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 *Файнштейна Данііла Володимировича*
(ПІБ)

академічної групи *122-19-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованого додатку для інтернет магазину оптики з використанням бібліотеки React*

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

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-3* *Файнштейна Д. В.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого додатку для інтернет магазину оптики з використанням бібліотеки React*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Спірінцев В.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Файнштейн Д. В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12,06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 107 с., 50 рис., 3 дод., 25 джерел.

Об'єкт розробки: веб-орієнтований додаток для інтернет магазину оптики з використанням бібліотеки React.

Мета кваліфікаційної роботи: розробка веб-орієнтованого додатку для інтернет магазину оптики з використанням бібліотеки React, який би підвищив ефективність його роботи за рахунок діяльності в сфері електронної комерції.

У вступі проводиться аналіз та досліджується поточний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, робиться обґрунтування актуальності теми і формується постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-орієнтованої інформаційної системи, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні веб-орієнтованого додатку, що забезпечує: можливість ефективної роботи інтернет-магазину, тобто поліпшує зв'язок з клієнтами та розширює аудиторії за рахунок можливості здійснювати торгівлю через Інтернет.

Актуальність розробки веб-орієнтованого додатку для діяльності компанії в сфері електронної комерції є значною для українського ринку, оскільки розробка інтернет-магазину, особливо у сучасних умовах війни, може допомогти компаніям залучати нових клієнтів та розширювати свою аудиторію за рахунок використання цифрових маркетингових інструментів, що є особливо актуальним у період економічної нестабільності.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРНЕТ-МАГАЗИН, БАЗА ДАНИХ, REACT, JAVASCRIPT, SPRING, SPRING BOOT, HTML, CSS, BOOTSTRAP, TENSORFLOW, MYSQL, AXIOS API, REST, ЕЛЕКТРОННА КОМЕРЦІЯ, ВІДСТЕЖЕННЯ ОБЛИЧЧЯ.

ABSTRACT

Explanatory note: 107 p., 50 figures, 3 apps, 25 sources.

Development object: a web-oriented application for an online optical store using the React library.

The purpose of the qualification work: the development of a web-oriented application for an online optical store using the React library, which would increase the efficiency of its work due to activities in the field of e-commerce.

In the introduction, an analysis is carried out and the current state of the problem is investigated, the purpose of the qualification work and the field of its application are specified, the relevance of the topic is substantiated, and the statement of the task is formed.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second section, available solutions are analyzed, a platform for development is chosen, the design and development of a web-oriented information system is performed, the operation of the system is described, the algorithm and structure of its functioning, as well as the call and download of the application, the input and output data are determined, the composition of the parameters of the technical means is characterized.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating the application is calculated, and the time for its creation is calculated.

The relevance of developing a web-based application for a company's e-commerce activities is significant for the Ukrainian market, since the development of an online store, especially in the modern conditions of war, can help companies attract new customers and expand their audience through the use of digital marketing tools, which are especially relevant in a period of economic instability.

Keywords List: INFORMATION SYSTEM, ONLINE STORE, DATABASE, REACT, JAVASCRIPT, SPRING, SPRING BOOT, HTML, CSS, BOOTSTRAP, TENSORFLOW, MYSQL, AXIOS API, REST, ELECTRONIC COMMERCE, FACE TRACKING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ -	Програмне забезпечення
ІС -	Інформаційна система
ОС -	Операційна система
БД -	База даних
СУБД -	Система управління базами даних
РСУБД -	Реляційна система управління базами даних
CMS -	Система управління контентом
CSS -	Cascading Style Sheets
SQL -	Structured Query Language
HTML -	Hyper Text Markup Language
JS -	JavaScript
ПК -	Персональний комп’ютер
DI	Dependency Injection
3D	3-dimensional

ВСТУП

Стрімкий розвиток інформаційних технологій і систем сприяє активному розвитку електронної комерції в багатьох сферах підприємницької діяльності, що призведе до суттєвих змін у взаємодії суб'єктів бізнесу. За останні десятиліття електронна комерція стала дедалі більш популярною в багатьох сферах підприємницької діяльності. Інтернет магазини дозволяють компаніям ефективно продавати свої продукти та послуги, а клієнтам – зручно та швидко здійснювати покупки онлайн. Це призводить до суттєвих змін у взаємодії суб'єктів бізнесу. Наприклад, відкриття інтернет магазину може дозволити компанії працювати з клієнтами з усієї країни та навіть з-за кордону. Крім того, розробка веб-орієнтованої інформаційної системи може значно спростити та автоматизувати процеси продажу, замовлення та доставки товарів.

