниковий пристрій - миша або тачпад. Старіші системи, та системи, не призначені для частої безпосередньої взаємодії з користувачем (приклад сервери) використовують інтерфейс командного рядка. Будь-який з цих підходів тим чи іншим чином реалізують оболонку, яка перетворює користувацькі команди - текстові з клавіатуриабо переміщення миші - у системні виклики. При виборі ОС ключовим моментом є архітектура комп'ютера зокрема центрального процесора, на якому вона буде працювати. На персональних комп'ютерах сумісних з IBM PC запускаються ОС сімейства «Microsoft Windows», «Linux» і «BSD».

Windows – загальна назва операційних систем для ЕОМ, розроблених корпорацією Microsoft. Перші версії були не повноцінними операційними системами, а лише оболонками для ОС MS-DOS.

Пакет Microsoft Windows складається з стандартних програми, таких як браузер Internet Explorer, поштовий клієнт Outlook Express, медіапрогравач Windows. За допомогою технологій COM і OLE їх компоненти можуть бути

вбудовані в інші застосунки, включаючи й сторонніх виробників. Ці продукти безкоштовні і їх можна завантажити з офіційного сайту Microsoft, але для встановлення деяких з них потрібно мати діючу ліцензію Windows. Запуск цих програм в інших операційних системах можливий за допомогою емуляторів Windows. Існують також версії деяких з них, спеціально розроблені для інших операційних систем. Ці версії поступаються оригінальним версіям за набором функцій і можливостей, а також кількістю оновлень[7].

Linux – загально прийнята назва операційних систем, UNIX подібний систем, заснованих на однойменному ядрі.Це один із найвдаліших прикладів розробки програмного забезпечення з вільним і відкритим кодом (open source). На відміну від власницьких операційних систем (таких як Microsoft Windows та MacOS X), їх вихідні коди є загальнодоступним для всіх, хто його може використовувати, змінювати та розповсюджувати (безкоштовно). Linux, спочатку був створений для використання окремими ентузіастами на своїх персональних комп'ютерах, згодом, завдяки підтримці таких компаній, як IBM, Sun Microsystems, HP, Novell та інших, він став дуже поширеним, як серверна операційна система.

Linux перенесено на значну кількість апаратних платформ. В даний час він досить успішно застосовується на мейнфреймах та суперкомп’ютерах, а також вбудований у багаттох інших пристроях (планшетні ПК, смартфони, маршрутизатори комп'ютерних мереж (роутери), системи управління телевізорами, пристрої автоматики та ігровою консоллю тощо)[8].

JavaScript був представлений в 1995 році як прийом додавання програмного коду на веб-сторінки в браузері Netscape Navigator. Основним завдання при створенні цього інструменту було створити «мову для склеювання» зіставних частин веб-застосунку.

JavaScript – це мова сценаріїв для Інтернет-браузерів. «Це інтерпретована мова, вона не потребує компілятору для перекладу коду у машинний, як то є в C або C++. JavaScript-код працює безпосередньо у веб-браузері [19]».

Вона зробила можливими сучасні веб-додатки, з якими можна взаємодіяти безпосередньо, без перезавантаження сторінки для кожної дії. JavaScript також використовується в більш традиційних веб-сайтах для забезпечення різних форм інтерактивності і інтелектуальності.

Важливо відзначити, що JavaScript не має майже нічого спільного з мовою програмування Java. Схожу назву було навіяно швидше маркетинговими міркуваннями, ніж здоровим глуздом. Коли JavaScript був представлений мова Java активно рекламувалася і набирала популярність.

Хтось подумав, що було б непогано спробувати скористатися цим успіхом. Після того як мову було прийнято за межами Netscape, був написаний стандартний документ. Для опису того, як повинна працювати мова JavaScript, щоб різні частини програмного забезпечення, які стверджували, що підтримують JavaScript, могли працювати разом. Цей документ називається стандартом ECMAScript, в честь організації Ecma International, яка провела стандартизацію. На практиці терміни ECMAScript і JavaScript можна використовувати як взаємозамінні - це дві назви однієї і тієї ж мови.

Є ті, хто говорить жахливі речі про JavaScript. Багато з них це правда. Але тут є реальна проблема: JavaScript поблажливий в тому, що він дозволяє. Ідея цієї конструкції полягала в тому, що вона зробить програмування на JavaScript простіше для початківців. Насправді, в більшості, це робить пошук проблем в ваших програмах, тому що система не буде вказувати вам на них [24].

Однак така гнучкість має і свої переваги. Вона залишає місце для багатьох технік, які неможливі в більш жорстких мовах.

Веб-браузери - не єдині платформи, на яких використовується JavaScript. Деякі бази даних, такі як MongoDB і CouchDB, використовують JavaScript в якості мови сценаріїв і мова запитів. Кілька платформ для настільного і серверного програмування, в першу чергу проект Node.js , надають середовище для програмування на JavaScript поза браузера [13].

Причини вибору мови програмування JavaScript.

Простота: Здебільшого бібліотеки в Node мають прості API, які легко зрозуміти та працювати інтуїтивно зрозуміло. Якщо одна з бібліотек з якихось причин не придатна для розробки, можна знайти заміну, оскільки їх величезна кількість.

Контроль: Програміст сам будує інфраструктуру проекту, вибираючи та комбінуючи невеликі модулі для конкретних завдань. Для цього потрібно більше часу, але результат того вартий. Як тільки ви це зрозумієте, можна легко застосувати досвід, отриманий у майбутньому.

Універсальність: JavaScript спочатку працював лише на клієнті. Спочатку він разом з Node перемістився на сервер, а нещодавно стало можливим успішно писати настільні (Eletctron) та мобільні програми. Більше того, для мобільних додатків є опція гібридних додатків (з використанням обгортки над браузером (Cordova)) або додатків з власним інтерфейсом (ReactNative, NativeScript). Для Node існує величезна різноманітність бібліотек, і його легко інтегрувати з іншими технологіями, базами даних, хмарними технологіями, різними форматами та протоколами, ви знайдете все.

Простота розгортання: Node дуже легко розгорнути на сервері: на Linux, а також на Windows. Після багатьох років роботи з .NET розгортання для мене щоразу було неприємним досвідом, з Node це навіть приємно. Це просто те, що ти повинен спробувати.

Продуктивність: Вузол є асинхронним і не блокує виконання під час тривалих операцій, таких як коректура файлу або доступ до бази даних. Це забезпечує високий рівень продуктивності в середовищі з одним різьбленням. З іншого боку, обчислення в JavaScript відбувається повільніше, ніж у статично набраних мовах. Для більшості проектів це не проблема. Якщо вам потрібно зробити обчислення, а не просто перетворення даних, краще написати окремий сервіс у чомусь іншому.

Одна мова на сервері та клієнті: Це зручно, оскільки дозволяє легко переносити код між клієнтом та сервером, простіше у розробці та обслуговуванні [22].

JavaScript використовують з HTML та CSS для створення веб-додатків або веб-сторінок. JavaScript підтримується більшістю сучасних веб-браузерів, таких як Google Chrome, Firefox, Safari, Microsoft Edge, Opera тощо. Більшість мобільних браузерів для Android та iPhone мають підтримку «JavaScript. JavaScript контролює динамічні елементи веб-сторінок. Він працює у веб-браузерах, а останнім часом і на веб-серверах».

Існують три компоненти, які створюють веб-сайти та програми: мова розмітки гіпертексту (HTML), таблиці каскадних стилів (CSS) та JavaScript. Кожен з них має роль у створенні веб-програми.

«HTML – це мова розмітки гіпертексту, яка створює скелет веб-сторінки. Всі абзаци, розділи, зображення, заголовки та текст записані в HTML. Вміст з’являється на веб-сайті в порядку, в якому вони написані в HTML ».

«CSS контролює стиль та додаткові аспекти компонування. CSS використовується для створення дизайну веб-сайту, задання кольорів, шрифтів, стовпців, границь тощо».

В чому різниця між CSS і HTML? HTML використовується для структурування вмісту сторінки. CSS використовується для форматування цього структурного вмісту.

Третій елемент – JavaScript. Він створює динамічну активність у веб додатку клієнта. Сценарій у JavaScript – це програмний код, що частіше за все керує функціями при натисканні кнопок, валідацією форм, медіа тощо [21].

Веб-браузер завантажує веб-сторінку, аналізує розмітку HTML і створює з вмісту те, що відоме як Document Object Model (DOM). DOM представляє собою інтерфейс, що дозволяє програмам і скриптам отримати доступ до вмісту HTML веб-сторінки в реальному часі.

ReactJS – це бібліотека Javascript, яка була розроблена Facebook для розробки інтерфейсу користувача в 2013 році. Вона використовується у розробці адаптивних односторінкових застосунків. DOM(Об'єктна модель документа) веб-сторінки – це те, на що спрямований ReactJS. React не оновлює всю веб-сторінку, натомість лише оновлює частину, яка змінюється, що значно підвищує продуктивність застосунку.

Для оптимізації оновлення змін в об’єктній моделі ReactJS використовує віртуальний DOM. «Віртуальний DOM – це легка копія реального DOM, що має всі його властивості. Коли відбувається оновлення, весь віртуальний DOM оновлюється [20]». Це звучить неефективно, але віртуальний DOM набагато швидше, ніж реальний. Після завершення оновлення ReactJS порівнює оновлений віртуальний DOM і попередньо оновлений оригінальний DOM. Порівнюючи обидва DOM, ReactJS точно знає, де різниця. Тож він оновлює оригінальний DOM лише там, де була відбулася зміна.

Переваги React:

– легкість вивчення, завдяки простому дизайну, використання JSX (HTML-подібного синтаксису) для шаблонів і дуже докладної документації;

– розробники витрачають більше часу на написання сучасного JavaScript і менше звертають уваги на код, специфічному для фреймворка;

– дуже швидкий, завдяки реалізації React Virtual DOM і різним оптимізаціям рендеринга;

– відмінна підтримка рендеринга на стороні сервера, що робить його потужною платформою для контенто-орієнтованих додатків;

– першокласна підтримка Progressive Web App (PWA) завдяки генератору додатків `create-react-app`;

– прив'язка даних є односторонньою, що означає менше небажаних побічних ефектів. Redux, одна з передових платформ для управління станом додатків в React, її легко можна освоїти та використовувати;

– react втілює концепції функціонального програмування (FP), створюючи простий в тестуванні і багаторазово використовуваний код;

– програми можна створюватиза допомогою TypeScript або Facebook's Flow, які мають вбудовану підтримку JSX;

– перехід між версіями, як правило, не складний: Facebook надає «кодові модулі» для автоматизації більшої частини процесу;

– навички, отримані в React, можуть бути застосовані до розробки на React Native [14].

Недоліки React:

– react не однозначний і залишає розробникам можливість вибирати кращий спосіб розвитку;

– це може бути вирішено сильним лідерством проекту і хорошими процесами. Спільнота ділиться по способам написання CSS в React, які поділяються на традиційні таблиці стилів (CSS Modules) і CSS-in-JS (тобто Emotion і Styled Components) ;

– відхід React від компонентів, що базуються на класах, може стати перешкодою для розробників, яким зручніше об'єктно-орієнтоване програмування (ООП) ;

– змішування шаблонів з логікою (JSX) може заплутати деяких розробників, коли вони вперше стикаються з React.

Компанії, які використовують React: Facebook, Instagram, Whatsapp, New York Times, Netflix, Codecademy, Microsoft, Airbnb, Yahoo, Khan Academy, Asana, Atlassian, Intercom, Dropbox, Slack, Storybook і багато інших [14].

2.5. Опис структури програми та алгоритмів її функціонування

Коли розробник планує архітектуру свого майбутнього веб-додатку, корисно заздалегідь подумати про його розширюваність. Модульна архітектура програми може забезпечити хороший ступінь розширюваності, а перевага

використання модульної архітектури полягає в тому, що можна оновити (замінити) модуль без необхідності змінювати решту системи. Існує досить багато способів реалізації такої архітектури, але всі вони схожі за своїми фундаментальними принципами: поділ понять, самодостатність, взаємна сумісність усіх компонентів.

Модульне програмування - це метод побудови складних програм в ієрархічному порядку на основі нескладних програмних блоків або модулів. Формально програмний модуль - це скінчений набір операторів, що реалізує певний алгоритм. Структурно модуль - це окрема функціонально закінчена програмна одиниця, яка може використовуватися як самостійно так і бути частиною програмного комплексу.

Модульне програмування - це розробка та вдосконалення процедурного програмування та бібліотек спеціальних програм. Головною особливістю модульного програмування є стандартизація інтерфейсу між окремими програмними блоками - це окрема функціонально-повна програмна одиниця, структурована стандартним чином щодо компілятора та щодо поєднання його з іншими подібними блоками та завантаженнями. Як правило, кожен модуль має таблицю даних, яка містить усі основні її характеристики: мова програмування, область обсягу, вхідні та вихідні змінні, їх формат, обмеження на них, точки введення, параметри налаштування тощо. Обсяг модуля зазвичай не перевищує 1000 комп'ютерних команд або операторів мови програмування. В іншому випадку модуль стає громіздким і важким для сприйняття та використання [25].

Модульне програмування - це мистецтво розподілу завдання на безліч різних модулів, можливість широко використовувати стандартні модулі шляхом їх параметричного регулювання, автоматизація збірки готових модулів з бібліотек, банків модулів.

Основні поняття модульного програмування:

- кожен модуль реалізує одну незалежну функцію;
- кожен модуль має одну точку введення та виведення;

– розмір модуля слід якомога менше мінімізувати;

– кожен модуль може бути розроблений і закодований різними членами команди програмістів, а також може бути протестований окремо;

– вся система побудована з модулів;

– модуль не повинен викликати побічних ефектів;

– кожен модуль не залежить від того, які інші модулі реалізовані.