Завдяки інтернет магазинам, бізнес може забезпечити своїм клієнтам більш зручний та швидкий спосіб покупки, що може призвести до збільшення продажів та залучення нових клієнтів.

Актуальність розробки веб-орієнтованого додатку для діяльності компанії в сфері електронної комерції полягає в тому, що:

– На ринку України у сучасних умовах війни актуальність у інтернет-магазинах є значною, так як війна може вплинути на бізнес-середовище та викликати певні складнощі для фізичних магазинів, зокрема через обмеження на пересування, зменшення кількості покупців та зниження настроїв споживачів. Проте, розробка та розвиток веб-орієнтованого додатку може стати спасінням для бізнесу у складних умовах війни.

– Значною мірою на цей веб-орієнтований додаток впливає інструментарій, який дозволяє користувачеві віртуально приміряти окуляри у режимі реального часу з відстежуванням обличчя, що полегшує вибір товару.

– Веб-орієнтований додаток, розроблений у межах цієї кваліфікаційної роботи, має зручний та простий інтерфейс, що допомагає користувачам, які мало знайомі з комп'ютером отримати кращий досвід у його

використані. Зважаючи на те, що додаток був написаний з використанням сучасних та популярних засобів розробки, таких як React та Spring Boot, він буде мати довгострокову та якісну технічну підтримку

Об'єктом дослідження є веб-орієнтований додаток для інтернет магазину оптики з використанням React.

Мета кваліфікаційної роботи: розробка веб-орієнтованого додатку для інтернет магазину оптики з використанням бібліотеки React, який би підвищив ефективність його роботи за рахунок діяльності в сфері електронної комерції.

Для вирішення поставленої мети необхідно вирішити наступні задачі:

– розглянути особливості електронної комерції в контексті розвитку національної економіки;

– визначити основні вимоги, які повинні бути враховані при розробці програмного продукту. Ці вимоги охоплюють функціональні характеристики, інформаційну безпеку, склад і параметри технічних засобів, інформаційну та програмну сумісність, а також зміст інформаційного наповнення системи;

– розробити веб-орієнтований додаток та описати його структуру, наповнення та алгоритми взаємодії;

– здійснити розрахунок трудомісткості та витрат на розробку програмного продукту.

Практичне значення полягає у створенні веб-орієнтованого додатку, що забезпечує: можливість ефективної роботи інтернет-магазину, тобто поліпшує зв'язок з клієнтами та розширює аудиторії за рахунок можливості здійснювати торгівлю через Інтернет. Забезпечує процеси замовлення, збереження даних клієнтів, управління товарами. Крім того, розроблена система дозволить забезпечити візуальну оцінку товару, віртуально приміряючи його, зручний та швидкий пошук необхідних товарів для клієнтів, надання повної інформації про товари, систему знижок для збільшення продажів. В результаті, створення такої системи дозволить збільшити конкурентоспроможність компанії та забезпечити її успішну роботу в електронному бізнесі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Особливості електронної комерції у сучасних умовах України

За останні роки спостерігається збільшення кількості людей на Україні, які обирають електронну комерцію для здійснення покупок. Це пов'язано зі зручністю, доступністю та широким асортиментом товарів та послуг, доступних онлайн. А з появою багатьох інтернет магазинів виникає зростаюча конкуренція. Компанії повинні працювати над залученням та утриманням клієнтів шляхом пропозицій цікавих товарів, зручних сервісів та конкурентних цін.

Електронна комерція дозволила компаніям розширити свої ринки, працюючи зі споживачами з різних регіонів країни або навіть за її межами. Це дає можливість досягти нових клієнт.

Але електронна комерція українських підприємств у сучасних умовах, у контексті війни, має свої особливості. З одного боку, зростання попиту на інтернет-торгівлю та збільшення кількості клієнтів може стати для бізнесу стимулом до розвитку та збільшення прибутку. З іншого боку, економічна нестабільність, високі ціни на енергоносії та інші ресурси, а також несприятливі фактори, пов'язані зі збройним конфліктом на сході країни, можуть ускладнити діяльність підприємств, що займаються електронною комерцією. На такому ринку необхідно вміти швидко реагувати на зміни у попиті та забезпечувати якісний сервіс, що становить виклик для підприємців, що діють у цій галузі.

Шляхи вирішення проблем електронної комерції у умовах війни на Україні можуть включати:

– компанії повинні звернути особливу увагу на забезпечення безпеки своїх інформаційних систем та захисту персональних даних клієнтів. Це може

включати використання надійних систем шифрування, захисту від кібератак та регулярне оновлення програмного забезпечення;

– умови війни можуть створювати перешкоди для доставки товарів, особливо у зоні конфлікту. Компанії повинні шукати альтернативні шляхи доставки, співпрацювати з надійними логістичними партнерами та враховувати можливі затримки та обмеження;

– компанії можуть розглядати можливість співпраці з локальними постачальниками та виробниками товарів. Це може допомогти зменшити залежність від імпорту та забезпечити постачання товарів навіть у важких умовах війни;

– компанії повинні бути готові до швидких змін на ринку та адаптуватися до нових умов. Це може включати перегляд асортименту товарів, перерозподіл рекламного бюджету, залучення нових цільових аудиторій та розробку конкурентних пропозицій;

– важливо зберігати комунікацію з клієнтами та надавати їм інформацію про можливі затримки у доставці або зміни в роботі компанії.

1.1.2. Аналіз готових існуючих рішень

Для аналізу обрано сайти Люксоптика [1] та Dnipro_Optika [2], які посвячені торгівлі оптикою. Зображення дизайну головних сторінок цих сайтів показані на рис. 1.1 – рис. 1.2.

Сайти мають привабливий та сучасний дизайн з гарним підбором кольорів та використанням візуальних елементів для зручності користувача.

На обох сайтах наявна головна сторінка, яка включає в себе :

– інформаційні банери;

– популярні продукти;

– наявні бренди.