При цьому підході складна система ділиться на кілька частин, одночасно створених різними програмістами. Кожен модуль реалізує одну функцію. Розмір модуля невеликий, тому тестування можна здійснити, і його можна провести ретельно. Після того, як всі модулі були закодовані та протестовані, вони інтегруються та перевіряється вся система [26].

Процес проектування у випадку модульного програмування - це планування зверху вниз програмної системи. Фаза кодування та налагодження здійснюється знизу вгору: всі модулі, які були визначені під час планування, кодуються, автономно регулюються; вся система побудована та налаштована.

Під час технічного обслуговування перевіряється та регулюється лише той модуль, який працює неправильно. Очевидні переваги легше писати та тестувати програми, а витрати на обслуговування зменшуються..

Для забезпечення повноцінної роботи платформи для моніторингу динаміки цін та всіх необхідних функціональних можливостей користувача вказаних в пункті 2.1. Були розроблені декілька модулів та файлів, що реалізують базову логіку та представляють основний інтерфейс взаємодії користувача з платформою.

Варто розглянути основні файли застосунку:

– index.js. Сценарій, який завантажує додаток (App.js) в об'єктну модель документа;

– App.js. Файл в якому розміщенно всі компоненти додатку та реалізована їх маршрутизація;

– AppStyle.css. Файл містить налаштування таблиці стилів для всіх елементів програмного застосунку.

Для входу та регістрації користувача на платформу реалізовані наступні модулі:

– validationLoginForm.js. Використовується для валідації форми входу користувача;

– validationRegisterForm.js. Використовується для валідації форми регістрації користувача;

– LoginForm.js. Основний файл форми входу користувача з всіма елементами;

– RegistrationForm.js. Основний файл форми регістрації користувача з всіма елементами.

За представлення каталогу товарів платформи відповідають наступні файли:

– Market.js. Файл в якому розміщенно всі необхідні компоненти для роботи сторінки каталогу товарів;

– PaginationComponent.js.Модуль який відповідає за пагінацію (порядок нумерації сторінок) каталогу товарів;

– MainProduct.js. Модуль який відповідає за відображення каталогу товарів на сторінці;

– Item.js. Відображає сторінку товару на яку перейшов користувач.

Для відображення профілю користувача використані такі файли:

– CategorList.js. Файл який відповідає за відображення категорій ;

– CategorItem.js. Використовується для видалення непотрібних категорій;

– Profile.js. Містить компоненти необхідні для роботи сторінки профілю;

– Product.js. Модуль який відповідає за відображення вподобаних товарів користувача.

За представлення головної сторінки використані такі файли:

– Home.js. Містить компоненти необхідні для роботи головної сторінки;

– MainProduct.js. Модуль який відповідає за відображення каталогу товарів на сторінці;

– Sidebar.js. Використовується для фільтрації та сортування товарів за необхідними критеріями.

Для відображення Footer та Header використані такі файли:

– Header.js. Використовується для пошуку необхідного товару;

– Navbar.js. Модуль який відповідає за навігація по веб-сайту;

– Footer.js. Модуль який відповідає за відображення нижнього колонтитулу на сторінці.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними та вихідними є дані, що безпосередньо стосуються платформи моніторингу динаміки цін та інформації про ті товари, що на цій платформі знаходяться. Інформація про товар має такі загальні атрибути як код товару, назва, короткий опис, поточна вартість товару, дата початку встановлення поточної вартості, дата закінчення поточної вартості, файли, що можуть стосуватися товару (фотографії, текстові документи і т.д.).

Якщо роздивлятись платформу для моніторингу та динаміки цін як інформаційну систему в глобальному сенсі, то вхідні та вихідні дані передаються у форматі JSON, у випадку вхідних даних, що містить інформацію про товари, перелік категорій товарів, дані про поточного корисувача, якщо виконаний авторизований вхід.Вихідними даними буде інформація про створені профілі користувачів (id користувача, електрона адреса та пароль), інформація про них, а також перелік вподобаних товарів та категорій користувачів у форматі JSON.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для користувача є важливим мати систему, спроможну запускати та працювати з сучасними браузерами, тож необхідно мати такі мінімальні параметри ЕОМ :

– Операційна система [OS]: Windows XP з пакетом оновлень 2 + Windows Vista / Windows 7/ Windows 8 / Windows 10;

– ЦП [CPU]: Intel Pentium 4 / Athlon 64 або більш пізньої версії з підтримкою SSE2;

– відеоадаптер [GPU]: 3D адаптер nVidia, Intel, AMD/ATI;

– звукова карта DirectX Compatible;

– відеопам’ять [VRAM]: 64 МБ;

– накопичувач [HDD]: 350 МБ;

– оперативна пам’ять [RAM]: 512 МБ.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

– Visual Studio Code;

– браузери (Google Chrome, Opera);

– Git, GitHub.

Також нам потрібно використовувати менеджер пакетів npm для швидкого і легкого завантажування та встановлювати пакетів компонентів для розробки, використовуючи командний рядок. Це можна зробити написавши в терміналі команду "npm install", також вона встановлює залежності між компонентами в додатках. На початку створення додатку вам потрібно написати команду "npm

init", щоб створити package.json, в якому буде зберігатися інформація про завантажені та встановлені програмні пакети, тобто їх версії та залежності.

Завантажити та встановити необхідний фреймворк, можна шляхом введення необхідної команди «npm install назва фреймворку --save ». Команда «--save» додасть фреймворк в список залежностей додатку і його можна використовувати з усіма встановленими компонентами.

Менеджером пакетів у Node.js є npm (скорочення від менеджера пакетів вузлів), npm - це стандартний менеджер пакетів, який автоматично встановлюється разом з Node.js. Він використовується для завантаження пакетів із хмарного сервера npm або для завантаження пакетів на ці сервери [18].

Завантаження та встановлення пакетів у цьому випадку здійснюється в терміналі середовища розробки Visual Studio Code.

Visual Studio Code - це сервіс, який позиціонується як "полегшений" редактор коду для розробки крос-платформних веб- та хмарних додатків. Це не повноцінний IDE, а зручний, простий і доступний редактор коду з розширеними функціями. У той же час його досить просто освоїти, має зручний інтерфейс та всі необхідні функції для створення додатків. Якщо чогось не вистачає, ви завжди можете це компенсувати, встановивши додаткові розширення.

Особливості Visual Studio Code:

– VS Code дозволяє розробляти консольні та графічні програми для користувальницького інтерфейсу, включаючи програми, що підтримують технологію Windows Forms, а також веб-сайти, веб-програми та веб-служби як у коді, так і в керованому коді для всіх платформ;

– редактор має вбудований налагоджувач, інструменти для роботи з Git та інструменти для рефакторингу, навігації кодом, автозавершення стандартних конструкцій та контекстну допомогу;

– продукт підтримує розробку платформ ASP.NET та Node.js і вважається легким рішенням, яке дозволяє обійтися без повноцінного інтегрованого середовища розробки;

– великим плюсом редактора є підтримка великої кількості мов, таких як C ++, C #, Python, PHP, JavaScript та інших [23].

Переваги Visual Studio Code:

– вбудовані засоби інтеграції з GitHub, GIT та Visual Studio Team Services для швидкого тестування, побудови, упаковки та розгортання різних типів програм;

– простота роботи з проектами Unity;

– робота з Mono та Node.js із вбудованим налагоджувачем;

– підтримка TypeScript та JavaScript;

– публікація створених програм у Microsoft Azure через Visual Studio Team Services;

– підтримка майже всіх мов програмування;

– написання коду для конкретного завдання з подальшою інтеграцією в проект (з надбудовою або безпосередньо);

– велика бібліотека шаблонів, готових фрагментів коду та фрагментів з можливістю додавання власних елементів;

– робота з декількома проектами одночасно (в декількох вікнах);

– інтерфейс можна розділити на дві панелі для порівняння коду;

– функція налагодження.

Git – це приклад розподіленої системи контролю версій (DVCS), яка здебільшого необхідна для розробки публічного комерційного та некомерційного програмного забезпечення. Також він дозволяє отримати повний доступ до кожного необхідного файлу, гілки а також ітерації проекту та дозволяє будь-якому користувачеві отримати доступ до повної та самодостатньої історії всіх внесених змін. На відміну від популярних колись централізованих систем контролю версій, «DVCS на зразок Git не вимагає постійного з'єднання з центральним сховищем. Розробники можуть працювати з будь-якого місця і співпрацювати асинхронно з будь-якого часового поясу [15]».

GitHub – це платформа по розміщенню коду для контроля версій та співпраці. Він дозволяє спільно працювати над проектами з будь-якого місця та зберігати версії проекту на віддаленому сервері.

2.7.3. Виклик та завантаження програми

Розроблений браузерний веб-сайт завантажується шляхом переходу по URL, скомпонованого із зазначених хосту та порту в налаштуваннях використовуваного серверу, за допомогою одного з сучасних браузерів. Прикладом на основних десктопних ОС, таких як Windows 10, Linux, Mac OS, є такі браузери, як Google Chrome, Firefox, Opera. Для мобільних платформ, а саме IOS, Android встановлюваний мобільний браузер.

2.7.4. Опис інтерфейсу користувача

За замовчуванням на початку роботи із застосунком користувачу відображається головна сторінка (рис. 2.2 і 2.3). Вона і буде основною, на її базі будуть з’являтися інші компоненти залежно від тієї чи іншої взаємодії користувача з інтерфейсом додатку (рис. 2.4 – 2.19). Такими компонентами взаємодії в більшості своїй є кнопки. Інтерфейс розроблявся з властивістю адаптуватися до екранів як звичайних моніторів так і мобільних пристроїв.

Після натиснення кнопки Sidebar в лівому куті екрану з'явиться список категорій товарів (рис. 2.5) . Щоб виконати фільтрацію товарів за необхідними кретеріями необхідно натиснути кнопку Фільтрація в правому куті екрану після цього з'явиться необхідне вікно (рис. 2.6). Якщо користувач хоче здійснити сортування товарів для цього йому необхідно знайти кнопку Сортування та після її натисненення обрати з випадаючого списку (рис. 2.7) необхідний параметер. При необхідності користувач може здійснити пошук потрібного товару за

назвою після заповнення пошукого рядку та написнення кнопки Знайти (рис. 2.4).

Для переходу на сторінку опису товару та перегляду графіку середньої вартості товару за певний період часу (рис. 2.12). Користувачеві потрібно на головній сторінці додатку обрати необхійдний товар та натиснути на нього. Якщо користувач зареєстрований в системі то на цій сторінці в нього буде можливість додати цей товар в список обраних (рис. 2.11).

Щоб користувач мав змогу отримати всі функціональні можливості додатку необхідно виконати регістрацію та вхід в систему. Для регістрації користувача необхідно заповнити всі поля (рис. 2.13) та натиснути кнопку регістрації. Після виконання успішної регістрації з'явиться відповідне вікно (рис. 2.14).Тепер користувачу доступний вхід в систему через створений обліковий запис (рис. 2.15). Після цього в додатку з'явиться новий розділ меню Профіль користувача (рис. 2.17) та ряд функцій доступних зареєстрованим користувачам.

Рис. 2.2. Головна сторінка платформи (500x900)

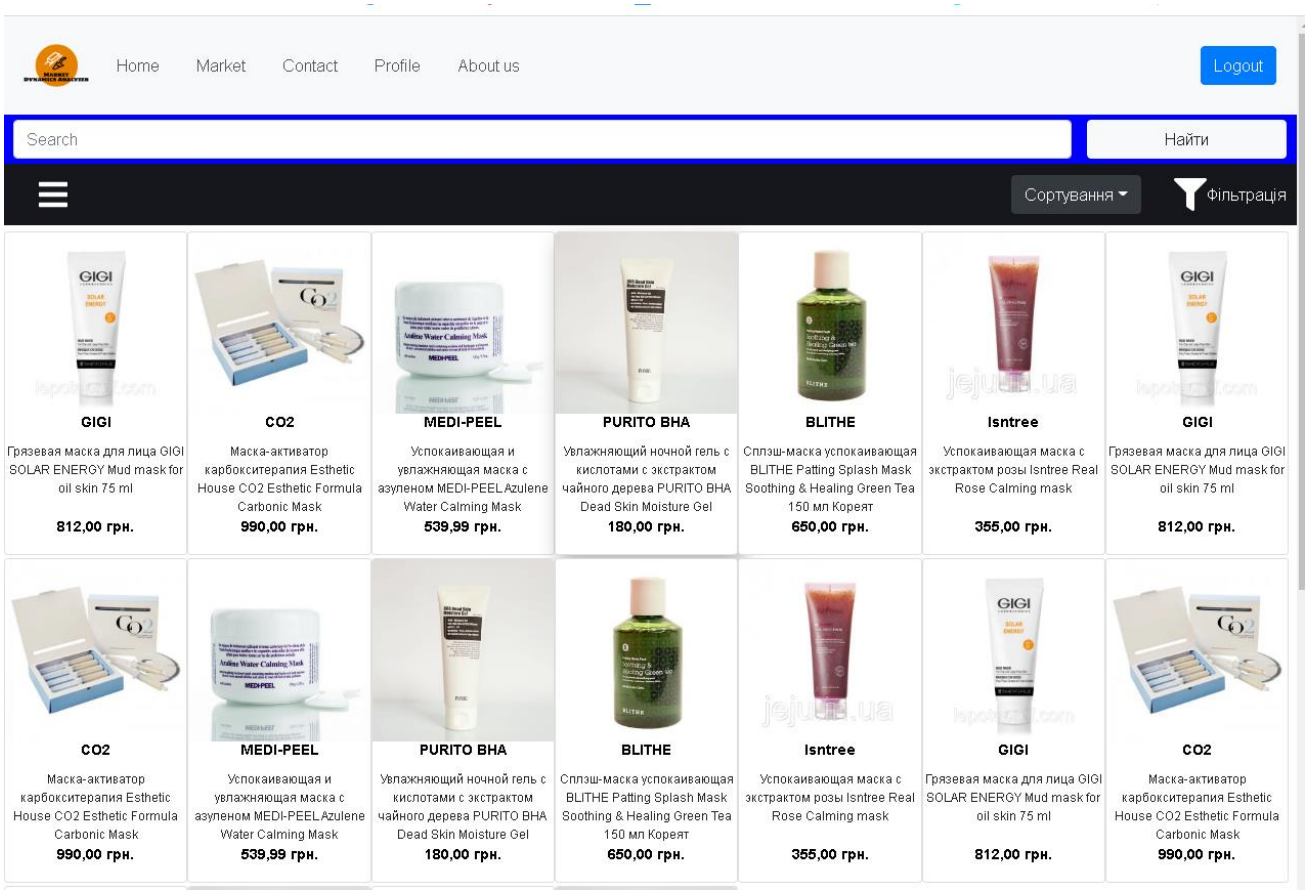

Рис. 2.3. Головна сторінка платформи (1280x1024)