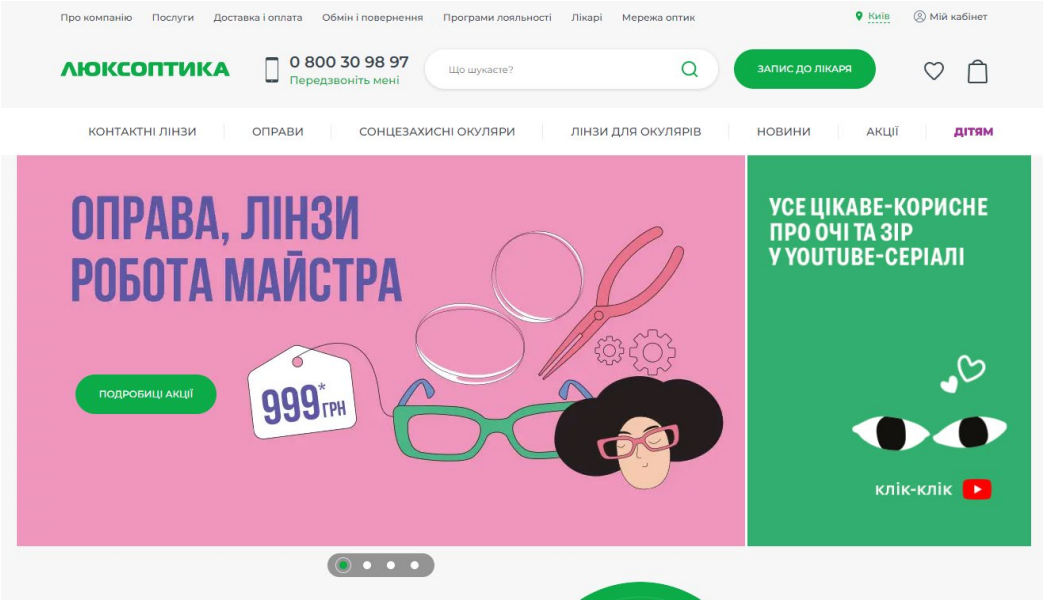

Рис. 1.1. Банери на сайті Люксоптика

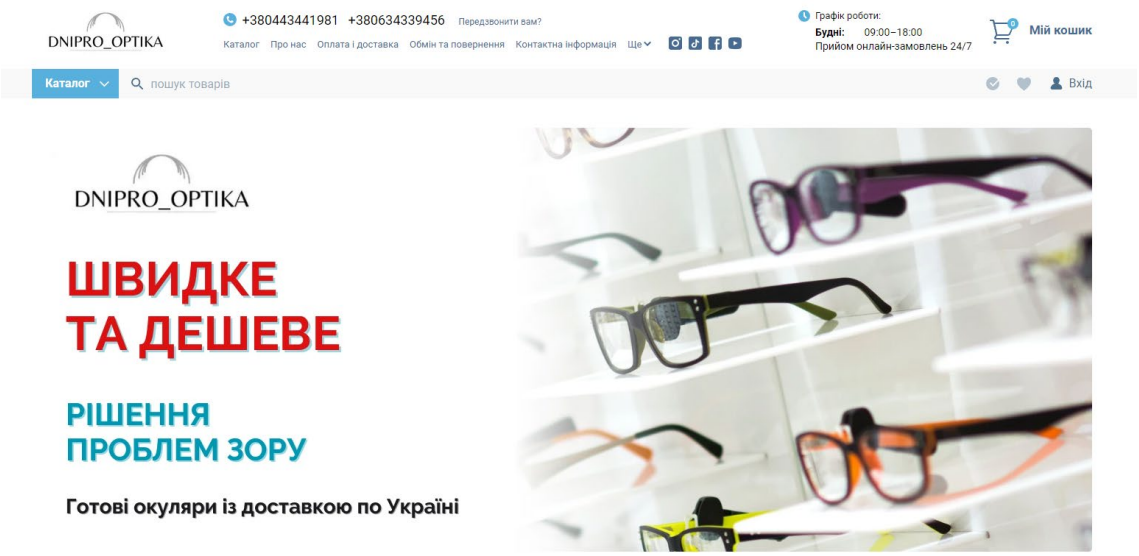

Рис. 1.2. Банери на сайті Dnipro_Optika

«Хедери» складаються з логотипу, категорій товарів, вибору мови інтерфейсу, корзини покупця, контактної інформації, навігації. Головне меню на сайті розташоване вгорі, де можна знайти відповідні категорії та підкатегорії. Дуже легко орієнтуватися на сайті та швидко знайти необхідний товар. «Футери» містять контактну інформацію, зворотній зв'язок та політику сайту.

Каталог (рис. 1.3 – рис. 1.4) складається з «карток» товарів з описом найменування та їх ціни. Збоку наявна панель фільтрів за характеристиками.

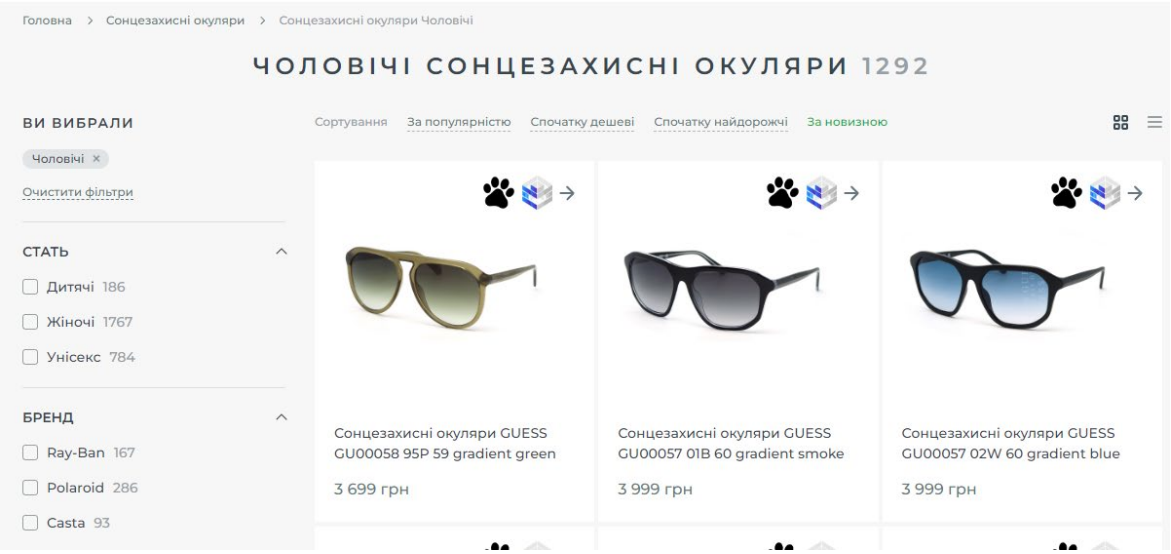

Рис. 1.3. Каталог на сайті Люксоптика

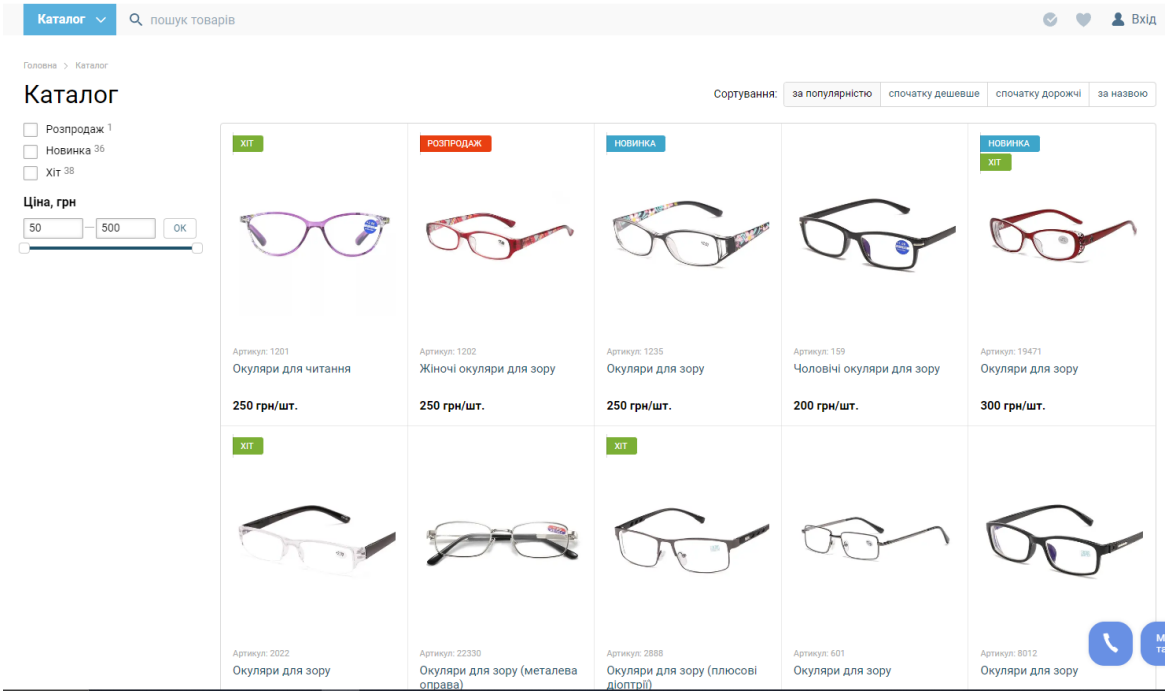

Рис. 1.4. Каталог на сайті Dnipro_Optika

Отже, аналіз цих технічних рішень показав, що для обох існуючих систем притаманний такий набір компонентів: головна сторінка, «хедер» та «футер» з навігацією та описом інформації про сайт, каталог товарів з фільтрами, інформаційна сторінка товару, механізм авторизації користувачів, можливість замовлення товару, локалізація.

Однак, аналізуючи цей набір даних був виявлений недолік, який полягає в тому, що в існуючих системах відсутній механізм віртуального ознайомлення з товаром. Тому, у межах цієї кваліфікаційної роботи, прийнято рішення про розробку інструментарію для користувача, який надав би змогу візуально оцінити товар, приміряючи його безпосередньо на сайті. Цей функціонал реалізований двома способами:

– з використанням фото користувача. В даному випадку клієнт завантажує своє фото на сайт та отримує зображення окулярів, яке він має змогу перетягнути на своє фото та за необхідністю збільшити чи зменшити масштаб окулярів;

– з відстеженням положення обличчя користувача у режимі реального часу. В даному випадку користувач дає дозвіл на використання сайтом своєї камери та наводячи її на обличчя отримує зображення окулярів, яке автоматично накладається на нього та масштабується в залежності від відстані до камери.

1.2. Призначення розробки та галузь застосування

У межах цієї кваліфікаційної роботи розглядається веб-орієнтована інформаційний додаток для діяльності інтернет магазину оптики «Vision». Ця система відповідно застосовується у галузі електронної комерції.

Причиною появи веб-додатку оптичного магазину є необхідність створити зручний і доступний спосіб для користувачів переглядати та купувати оптичні товари через Інтернет. Веб-додаток для магазину оптики надає багато переваг як клієнтам, так і самому магазину. Деякі з цих причин включають:

– веб-додаток дозволяє клієнтам робити покупки оптичних виробів з будь-якого місця в будь-який час. Вони можуть переглядати асортимент, вибирати товари та оформляти замовлення зручним для них способом;

– має інструменти для забезпечення кращого вибору товару;

– UI взаємодіє з API, що робить цей проект універсальним та гнучким;

– веб-додаток оптичного магазину може запропонувати широкий вибір продуктів. Клієнти можуть легко переглядати різні варіанти, порівнювати ціни та приймати зважене рішення про покупку;

– веб-додаток дозволяє клієнтам купувати оптичні товари зручним способом без необхідності відвідувати звичайний магазин. Вони можуть замовити товари онлайн і вибрати зручний для них спосіб доставки;

– веб-додаток дозволяє магазину оптимізувати процеси управління та оновлення асортименту, цін, акцій тощо. Надає можливість швидко реагувати на зміни кон'юнктури ринку та потреб покупців.

Усі ці фактори роблять програму інтернет-магазину оптики ефективним і зручним інструментом для продажу оптичних товарів і задоволення потреб клієнтів.

Розроблена інформаційна система призначена для:

– забезпечення доступу до якісних та доступних оптичних виробів в онлайн-режимі;

– забезпечення зручності та ефективності покупки оптичних виробів. Клієнти можуть швидко та легко знайти необхідні вироби, зробити замовлення, скористатися знижками та віртуально приміряти товар;

– забезпечення інструментів керування магазином адміністратором;

– дозволяє замовити та придбати продукцію зручним для клієнта способом.