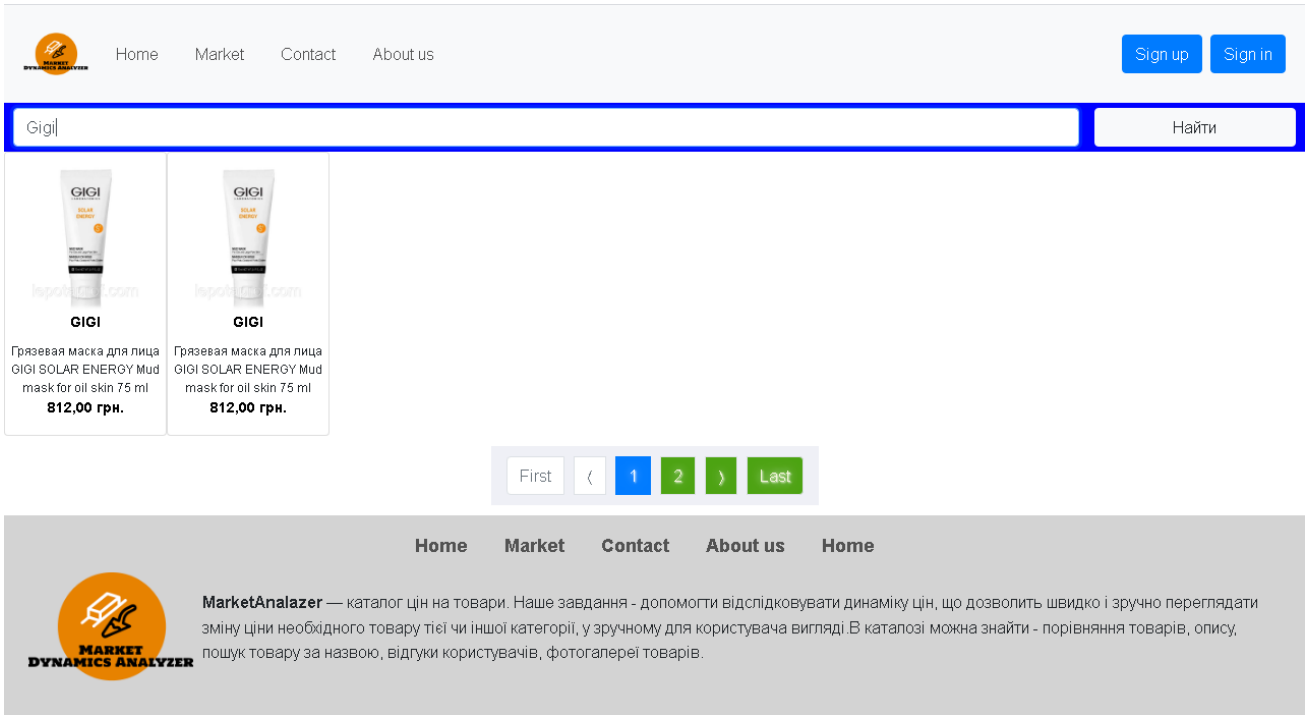

Рис. 2.4. Пошук товару (1280x1024)

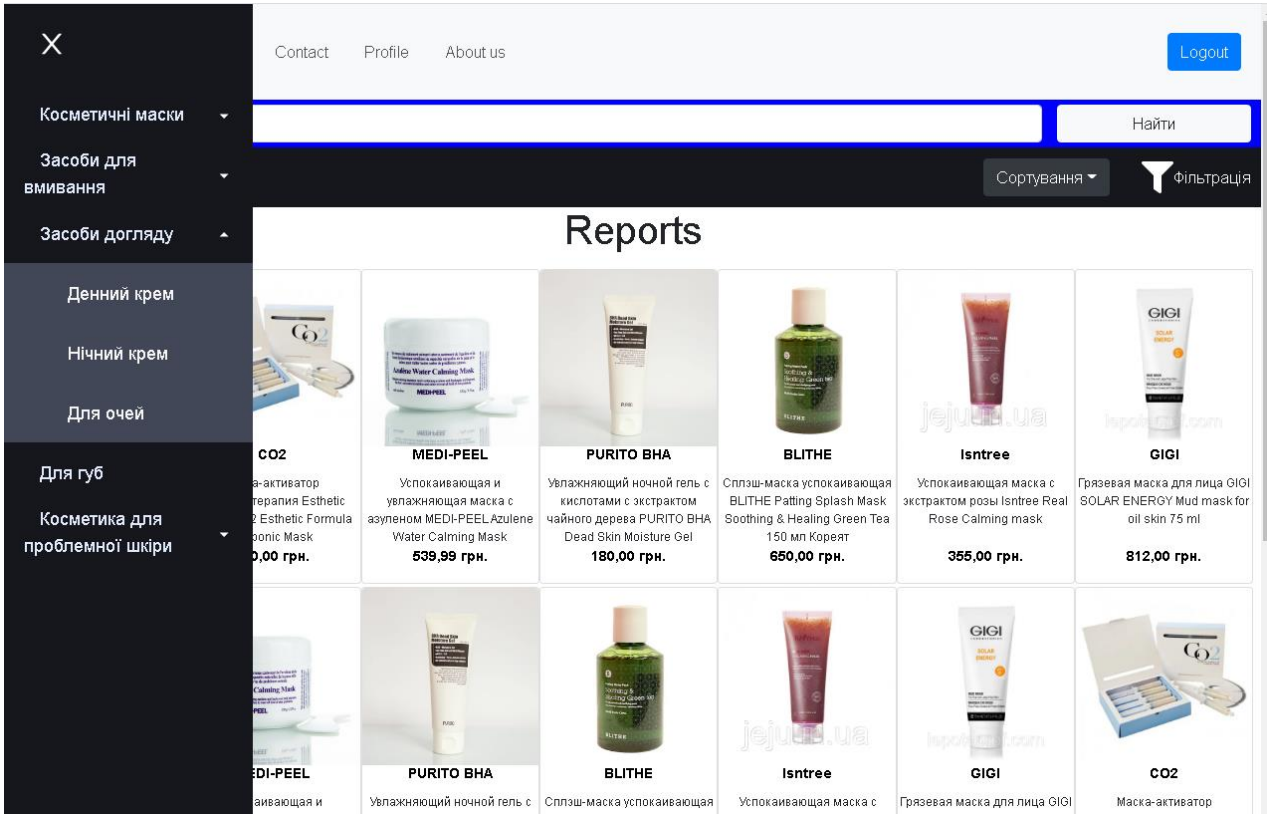

Рис. 2.5. Список категорій (1280x1024)

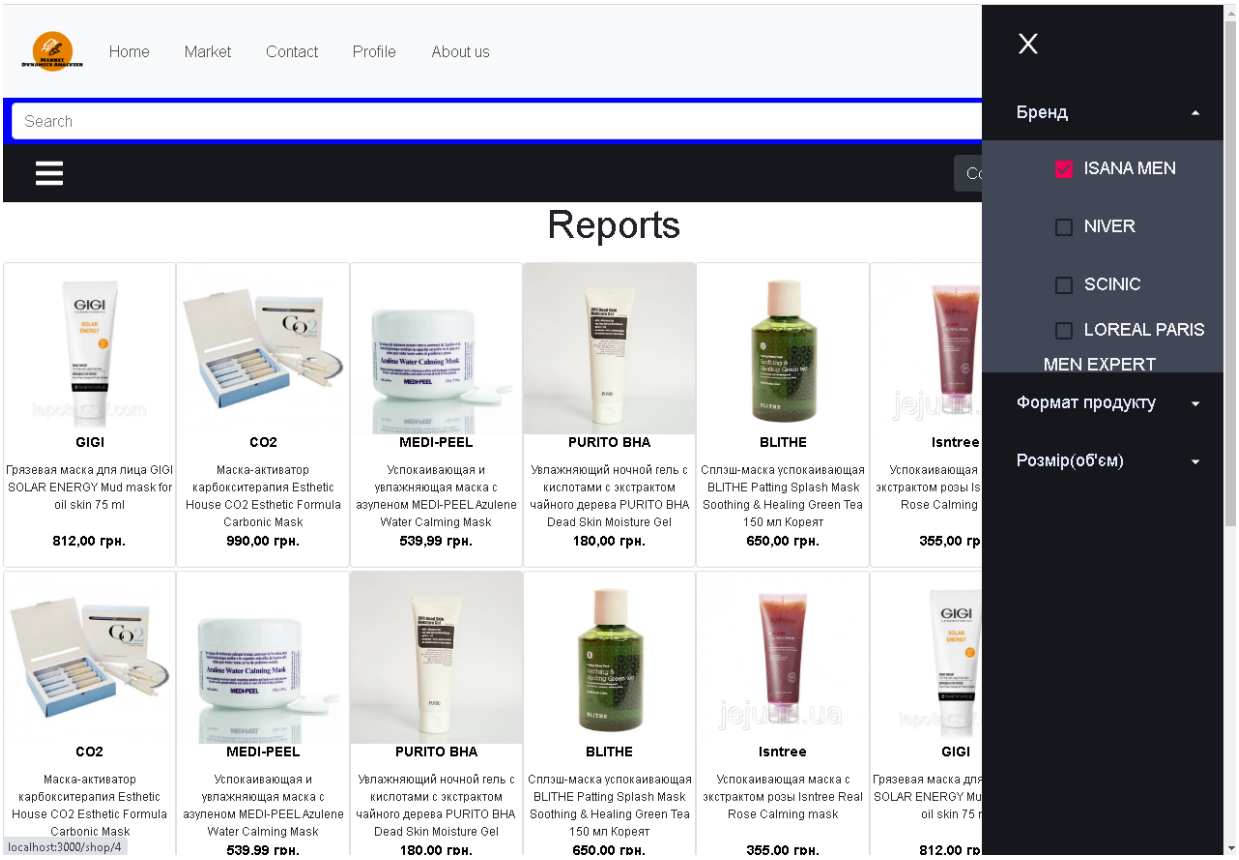

Рис. 2.6. Фільтрація товарів на готовній сторінці (1280x1024)

Рис. 2.7. Сортування товарів на готовній сторінці (500x900)

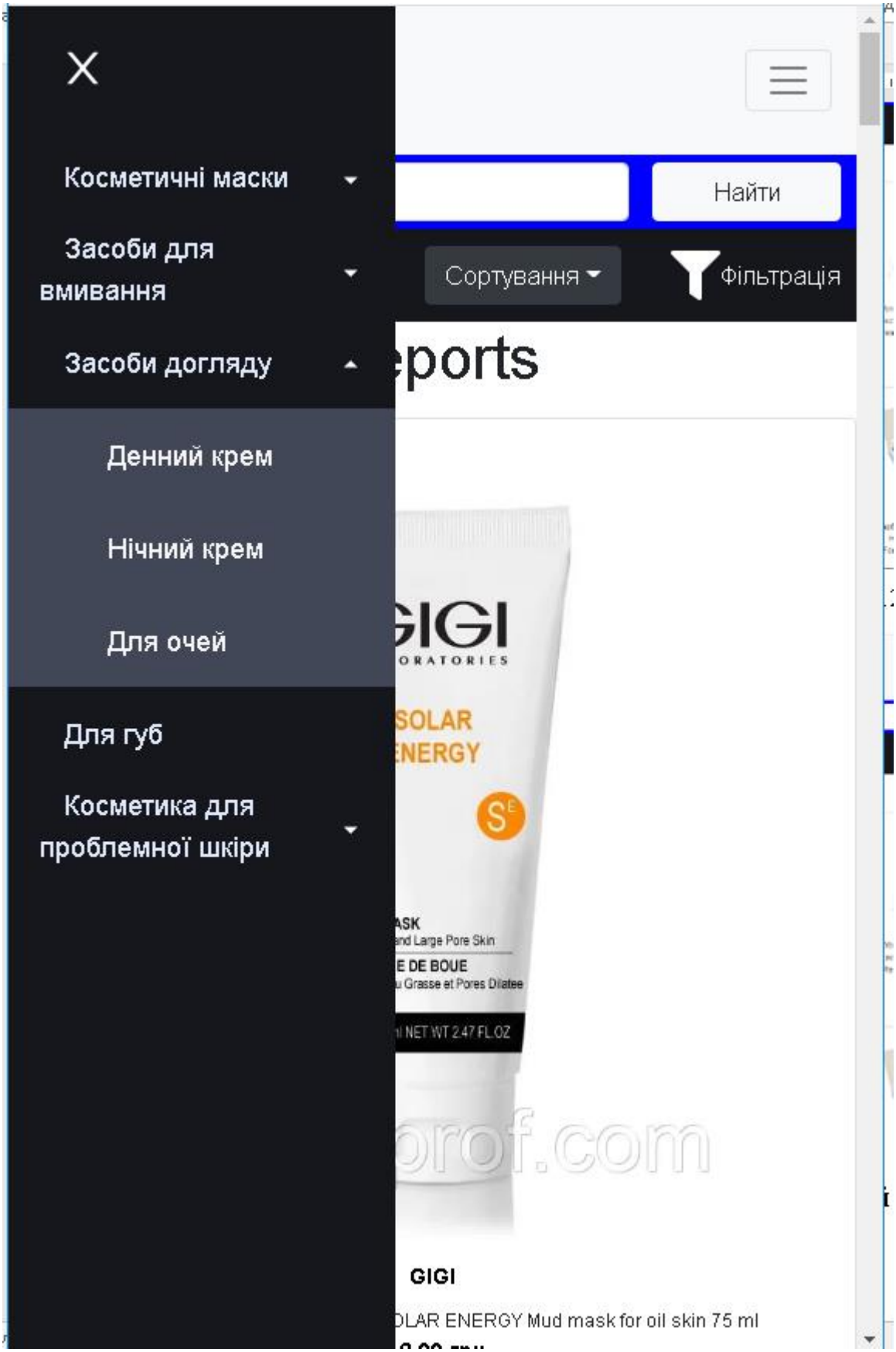

Рис. 2.8 Список категорій (500x900)