Веб-орієнтований додаток інтернет магазину оптики повинен забезпечувати виконання таких функції як:

– підтримка каталогу товарів;

– обробка клієнтських замовлень онлайн;

– зрозумілий інтерфейс.

Проведений аналіз існуючих технічних рішень сформував основні вимоги, щодо сучасного інтернет магазину:

– орієнтованість на клієнта;

– наявність каталогу товарів в інтернет-магазині оптики. Швидкий та зручний пошук товару за категоріями;

– зберігання даних у базі даних магазину;

– додавання товарів в кошик покупця та оформлення замовлення;

– обробка замовлень та перегляд їх у персональному кабінеті користувача;

– інструменти, які забезпечують візуальну оцінку товару клієнтом, віртуально приміряючи його.

Аналізуючи вищезазначені вимоги, було зроблено висновок про те, що обов'язково буде реалізовано в проекті:

– головна сторінка веб-магазину має вітрину з привабливими пропозиціями та акційні товари;

– персональний кабінет користувача;

– управління замовленнями;

– каталог товарів має фільтрацію за необхідними категоріями;

– кольори товару можна перемикати зі сторінки самого товару;

– локалізація інтерфейсу користувача;

– наявна система знижок;

– на сторінці продукту є інструмент для примірювання товару онлайн за фото чи у режимі реального часу з відстежуванням обличчя;

– наявні інструменти керування магазином для адміністратора.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

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

– ОПП за спеціальністю 122 «Комп’ютерні науки»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;

– завдання на дипломний проект на тему «Розробка веб-орієнтованого додатку для інтернет магазину оптики з використанням бібліотеки React».

1.4. Постановка завдання

В даній кваліфікаційній роботі розглядається створення веб-орієнтованого додатку для Інтернет магазину оптики.

Мета цієї роботи: розробка веб-орієнтованого додатку для інтернет магазину оптики з використанням бібліотеки React, який би підвищив ефективність його роботи за рахунок діяльності в сфері електронної комерції.

Інформаційна система створюється для надання електронно-комерційних послуг з продажу окулярів. Аудиторію веб-орієнтованого додатку становлять люди будь-якого віку. Тематикою веб-орієнтованого додатку є реалізація продажу оптику через мережу «Інтернет».

Веб-орієнтований додаток повинен задовольняти наступним основним вимогам: висока швидкість завантаження сторінок; максимальна зручність роботи із системою для користувача; оптимізація сторінок під пошукові системи; легкість сприйняття інформації; простота й повнота управління змістом.

Створення й розробка додатку повинна включати наступні етапи:

– аналіз вимог та обґрунтування необхідності: визначення функціональної та нефункціональної вимог до додатку, а також визначення його призначення та потенційної аудиторії;

– проектування: складання технічного завдання, створення концепції та дизайну інтерфейсу додатку, розробка архітектури та вибір технологій;

– розробка: програмування функціональності, розробка бази даних, тестування та відлагодження додатку;

– підтримка та розвиток: надання технічної підтримки, оновлення та покращення функціональності додатку згідно з вимогами та потребами користувачів.

Кожен з цих етапів має велике значення для успішної розробки додатку та його подальшого ефективного функціонування.

Головна сторінка першою постає перед користувачем, тому її дизайну та функціональності необхідно приділити чимало уваги, щоб надати змогу забезпечити користувача достатньою кількістю необхідної інформації, співвіднесеною з реальним продавцем-консультантом.

На рис. 1.5. зображено макет головної сторінки за допомогою сервісу «Figma».

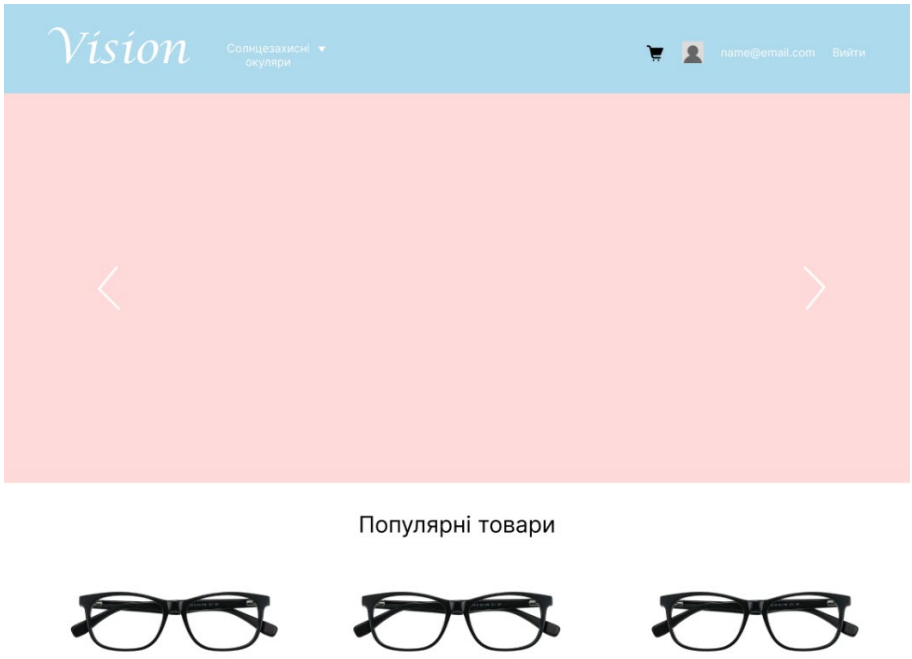

Рис.1.5. Макет головної сторінки інформаційної системи

Головна сторінка містить:

– «хедер», на якому розміщуються логотип, меню та поле авторизації/реєстрації. Якщо користувач авторизований, з’являється поле з його інформацією та посиланням на персональний кабінет;

– компонент «карусель» з пропозиціями;

– товари, які користуються найбільшим попитом;
– акційні пропозиції;
– «футер» з інформацією про компанію-власник.

Відповідно до макету головної сторінки було здійснено її верстку із застосуванням React, HTML, CSS та Bootstrap.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставленої в роботі мети в інформаційній системі, що розробляється, повинні бути реалізовані:

– сучасні технології для розробки;
– можливість отримати онлайн послуги з закупівлі оптики;
– легкий пошук необхідних товарів;
– кольори товару можна перемикати зі сторінки самого товару;
– вхідні дані користувача валідуються;
– персональний кабінет користувачів;
– можливість приміряти товар онлайн за фото чи у режимі реального часу з відстежуванням обличчя;
– локалізація інтерфейсу користувача;
– наявна система знижок;
– інструменти керування магазином адміністратором;
– простий та інтуїтивно зрозумілий для користувача інтерфейс.

Користувач контактує з сайтом шляхом реєстрації персонального акаунту, переглядання наявних товарів, замовлення товару. У відповідь отримую швидке задоволення свої вимог не виходячи з дому.

1.5.2. Вимоги до інформаційної безпеки

Під терміном "інформаційна безпека" розуміється стан, в якому системи обробки та зберігання даних забезпечують конфіденційність, доступність та цілісність інформації, а також використовуються та розвиваються в інтересах громадян, суспільства та держави. Це досягається шляхом впровадження комплексу заходів, спрямованих на захист інформації від несанкціонованого доступу, використання, розголошення, пошкодження, внесення змін, ознайомлення, перевірки запису або знищення. Реалізовані критерії безпеки системи:

– валідація вхідних даних;

– повідомлення про помилки;

– пароль користувача шифрується за допомогою кодувальнику BCrypt із параметром «сили» - 10;

– користувачі поділяються на ролі, задля розподілення доступу;

– заради захищеного зберігання даних користувача та даних магазину, з серверу передається лише максимально необхідна інформація.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування веб-орієнтованого додатку, повинні виконуватися певні вимоги до технічних засобів.

Для клієнтської частини знадобляться:

– інтернет браузер (Microsoft Internet Explorer, MozillaFireFox, Opera, Google Chrome);

– доступ до мережі Інтернет;

– пристрої вводу (клавіатура, комп’ютерна миша).

Для серверної частини:

– Windows XP, Windows 7 (32/64 bit) або вище;

– оперативна пам'ять: мінімум 8 Гб;

– 10 GB вільного дискового простору;

– принаймні один інтернет браузер, наприклад Chrome, Firefox, Microsoft Edge;

– Node.js;

– мінімальна швидкість активного підключення до Інтернету 512 Кбіт/с і вище.

Для серверу бази даних:

– процесор: Intel Core або Xeon 3ГГц (або Dual Core 2ГГц) чи подібний їм AMD процесор;

– оперативна пам'ять: 4 Гб;

– графічні прискорювачі: nVidia або ATI з підтримкою OpenGL 1.5 або вище.

Технічні характеристики, наведені вище, є рекомендованими. Це означає, що якщо в наявності є технічні засоби, які відповідають або перевищують зазначені характеристики, розроблений програмний продукт буде працювати з надійністю, швидкістю обробки даних і безпекою, що вимагається замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати веб-орієнтована підсистема, відповідало наступним вимогам:

– операційна система Unix, Linux, Microsoft Windows XP/7/8/10/11;

– інтернет браузер (Microsoft Internet Explorer, MozillaFireFox, Opera, Google Chrome).