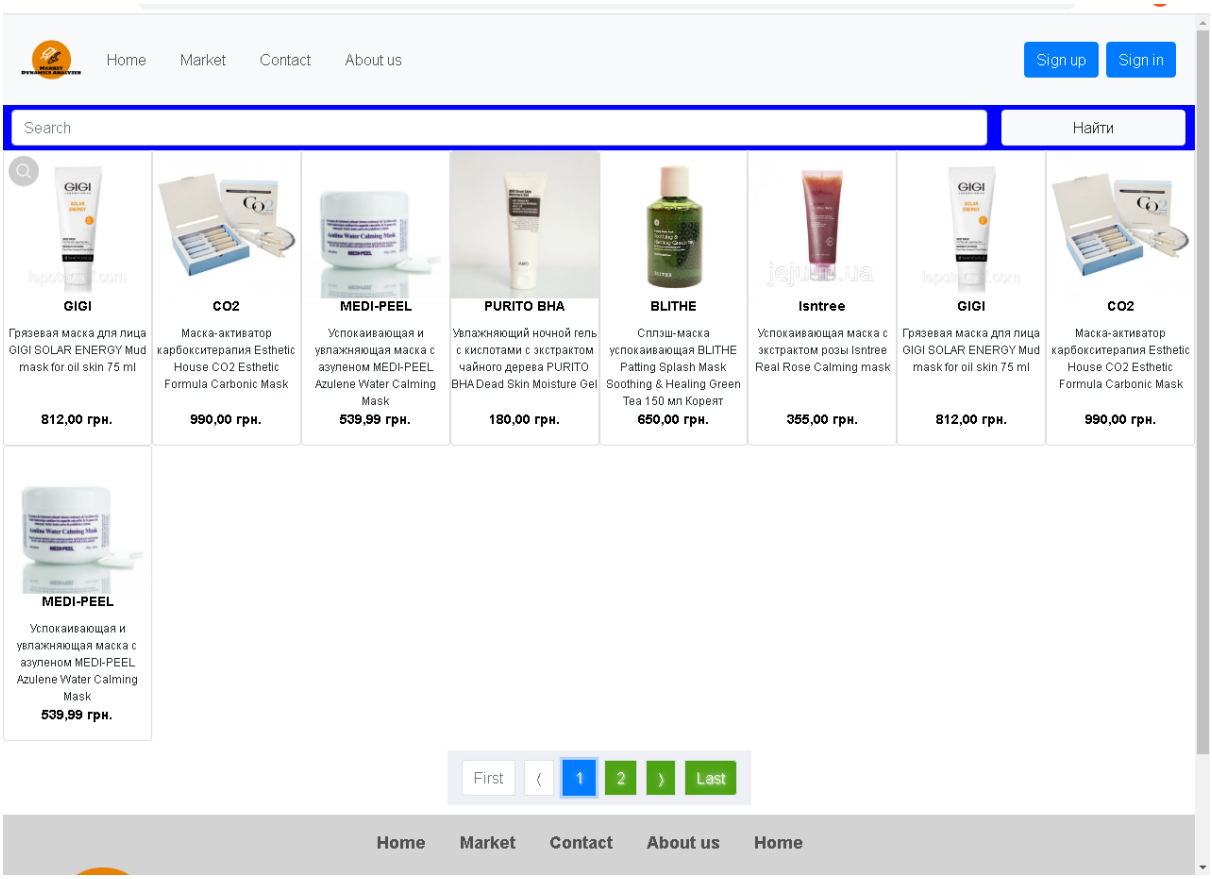

Рис. 2.9. Паджинація сторінки (1280x1024)

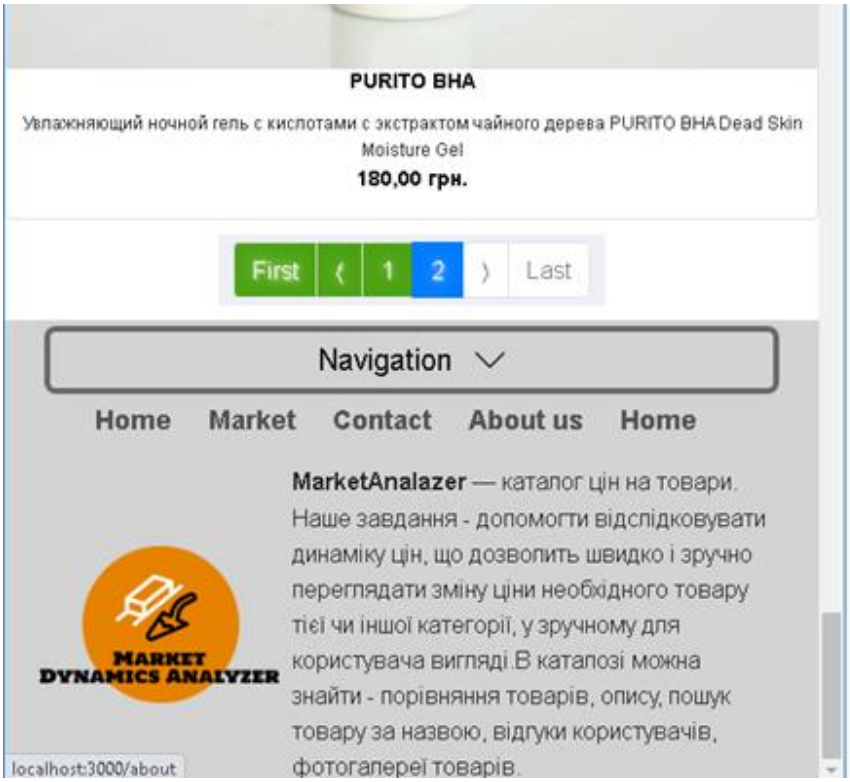

Рис. 2.10. Паджинація та футер (500x900)

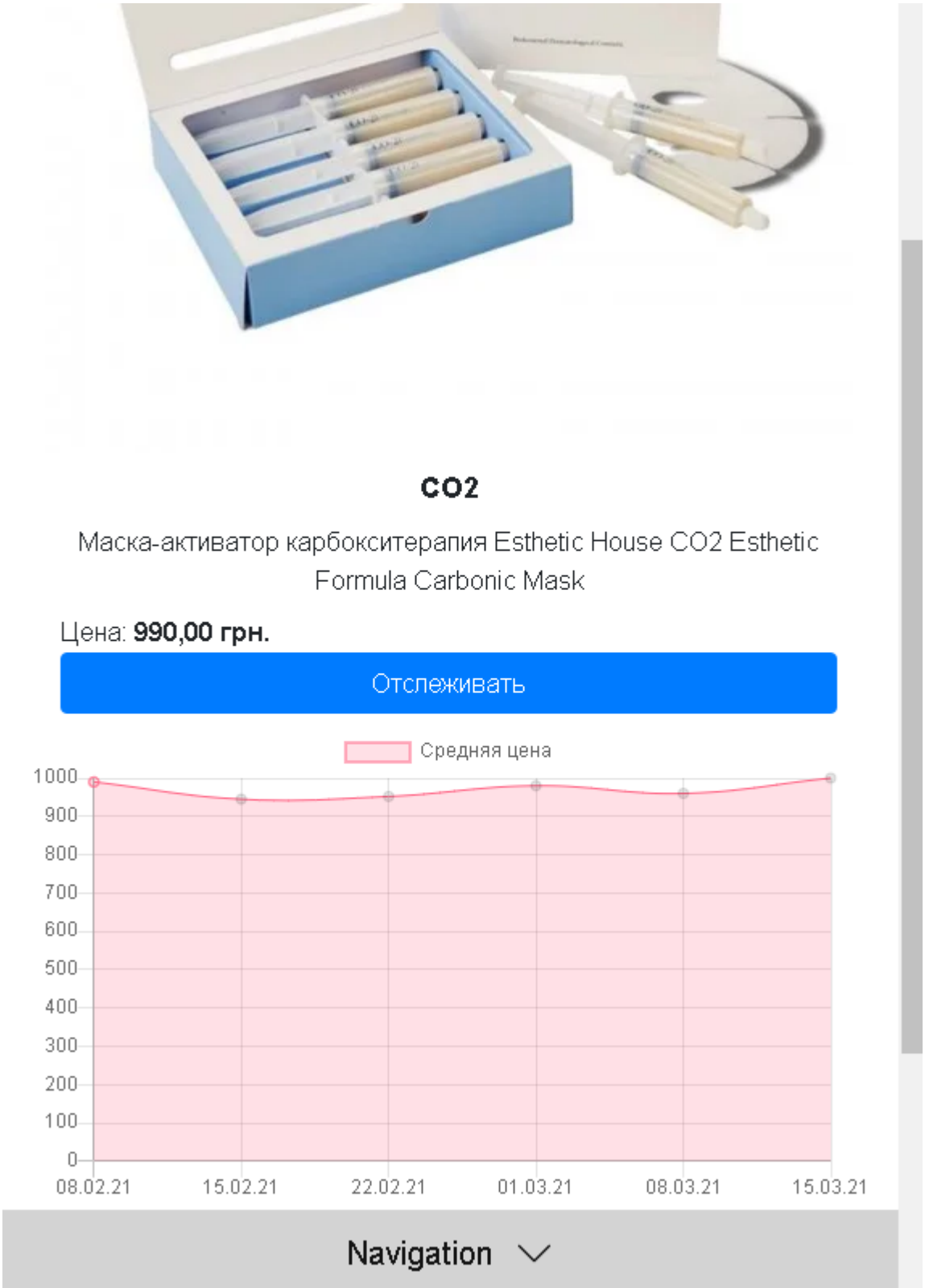

Рис. 2.11. Перегляд товару та графік середньої ціни (500x900)

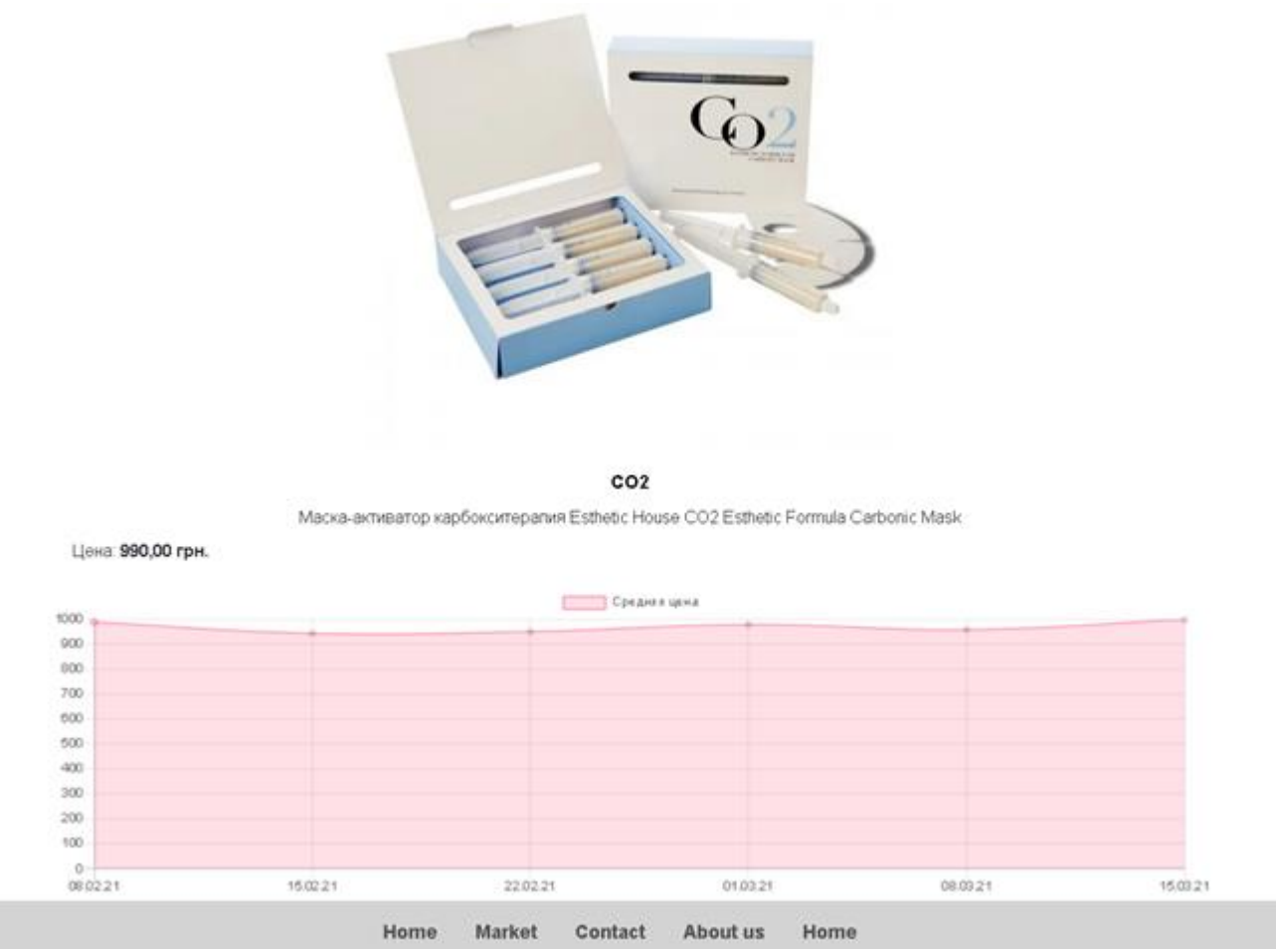

Рис. 2.12. Перегляд товару та графік середньої ціни (1280x1024)

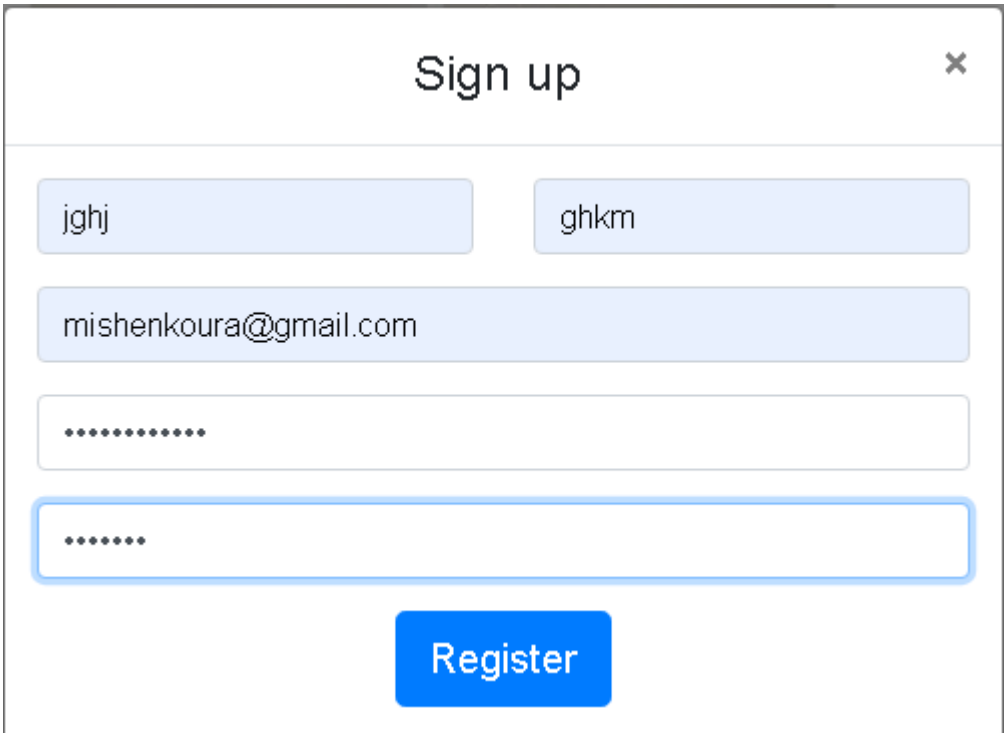

Рис. 2.13. Форма регістрації користувача

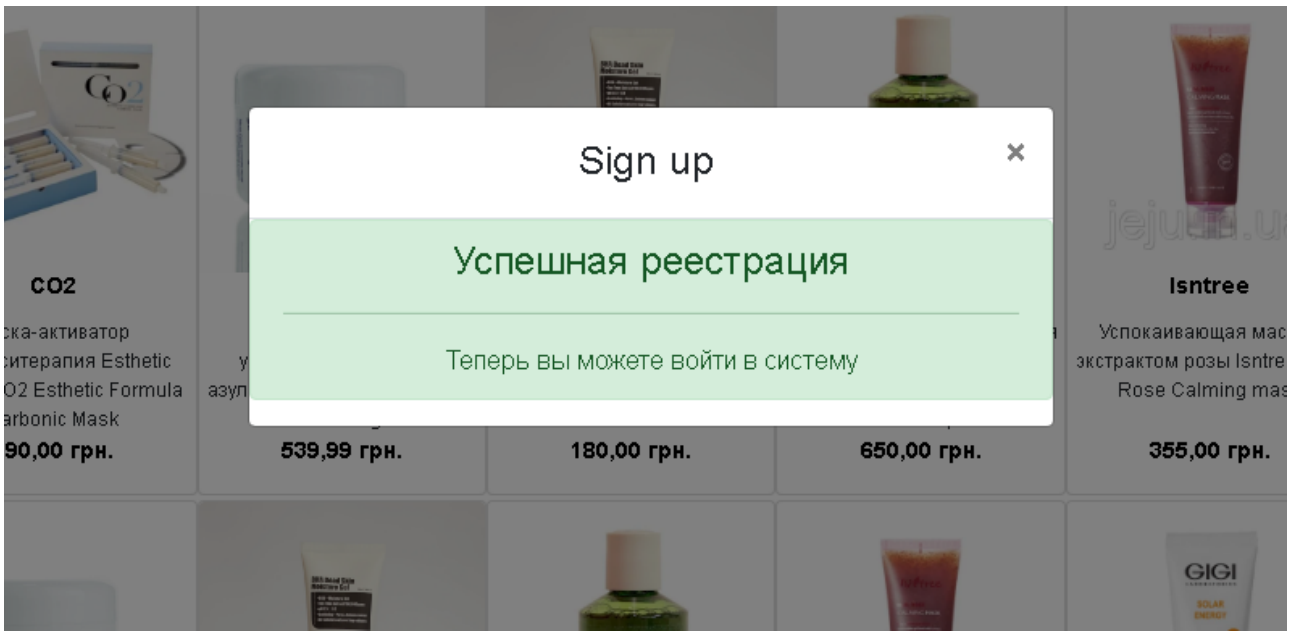

Рис. 2.14. Вікно успішної регістрації

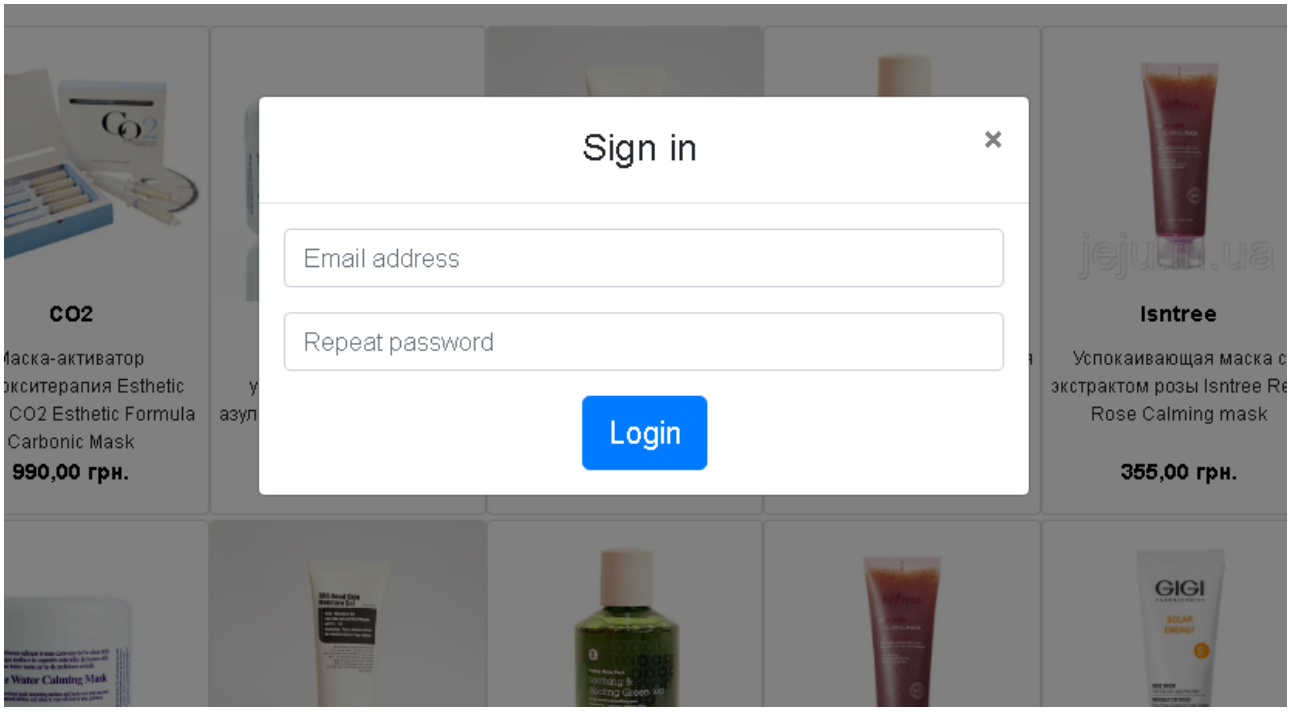

Рис. 2.15. Вікно входу користувача на платформу

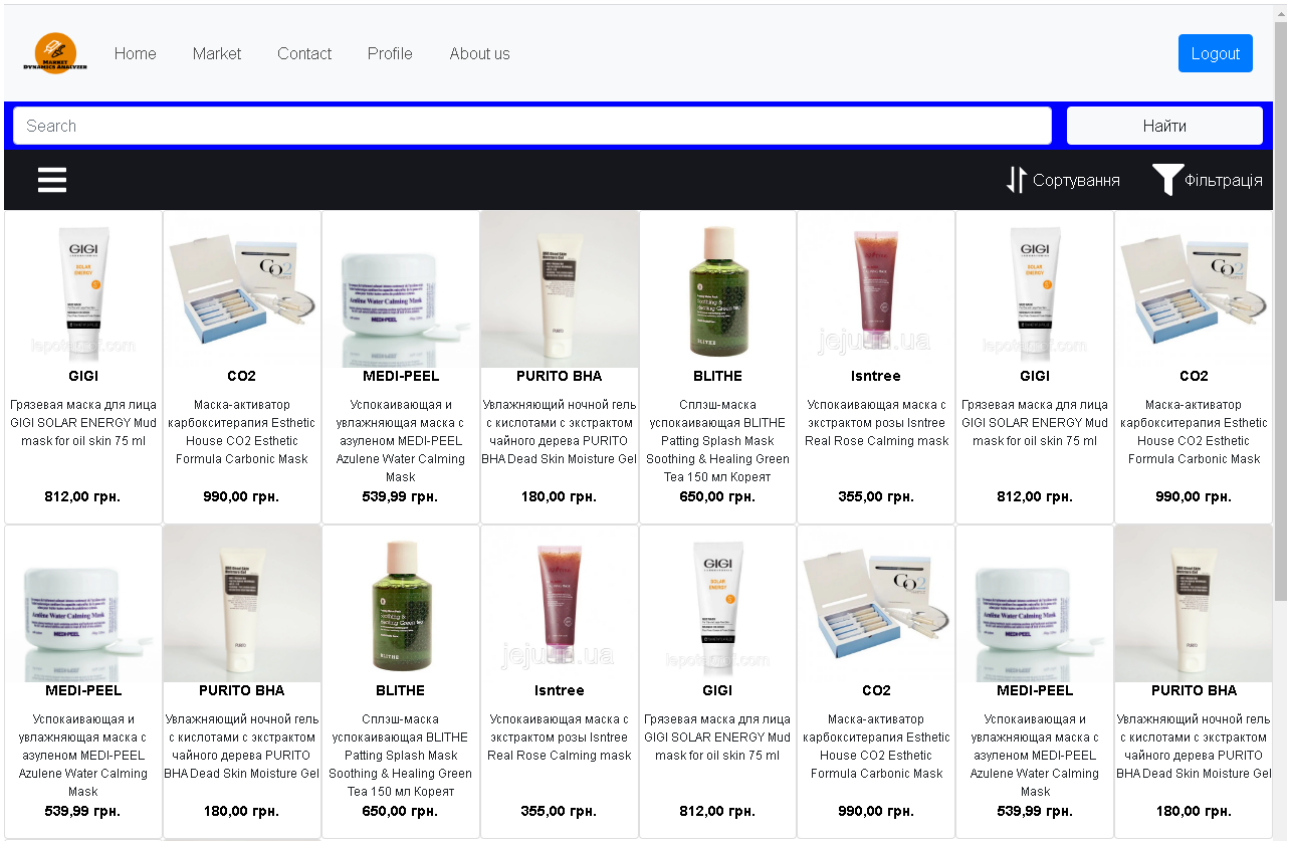

Рис. 2.16. Головна сторінка платформи після входу (1280x1024)

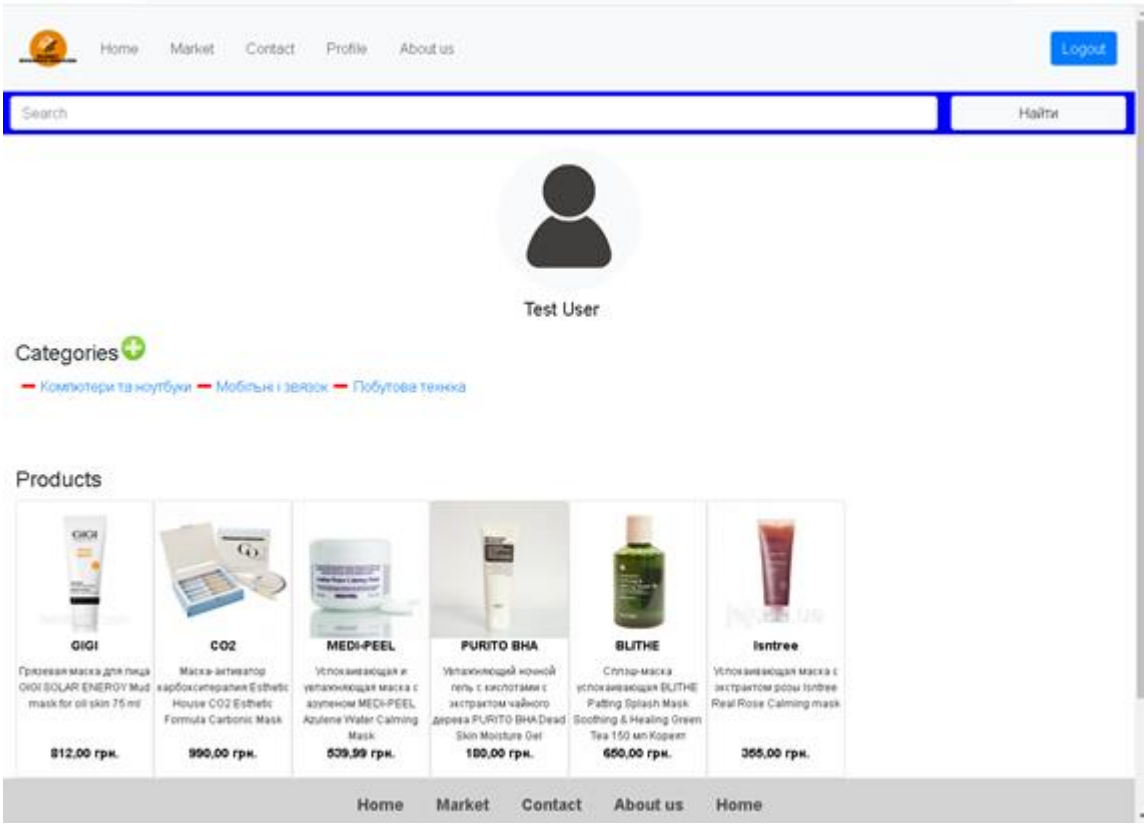

Рис. 2.17. Сторінка профілю користувача (1280x1024)

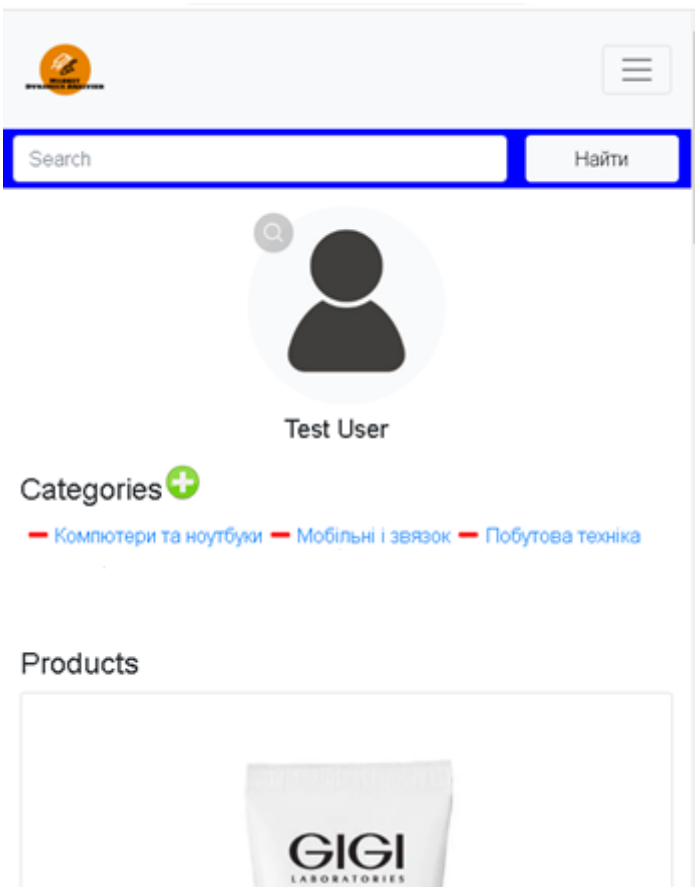

Рис. 2.18. Сторінка профілю користувача (500x900)

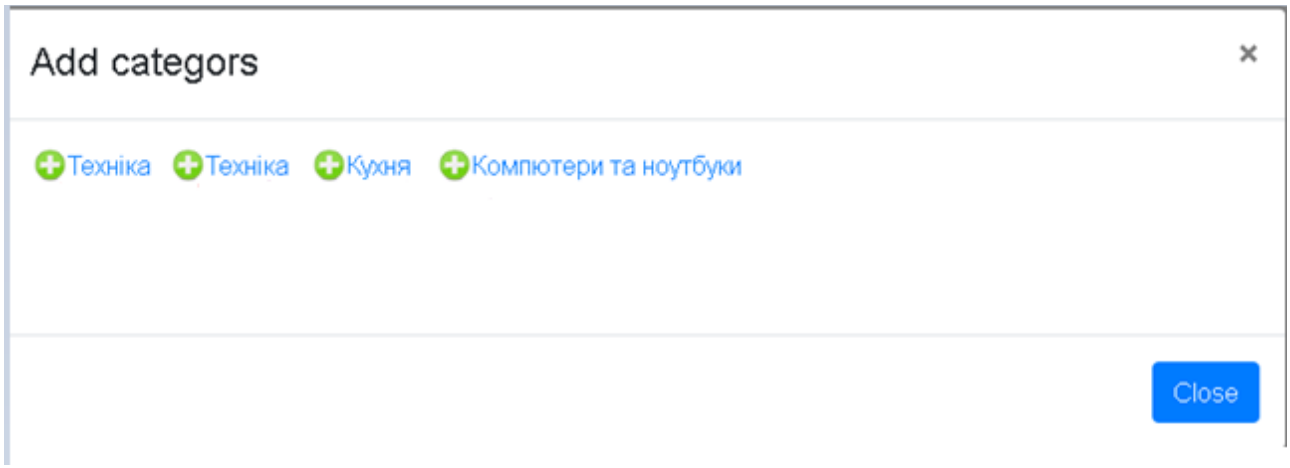

Рис. 2.19. Форма додання категорій користувача

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1235;
2. коефіцієнт складності програми – 1,4;
3. коефіцієнт корекції програми в ході її розробки – 0,06;
4. годинна заробітна плата програміста – 75 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності –1,1;
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:

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

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):

$$Q = q \cdot C \cdot (1 + p), \text{ де} \qquad (3.2)$$

q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1235 \cdot 1{,}4 \cdot (1 + 0{,}06) = 1832{,}74;$$

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

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \text{ людино-годин,} \qquad (3.3)$$

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

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

$$t_u = \frac{1832{,}74 \cdot 1{,}2}{85 \cdot 1{,}1} = 23{,}52, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

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

$$t_a = \frac{1832,74}{20 \cdot 1,1} = 83,97, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

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

$$t_n = \frac{1832,74}{25 \cdot 1,1} = 66,65, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

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

$$t_{отл} = \frac{1832,74}{5 \cdot 1,1} = 333,23, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

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

$$t_{отл}^{к} = 1,2 \cdot 333,23 = 399,88, \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial p} + t_{\partial o}; \quad (3.8)$$

де $t_{\partial p}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{\partial \text{р}} = \frac{Q}{(15...20)\cdot K}; \quad (3.9)$$

$$t_{\partial \text{р}} = \frac{1832,74}{20 \cdot 1,1} = 83,31, \text{ людино-годин.}$$

$t_{\partial o}$– трудомісткість редагування, печатки й оформлення документації

$$t_{\partial o} = 0,75 \cdot t_{\partial \text{р}}; \quad (3.10)$$

$$t_{\partial o} = 0,75 \cdot 83,31 = 62,48, \text{ людино-годин.}$$

$$t_{\partial} = 83,31 + 62,48 = 145,79, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$\text{t} = 50 + 23,52 + 83,97 + 66,65 + 333,23 + 145,79 = 703,16, \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 703,16 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

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

$$Жс_{пм} = ,_{,n} + ,_{Ме\,\mp}\text{, грн,} \qquad (3.11)$$

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

$$З_{ЗП} = t \cdot С_{ПР}\text{, грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

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

$$З_{ЗП} = 703{,}16 \cdot 75 = 52737\text{, грн.}$$

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

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

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

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

$$З_{МВ} = 333{,}24 \cdot 15 = 4998{,}6\text{ грн.}$$

$$К_{ПО} = 52737 + 4998{,}6 = 57735{,}6\text{ грн.}$$

Очікуваний період створення ПЗ:

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

де B_k- число виконавців;

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

$$T = \frac{703{,}16}{1 \cdot 176} = 4\text{міс.}$$

Висновки. На розробку даного програмного забезпечення піде 703,16 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 4 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 57735,6 грн.

ВИСНОВКИ

В даній кваліфікаційній роботі була розроблена платформа Market Dynamics Analyzer для моніторингу динаміки цін необхідних товарів. Веб-застосунок має бути, сумісний з десктопними та мобільними пристроями, який буде давати доступ до платформи, адаптованої для користувачів, бажаючих отримувати інформація про минулі та поточні ціна на необхідний користувачеві товар.

Призначення розробки кваліфікаційної роботи є реалізація програмного додатку Market Dynamics Analyzer для моніторингу динаміки цін в Інтернеті. Дане програмне забезпечення дасть змогу користувачам, переглядати динаміку змін цін необхідного товару у зручному форматі. Для звичайних користувачів це може стати корисним тим, що допоможе знайти найбільш вигіднішу пропозиція серед цікавивших його товарів. А для деяких інтернет-магазинів може стати в нагоді для отримання необхідної інформації яка допоможе в прийняті правильного рішень щодо формування власної цінової політики.

Під час виконання даного кваліфікаційної роботи були виконані наступні задачі:

- проаналізовано предметну область задачі, що вирішується;
- проведено порівняння з можливостей існуючих подібних систем;
- вжиті заходи щодо забезпечення інформаційної безпеки додатку;
- обрано оптимальну структуру і параметри додатку;
- написано програмний код веб-застосунку;
- розроблено рекомендації щодо використання застосунку.

Програма реалізована на таких мовах програмування як JavaScript, HTML та CSS з використанням ReactJS бібліотеки Javascript.

Також у кваліфікаційній роботі було визначено трудомісткість розробленого програмного продукту (703,16 люд-год), проведений підрахунок вартості роботи по створенню програми (57735,6 грн) та розраховано час на його створення (4 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sidorchuk, R.; Skorobogatykh, I.; Kiselev, V.; Yemets, V.; Valiullina, R. Price Monitoring of Socially Significant Groceries: A Results Review of Two Research Waves. Mediterr. J. Soc. Sci. 2015.242 с.

2. Економічний аналіз. Частина 1 / ред. Войтоловський Н. В., Калініна А. П., Мазурова І. І. - М: Юрайт. 2019. 292 с.Keith J. Grant. CSS in Depth. Manning, 2018. 472 с.

3. Макшанов А.В. Технології інтелектуального аналізу даних. - М: Лань. 2019. 212 с.

4. Миркин Б. Г. Введення в аналіз даних. — М.: Юрайт. 2020. 175 с.

5. Офіційний опис можливостей uXprice [Електронний ресурс]. Режим доступу: https://uxprice.com/ (переглянуто 17.05.21).

6. Robin Wieruch. The Road to Learn React: Your Journey to Master Plain Yet Pragmatic React. Js. CreateSpace Independent Publishing Platform, 2018. 208 с.

7. Назаров С. В., Гудино Л. П., Кириченко А. А. Операційні системи. Практикум для бакалаврів; КноРус - Москва, 2012. - 376 с.

8. Дейтел Х. М., Дейтел П. Дж., Чофнес Д. Р. Операційні системи. Частина 1. Основи і принципи; Біном-Пресс - Москва, 2011. - 448 с.

9. Robin Wieruch. The Road to Learn React: Your Journey to Master Plain Yet Pragmatic React. Js. CreateSpace Independent Publishing Platform, 2018. 208 с.

10. Паттерни проектування - Ерік Фрімен Елізабет Фрімен Кетті Сьєрра Берт Бейтс, 2016.- 656 с.

11. RSDN [Електронний ресурс]: Model-View- Controller. - Режим доступу: http://rsdn.org/article/pattems/generic-mvc.xml (переглянуто 19.05.21).

12. Bootstrap в прикладах. / Пер. з англ. Рагімов Р. Н. / Наук. ред. Кисельов А. Н. - М: ДМК Пресс, 2017. - 314 с.

13. Marijn Haverbeke. Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming. O’Reilly Media, 2019. 474 с.

14. Robin Wieruch. The Road to Learn React: Your Journey to Master Plain Yet Pragmatic React. Js. CreateSpace Independent Publishing Platform, 2018. 208 с.

15. Brent Laster. Professional Git. 1st Edition. Wrox, 2016. 433 с.

16. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 « Інженерія програмного забезпечення » галузі знань 12 Інформаційні технології/, Л.М. Коротенко, О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

17. Методичні рекомендації щодо написання, оформлення та представлення учнівських науково-дослідницьких робіт учнів – членів Малої академії наук України / Г.Г. Півняк, Л.М. Коротенко, І.М. Удовик, Є.М. Головня – Д.: ДВНЗ «Національний гірничий університет», 2017. – 24 с

18. Хабрахабр [Електронний ресурс]: npm для простих смертних. - Режим доступу: https://habrahabr.ru/post/243335/ (переглянуто 18.05.21).

19. Сучасний підручник Javascript [Електронний ресурс]: Введення в JavaScript - Режим доступу: https://leam.javascript.ru/intro (переглянуто 18.05.21).

20. Siteacademy [Электронный ресурс]: React JS для начинающих. - Режим доступа: http://siteacademy.ru/jquery/reactjs- overview (переглянуто 19.05.21).

21. Eric Elliott Programming JavaScript ApplicationsRobust Web Architecture With Node, HTML5, and Modern JS Libraries. / Publisher: O'Reilly Media, February 2013. – 300с.

22. Девід Фленаган. JavaScript. Детальне керівництво, 2018.1080 с.

23. Хейлсберг А., Торгерсен М., Вілтамут С., Голд П. Мова програмування для Microsoft .NET. Класика Computers Science,. - СПб: Пітер, 2012. - 784 с.

24. Jeremy McPeak, Paul Wilton. Beginning JavaScript, 2015.- 768 с.

25. Романов, Є.Л. Сі / Сі ++. Від дилетанта до професіонал / Модульне програмування, 2014. - 600 с.

26. Спрол, Антон.Думай як програміст: креативний підхід до створення коду. С ++ версія / Антон Спрол. - Москва: Ексмо, 2018. - 272 с.

ДОДАТОК А

КОД ПРОГРАМИ

App.js

```
import { React, useState, useEffect } from "react";
import Header from "./components/Header/Header.js";
import Market from "./components/page/Market/Market.js";
import Footer from "./components/Footer/Footer.js";
import About from "./components/page/About.js";
import Profile from "./components/page/Profile/Profile.js";
import Contact from "./components/page/Contact";
import Home from "./components/page/Home";
import Item from "./components/page/Item";
import userData from "./components/db/profile.json";
import "./AppStyle.css";
import "./index.css";
import "bootstrap/dist/css/bootstrap.min.css";

import {
  BrowserRouter as Router,
  Switch,
  Route,
  Redirect,
} from "react-router-dom";

export default function App() {
  const [searchItem, setSearchItem] = useState("");
  const [token, setToken] = useState("");

  useEffect(() => {
    if (localStorage.getItem("token")) {
      setToken(localStorage.getItem("token"));
    } else {
      setToken("");
    }
  }, []);

  useEffect(() => {
    setSearchItem(searchItem);
    return () => {
      setSearchItem("");
    };
  }, [searchItem]);

  return (
    <Router>
      <>
        <Header
          setSearchItem={setSearchItem}
          searchItem={searchItem}
          token={token}
          setToken={setToken}
```

```
      />
      <Switch>
        <Route
          path="/"
          exact
          component={() => (
            <Home searchItem={searchItem} setSearchItem={setSearchItem} />
          )}
        />
        <Route
          path="/market"
          exact
          component={() => (
            <Market searchItem={searchItem} setSearchItem={setSearchItem} />
          )}
        />
        <Route path="/shop/:id" component={Item} />
        <Route
          path="/profile"
          component={() => (
            <Profile
              searchItem={searchItem}
              setSearchItem={setSearchItem}
              data={userData}
            />
          )}
        >
          {!localStorage.getItem("token") ? <Redirect to="/" /> : ""}
        </Route>
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
      </Switch>
      <Footer />
    </>
  </Router>
 );
}
```

AppStyle.css

```
@import url("https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;1,100;1,300;
1,400;1,500;1,700&display=swap");

body {
   margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;

}

button.d-sm-none.navbar-toggler.collapsed img {
  transform: rotate(-90deg);
  transition: 0.5s;
}
```

```
button.d-sm-none.navbar-toggler img {
  transform: rotate(90deg);
  transition: 0.5s;
}

.dropdown-basic-button {
  background-color: rgb(7, 7, 7);
}
.page-link {
  text-shadow: 1px 1px 2px white, 0 1px 2em white, 0 0 0.5em white;
  color: white;
  background-color: #4ca312;
  color: beige;
}
.page-link:hover {
  background-color: #b14713;
  color: white;
}

@media (min-width: 1200px) {
  .col-xl {
    flex: 0 0 14.2%;
  }
}
.card {
  margin: 2px 0px 1px 0px;
  overflow: hidden;
}
.card-body {
  padding: 0px;
}
.card:hover {
  box-shadow: 5px 10px 20px 1px rgba(0, 0, 0, 0.273) !important;
}
.card-img-top:hover {
  transform: scale(1.1);
}
.overflow {
  overflow: hidden;
}

.pagination {
  display: flex;
  flex-wrap: wrap;
  max-width: max-content;
  padding: 5px 10px;
  margin: 10px auto;
  border-radius: 0;
  background-color: #eff0f6;
}
.page-item {
  margin: 5px;
}

.next .page-link,
.previous .page-link {
  text-shadow: 1px 1px 2px white, 0 1px 2em white, 0 0 0.5em white;
  color: #4fa517;
  background-color: #eff0f6;
  border: 0;
}
```

```
.next .page-link:hover,
.previous .page-link:hover {
  background-color: #b14713;
  color: white;
}

@media (max-width: 600px) {
  .page-item {
    margin: 0;
    padding: auto;
  }
}
.nav-item > a {
  color: rgb(83, 83, 83);
  text-decoration: none;
}
.nav-item > a:hover {
  color: rgb(49, 49, 49);
  text-decoration: none;
}
@media (min-width: 780px) {
  .footer-navbar-nav-one {
    margin-bottom: -2.5rem;
  }
}

.nav-link.nav-item {
  padding-right: 25px;
  justify-content: center;
}

@media (min-width: 700px) {
  .nav-item > a {
    padding-right: 20px;
  }
}
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: "Lato", sans-serif;
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
    monospace;
}
.wrapper {
  padding-top: 5rem;
  margin: 0 auto;
  width: 600px;
}
.rm {
  width: 25px;
  height: 25px;
  background-size: contain;
  background-image: url(./img/Minus.png);
  background-repeat: no-repeat;
  border-radius: 50%;
  margin: 0.1rem;
  border: none;
```

```
}
.rm1 {
  width: 20px;
  height: 20px;
  background-size: contain;
  background-image: url(./img/Plus.png);
  background-repeat: no-repeat;
  border-radius: 50%;
  margin: 0.1rem;
  border: none;
}

.rn {
  width: 28px;
  height: 28px;
  background-image: url(./img/Plus.png);
  background-repeat: no-repeat;
  border-radius: 50%;
  margin: 0.1rem;
  border: none;
}

.rq {
  width: 15px;
  height: 15px;
  background-size: contain;
  background-image: url(./img/Minus.png);
  background-repeat: no-repeat;
  border-radius: 50%;
  border: none;
}

.rq1 {
  width: 15px;
  height: 15px;
  background-size: contain;
  background-image: url(./img/Minus.png);
  background-repeat: no-repeat;
  border-radius: 50%;
  border: none;
}

.r {
  margin: -1rem 0 3rem 1rem;
  font-size: 10pt;
}
.u {
  color: #800080;
}

.done {
  text-decoration: line-through;
}

.about_right {
  display: flex;
  flex-flow: row nowrap;
  align-items: center;
}
.about_right .text_wrap {
  text-align: left;
```

```
}

.text_wrap {
  padding: 0 20px 0 0;
  flex: 0 1 85%;
}

.image_wrap {
  flex: 0 0 auto;
  overflow: hidden;
}

.home,
.reports,
.products,
.team,
.reports {
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 3rem;
}
```

Footer.js

```
import React from "react";
import { Container, Row, Col, Nav, Navbar, Image } from "react-bootstrap";
import Arrow from "../../img/arrow.svg";
import img from "../../img/logo.png";
import { Link } from "react-router-dom";
export default function Footer() {
  return (
    <Container fluid style={{ backgroundColor: "lightgray" }}>
      <Row style={{ justifyContent: "center" }}>
        <Col sm={6}>
          <Navbar
            style={{
              fontSize: "18px",
              fontWeight: "bold",
              justifyContent: "center",
            }}
            className="footer-nav"
            expand="sm"
          >
            <Navbar.Toggle
              style={styleToggle}
              className="d-sm-none"
              aria-controls="footer-navbar-nav-one"
            >
              <Navbar.Brand as="a">Navigation</Navbar.Brand>
              <Image
                src={Arrow}
                alt="image"
                height="20"
                width="20"
                style={{ margin: "auto 0" }}
              />
            </Navbar.Toggle>
```

```
          <Navbar.Collapse
            style={{ justifyContent: "center" }}
            className="footer-navbar-nav-one"
          >
            <Nav as="Navbar" className="flex-row">
              <Nav.Item className="nav-link">
                <Link to="/">Home</Link>
              </Nav.Item>
              <Nav.Item className="nav-link">
                <Link to="/market">Market</Link>
              </Nav.Item>
              <Nav.Item className="nav-link">
                <Link to="/contact">Contact</Link>
              </Nav.Item>
              <Nav.Item className="nav-link">
                <Link to="/about">About us</Link>
              </Nav.Item>
              <Nav.Item className="nav-link">
                <Link to="/">Home</Link>
              </Nav.Item>
            </Nav>
          </Navbar.Collapse>
        </Navbar>
      </Col>
    </Row>

    <div class="about_right">
      <Image src={img} height="180px" />
      <div class="text_wrap">
        <b>MarketAnalazer </b>
        <a>
          — каталог цін на товари. Наше завдання - допомогти відслідковувати
          динаміку цін, що дозволить швидко і зручно переглядати зміну ціни
          необхідного товару тієї чи іншої категорії, у зручному для
          користувача вигляді. В каталозі можна знайти - порівняння товарів,
          опису, пошук товару за назвою, відгуки користувачів, фотогалереї
          товарів.
        </a>
      </div>
    </div>
    <div></div>
  </Container>
 );
}

let styleToggle = {
 display: "flex",
 flex: 1,
 justifyContent: "center",
 border: "0",
};
```

Header.js

```
import React from "react";
import Navibar from "./Navbar.js";
```

```
import SearchForm from "../forms/SearchForm";

export default function Header(props) {
  return (
    <>
      <Navibar
        setSearchItem={props.setSearchItem}
        token={props.token}
        setToken={props.setToken}
      />
      <SearchForm setSearchItem={props.setSearchItem}></SearchForm>
    </>
  );
}
```

Navbar.js

```
import { React, useState } from "react";
import im from "../../img/logo.png";
import RegistrationForm from "../forms/RegistrationForm";
import LoginForm from "../forms/LoginForm";
import { Nav, Button, Navbar, Container, Image } from "react-bootstrap";
import { Link } from "react-router-dom";

export default function Navibar(props) {
  const [expanded, setExpanded] = useState(false);
  const [showRegisterForm, setShowRegisterForm] = useState(false);
  const [showLoginForm, setShowLoginForm] = useState(false);

  const handleShowRegisterForm = () => {
    setShowRegisterForm(true);
  };
  const handleCloseRegisterForm = () => {
    setShowRegisterForm(false);
  };
  const handleLoginForm = () => {
    setShowLoginForm(true);
  };
  const handleCloseLoginForm = () => {
    setShowLoginForm(false);
  };
  const handleCloseNavbar = () => {
    if (expanded !== false) setExpanded(false);
  };
  const logout = (
    <Button
      variant="primary"
      className="m-1"
      onClick={() => {
        localStorage.removeItem("token");
        props.setToken("");
```

```
    }}
  >
    Logout
  </Button>
);

const login = (
  <>
    <Button
      variant="primary"
      onClick={handleShowRegisterForm}
      className="m-1"
    >
      Sign up
    </Button>
    <Button variant="primary" onClick={handleLoginForm} className="m-1">
      Sign in
    </Button>
  </>
);
return (
  <Container fluid className="p-0">
    <Navbar expanded={expanded} expand="md" bg="light" variant="light">
      <Navbar.Brand onClick={() => handleCloseNavbar()}>
        <Link to="/">
          <Image src={im} height="70px" />
        </Link>
      </Navbar.Brand>
      <Navbar.Toggle
        aria-controls="responsive-navbar-nav"
        onClick={() => setExpanded(prev => !prev)}
      />
      <Navbar.Collapse id="responsive-navbar-nav">
        <Nav className="mr-auto">
          <Nav.Item className="nav-link" onClick={() => handleCloseNavbar()}>
            <Link to="/">Home</Link>
          </Nav.Item>
          <Nav.Item className="nav-link" onClick={() => handleCloseNavbar()}>
            <Link to="/market">Market</Link>
          </Nav.Item>
          <Nav.Item className="nav-link" onClick={() => handleCloseNavbar()}>
            <Link to="/contact">Contact</Link>
          </Nav.Item>

          {props.token !== "" ? (
            <Nav.Item
              className="nav-link"
              onClick={() => handleCloseNavbar()}
            >
              <Link to="/profile">Profile</Link>
            </Nav.Item>
          ) : (
            ""
          )}
          <Nav.Item className="nav-link" onClick={() => handleCloseNavbar()}>
            <Link to="/about">About us</Link>
          </Nav.Item>
        </Nav>
        <Nav>{props.token === "" ? login : logout}</Nav>
      </Navbar.Collapse>
    </Navbar>
```

```
      {showRegisterForm === true && (
        <RegistrationForm
          show={showRegisterForm}
          onHide={handleCloseRegisterForm}
        />
      )}
      {showLoginForm === true && (
        <LoginForm
          show={showLoginForm}
          onHide={handleCloseLoginForm}
          setToken={props.setToken}
        />
      )}
    </Container>
  );
}
```

Market.js

```
import { React, useEffect, useState } from "react";
import MainProduct from "./ProductCard";
import PaginationComponent from "./PaginationComponent.js";
import "bootstrap/dist/css/bootstrap.min.css";
import Pagination from "react-js-pagination";
import items from "../../db/main.json";

export default function Main(props) {
  const [posts, setPosts] = useState([]);
  const [loading, setLoading] = useState(false);
  const [currentPage, setCurrentPage] = useState(1);
  const [postsPerPage, setPostsPerPage] = useState(9);

  useEffect(() => {
    const fetchPosts = async () => {
      setLoading(true);
      const res = items.products;
      setPosts(res);
      setLoading(false);
    };
    fetchPosts();
  }, []);

  ///   Get current posts
  const indexOfLastPost = currentPage * postsPerPage;
```

```
  const indexOfFirstPost = indexOfLastPost - postsPerPage;
  const currentPosts = posts.slice(indexOfFirstPost, indexOfLastPost);
  //change page
  const paginate = pageNumber => setCurrentPage(pageNumber);

  return (
    <>
      <MainProduct
        posts={currentPosts}
        searchItem={props.searchItem}
        setSearchItem={props.setSearchItem}
      />

      <Pagination
        activePage={currentPage}
        totalItemsCount={posts.length}
        itemsCountPerPage={postsPerPage}
        onChange={paginate}
        itemClass="page-item"
        linkClass="page-link"
        firstPageText="First"
        lastPageText="Last"
      />
    </>
  );
}
```

ProductCard.js

```
import { React, useEffect, useState } from "react";
import { Card, Container, Row, Col } from "react-bootstrap";
import { Link } from "react-router-dom";

export default function MainProduct(props) {
  useEffect(() => {
    console.log("UE card mount");
    return () => {
      props.setSearchItem("");
      console.log("UE card unmount");
```

```
  };
}, []);

const search = (
  <Row className="">
    {props.posts
      .filter(posts => {
        if (props.searchItem === "") return posts;
        else if (
          posts.title.toLowerCase().includes(props.searchItem.toLowerCase())
        ) {
          return posts;
        }
      })
      .map(posts => (
        <Card
          as={Col}
          md={4}
          lg={3}
          sm={6}
          key={posts.id}
          className="p-0 text-center col-xl"
          style={{}}
        >
          <Card.Link href="#">
            <Link to={`/shop/${posts.id}`}>
              <div className="overflow">
                <Card.Img variant="top" src={posts.img} />
              </div>
            </Link>
          </Card.Link>
          <Card.Body>
            <Card.Link href="#">
              <Card.Title style={BoldText}>{posts.title}</Card.Title>
            </Card.Link>
            <Card.Text className="pt-0 pb-0" style={{ fontSize: "12px" }}>
              {posts.description}
            </Card.Text>
          </Card.Body>
          <Card.Link href="#">
            <p style={BoldText}>{posts.price}</p>
```

```
          </Card.Link>
        </Card>
      ))}
    </Row>
  );

  return (
    <Container fluid className="  mt-1">
      {search}
    </Container>
  );
}
const BoldText = { fontSize: "14px", fontWeight: "bold", color: "#000" };
```

SearchForm.js

```
import React from "react";
import { Form, Button, Col, FormControl, Container } from "react-bootstrap";
import { useHistory } from "react-router-dom";

export default function SearchForm(props) {
  const history = useHistory();

  const navigateTo = () => {
    props.setSearchItem(document.querySelector(".search-input").value);
    document.querySelector(".search-input").value = "";
    history.push("/market");
  };

  return (
    <Container fluid style={{ backgroundColor: "blue", padding: 0 }}>
      <Form className="d-flex">
        <Col xl={10} sm={10} xs={9} className="pl-1">
          <FormControl
            type="text"
            placeholder="Search"
            style={{ margin: "5px" }}
            className="search-input"
          />
```

```
      </Col>
      <Col lx={2} sm={2} xs={3} style={{ paddingLeft: 0 }}>
        <Button
          variant="light"
          style={{
            margin: "5px",
            marginRight: 0,
            overflow: "hidden",
            width: "100%",
          }}
          onClick={() => {
            navigateTo();
          }}>
          <span>Найти</span>
        </Button>
      </Col>
    </Form>
  </Container>
 );
}
```

LoginForm.js

```
import React from "react";
import validate from "../validation/validationLoginForm";
import useForm from "../Hooks/useForm.js";
import InputWarning from "../forms/InputWarning";

import { Form, Button, Modal, Container } from "react-bootstrap";

export default function Login(props) {
 const { handleChange, values, handleSubmit, errors } = useForm(
   submit,
   validate
 );

 function submit() {
   localStorage.setItem("token", "qoifhqwfqwoifhqwfioh");
   props.setToken("qoifhqwfqwoifhqwfioh");
```

```
  props.onHide();
}

return (
  <>
    <Modal centered show={props.show} onHide={props.onHide}>
      <Modal.Header closeButton>
        <Modal.Title className=" pl-4 ml-auto">Sign in</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form onSubmit={handleSubmit}>
          <Form.Group controlId="formLoginEmail">
            <Form.Control
              type="email"
              name="email"
              placeholder="Email address"
              value={values.email}
              onChange={handleChange}
            />
            {errors.email && <InputWarning email={errors.email} />}
          </Form.Group>
          <Form.Group controlId="formLoginPassword">
            <Form.Control
              type="password"
              name="password"
              placeholder="Repeat password"
              value={values.password}
              onChange={handleChange}
            />
            {errors.password && <InputWarning email={errors.password} />}
          </Form.Group>

          <Container className="d-flex justify-content-center">
            <Button variant="primary" size="lg" type="submit">
              Login
            </Button>
          </Container>
        </Form>
      </Modal.Body>
    </Modal>
  </>
```

```
  );
}
```

RegistrationForm.js

```
import { React, useState, useEffect } from "react";
import validate from "../validation/validationRegisterForm.js";
import InputWarning from "./InputWarning";
import useForm from "../Hooks/useForm.js";
import {
  Form,
  Button,
  Modal,
  Row,
  Col,
  Container,
  Alert,
} from "react-bootstrap";

export default function RegisterForm(props) {
  const [isRegister, setIsRegister] = useState(false);
  const { handleChange, values, handleSubmit, errors } = useForm(
    submit,
    validate
  );

  function submit(e) {
    console.log("Registered");
    setIsRegister(true);
  }

  useEffect(() => {
    console.log("Start modal form registration");
    return () => console.log("END modal form registration");
  }, []);

  const register = (
    <Modal.Body>
      <Form onSubmit={handleSubmit}>
```

```
<Row>
 <Col sm={6}>
  <Form.Group controlId="formRegisterFirstName">
   <Form.Control
    type="name"
    placeholder="First name"
    name="firstName"
    value={values.firstName}
    onChange={handleChange}
   />
   {errors.firstName && <InputWarning email={errors.firstName} />}
  </Form.Group>
 </Col>
 <Col sm={6}>
  <Form.Group controlId="formRegisterLastName">
   <Form.Control
    type="name"
    name="lastName"
    placeholder="Last name"
    value={values.lastName}
    onChange={handleChange}
   />
   {errors.lastName && <InputWarning email={errors.lastName} />}
  </Form.Group>
 </Col>
</Row>
<Form.Group controlId="formRegisterEmail">
 <Form.Control
  type="email"
  name="email"
  placeholder="Email address"
  value={values.email}
  onChange={handleChange}
 />
 {errors.email && <InputWarning email={errors.email} />}
</Form.Group>
<Form.Group controlId="formRegisterPasswordCreate">
 <Form.Control
  type="password"
  name="password"
  placeholder="Create password"
```

```
          value={values.password}
          onChange={handleChange}
        />
        {errors.password && <InputWarning email={errors.password} />}
      </Form.Group>
      <Form.Group controlId="formRegisterPasswordRepeat">
        <Form.Control
          type="password"
          name="confirmPassword"
          placeholder="Repeat password"
          value={values.confirmPassword}
          onChange={handleChange}
        />
        {errors.confirmPassword && (
          <InputWarning email={errors.confirmPassword} />
        )}
      </Form.Group>

      <Container className="d-flex justify-content-center">
        <Button variant="primary" size="lg" type="submit">
          Register
        </Button>
      </Container>
    </Form>
  </Modal.Body>
);

const registered = (
  <Alert variant={"success"}>
    <Alert.Heading style={{ textAlign: "center" }}>
      Успешная реестрация
    </Alert.Heading>
    <hr />
    <p className="mb-0" style={{ textAlign: "center" }}>
      Теперь вы можете войти в систему
    </p>
  </Alert>
);

return (
  <Modal centered show={props.show} onHide={props.onHide}>
```

```
      <Modal.Header closeButton>
        <Modal.Title className="pl-4 ml-auto">Sign up</Modal.Title>
      </Modal.Header>
      {isRegister === false ? register : registered}
    </Modal>
  );
}
```

Sidebar.js

```
import React, { useState } from "react";
import { DropdownButton, Dropdown, Form } from "react-bootstrap";
import styled from "styled-components";
import { Link } from "react-router-dom";
import * as FaIcons from "react-icons/fa";
import * as AiIcons from "react-icons/ai";
import * as BiIcons from "react-icons/bi";
import SubMenu from "./SubMenu";
import SubMenuS from "./SubMenuS";
import { IconContext } from "react-icons/lib";
import items from "../db/categories.json";
import items1 from "../db/Filters.json";

const Nav = styled.div`
  background: #15171c;
  height: 60px;
  display: flex;
  justify-content: space-between;
  align-items: center;
`;

const NavIcon = styled(Link)`
  margin-left: 2rem;
  font-size: 2rem;
  height: 80px;
  display: flex;
  justify-content: flex-start;
  align-items: center;
`;

const SidebarNav = styled.nav`
  background: #15171c;
  width: 250px;
  height: 1373px;
  display: flex;
  justify-content: center;
  position: absolute;
  top: 0;
  left: ${({ sidebar }) => (sidebar ? "0" : "-100%")};
  transition: 350ms;
  z-index: 10;
`;
const SidebarNavSort = styled.nav`
```

```
  background: #15171c;
  width: 250px;
  height: 1373px;
  display: flex;
  justify-content: center;
  position: absolute;
  top: 0;
  left: ${({ sidebarSort }) => (sidebarSort ? "-100%" : "80.2%")};
  z-index: 10;
`;

const SidebarWrap = styled.div`
  width: 100%;
`;

export default function Sidebar(props) {
  const [sidebar, setSidebar] = useState(false);
  const showSidebar = () => setSidebar(!sidebar);
  const [sidebarSort, setSidebarSort] = useState(true);
  const showSidebarSort = () => setSidebarSort(!sidebarSort);

  return (
    <>
      <IconContext.Provider value={{ color: "#fff" }}>
        <Nav>
          <NavIcon to="#">
            <FaIcons.FaBars onClick={showSidebar} />
          </NavIcon>
          <div
            style={{
              font: '1.4  em "Roboto"',
              color: "#fff",
              display: "flex",
              alignItems: "center",
              paddingRight: "10px",
            }}
          >
            <DropdownButton
              id="dropdown-basic-button"
              variant="dark"
              title="Сортування"
            >
              <Dropdown.Item href="#/action-1">За найменуванням</Dropdown.Item>
              <Dropdown.Item href="#/action-2">
                Від дешевих до дорогих
              </Dropdown.Item>
              <Dropdown.Item href="#/action-3">
                Від дорогих до дешевих
              </Dropdown.Item>
            </DropdownButton>
            <NavIcon to="#">
              <FaIcons.FaFilter onClick={showSidebarSort} />
            </NavIcon>
            Фільтрація
          </div>
        </Nav>
        <SidebarNav sidebar={sidebar}>
          <SidebarWrap>
            <NavIcon to="#">
              <AiIcons.AiOutlineClose onClick={showSidebar} />
            </NavIcon>
```

```
          {items.categories.map((item, index) => {
            return <SubMenu item={item} key={index} />;
          })}
        </SidebarWrap>
      </SidebarNav>
      <SidebarNavSort sidebarSort={sidebarSort}>
        <SidebarWrap>
          <NavIcon to="#">
            <AiIcons.AiOutlineClose onClick={showSidebarSort} />
          </NavIcon>
          {items1.filters.map((item, index) => {
            return <SubMenuS item={item} key={index} />;
          })}
        </SidebarWrap>
      </SidebarNavSort>
    </IconContext.Provider>
  </>
 );
}
```

SubMenu.js

```
import React, { useState } from "react";
import { Link } from "react-router-dom";
import styled from "styled-components";
import * as RiIcons from "react-icons/ri";

const SidebarLink = styled(Link)`
  display: flex;
  color: #e1e9fc;
  justify-content: space-between;
  align-items: center;
  padding: 20px;
  list-style: none;
  height: 60px;
  text-decoration: none;
  font-size: 18px;
  &:hover {
    background: #252831;
    border-left: 4px solid #632ce4;
    cursor: pointer;
  }
`;

const SidebarLabel = styled.span`
  margin-left: 16px;
```

```
`;

const DropdownLink = styled(Link)`
  background: #414757;
  height: 60px;
  padding-left: 3rem;
  display: flex;
  align-items: center;
  text-decoration: none;
  color: #f5f5f5;
  font-size: 18px;
  &:hover {
    background: #632ce4;
    cursor: pointer;
  }
`;

const SubMenu = ({ item }) => {
  const [subnav, setSubnav] = useState(false);

  const showSubnav = () => setSubnav(!subnav);

  return (
    <>
      <SidebarLink to={item.path} onClick={item.subNav && showSubnav}>
        <div>
          <SidebarLabel>{item.title}</SidebarLabel>
        </div>
        <div>
          {item.subNav && subnav ? (
            <RiIcons.RiArrowUpSFill />
          ) : item.subNav ? (
            <RiIcons.RiArrowDownSFill />
          ) : null}
        </div>
      </SidebarLink>
      {subnav &&
        item.subNav.map((item, index) => {
          return (
            <DropdownLink to={item.path} key={index}>
              <SidebarLabel>{item.title}</SidebarLabel>
```

```
          </DropdownLink>
        );
       })}
    </>
  );
};

export default SubMenu;
```

Home.js

```
import React from "react";
import { Container } from "react-bootstrap";
import Sidebar from "../bar/Sidebar";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import { Reports, ReportsOne, ReportsTwo, ReportsThree } from "../page/Reports";
import MainProduct from "./HomeCard";
import items from "../db/main.json";
export default function Home(props) {
  return (
    <Container fluid className="p-0" style={{ Width: 1250 }}>
      <Router>
        <Sidebar />
        <Switch>

          <Route path="/reports" exact component={Reports} />
          <Route path="/reports/reports1" exact component={ReportsOne} />
          <Route path="/reports/reports2" exact component={ReportsTwo} />
          <Route path="/reports/reports3" exact component={ReportsThree} />
        </Switch>
      </Router>
      <MainProduct
        items={items}
        searchItem={props.searchItem}
        setSearchItem={props.setSearchItem}
      />
    </Container>
  );
}
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

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