Для створення системи на основі React знадобиться мінімальна конфігурація, яка наведена нижче:

– Windows XP, Windows 7 (32/64 bit) або вище;

– оперативна пам'ять: мінімум 4 Гб;

– 10 GB вільного дискового простору;

– принаймні один інтернет браузер, наприклад Chrome, Firefox, Microsoft Edge;

– Node.js;

– мінімальна швидкість активного підключення до Інтернету 512 Кбіт/с і вище;

– принаймні один встановлений редактор коду для тестування та налагодження коду, наприклад: Atom, Sublime, Visual studio code.

Для створення системи на основі Spring конфігурація складає:

– Java 8 або вище;

– Spring Framework 5.0.4.RELEASE або вище;

– підтримка збірки надається для Maven 3.2+ і Gradle 4.

Spring Boot підтримує такі сервлет контейнери:

– Tomcat 8.5;

– Jetty 9.4;

– Undertow 1.4.

Вимоги до обладнання та програмного забезпечення для Java:

– оперативна пам'ять: мінімум 128 Мб;

– дисковий простір: 124 МБ для JRE, 2 МБ для оновлення Java;

– процесор: мінімум процесор Pentium 2 266 МГц.

Веб-орієнтований додаток має бути реалізовано на мові програмування Java для серверної частини, HTML5 і CSS для верстання веб-сторінок, бібліотеки React для написання інтерфейсу користувача, з використанням веб-серверу Apache Tomcat 7 та СУБД MySQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Веб-орієнтований додаток для інтернет магазину оптики призначений для комерційної діяльності онлайн. Виходячи з цього функціональне призначення цієї інформаційної системи включає наступне:

– ведення бази даних магазину;

– відображення товарів в магазині інтернет магазину оптики;

– додавання товарів в кошик покупця та оформлення замовлення;

– додавання, перегляд або видалення обраних товарів з кошику;

– обробка замовлень та перегляд їх у персональному кабінеті користувача;

– локалізація інтерфейсу користувача;

– авторизація та аутентифікація користувачів;

– підтримка різних методів оплати та доставки;

– система знижок;

– інструмент для зручного вибору окулярів онлайн за фото чи у режимі реального часу з відстежуванням обличчя;

Експлуатаційне призначення системи включає:

– зручний та простий інтерфейс для користувачів, що забезпечує швидкий та зручний пошук та замовлення товарів;

– використання Rest API, що збільшує адаптивність додатку під інші платформи;

– автоматизоване управління запасами та замовленнями, що зменшує ризик виникнення помилок та забезпечує ефективність управління;

– збільшення обсягів продажів через можливість розширення аудиторії, зокрема через залучення клієнтів з інших регіонів;

– можливість швидкої обробки замовлень, що забезпечує задоволеність клієнтів та позитивний імідж бренду;

– наявність інструментів керування магазином у адміністраторській частині;

– зниження витрат на обслуговування клієнтів через автоматизацію процесів замовлення товарів.

2.2. Опис застосованих математичних методів

У процесі розробки веб-орієнтованого додатку були використані формули розрахунку відстані між двома точками та пропорційної зміни розміру зображень. Ці математичні формули були використані для налаштування розмірів зображення окулярів для накладання на обличчя користувача у режимі реального часу.

Формула розрахунку відстані між двома точками була використана для визначення ширини зображення окулярів, які накладаються на обличчя користувача:

$$\text{B} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \tag{2.1}$$

де B – відстань між двома точками (ширина зображення),

x_1 та y_1 – координати першої точки,

x_2 та y_2 – координати другої точки.

Формула пропорційної зміни розміру зображень використана задля зміни масштабу зображення окулярів в залежності від відстані користувача до камери у режимі реального часу:

$$\frac{H_n}{H_o} = \frac{W_n}{W_o}, \tag{2.2}$$

$$\text{звідки знайдено } H_n = \frac{H_o * W_n}{W_o}, \tag{2.3}$$

де H_n та H_o – відповідно нова та стара висота зображення, тобто висота зображення змінюються від H_o до H_n,

W_n та W_o – відповідно нова та стара ширина зображення, тобто ширина зображення змінюються від W_o до W_n.

2.3. Опис використаних технологій та мов програмування

Для реалізації веб-орієнтованого додатку було використано наступні мови програмування:

– Java для «backend»;
– JavaScript для «frontend»;
– SQL для бази даних.

Серед технологій було використано:

– фреймворк Spring (Spring Boot, Spring Boot JPA);
– бібліотека React;
– база даних MySQL;
– TensorFlow;
– Axios API;
– Bootstrap.

2.3.1. Огляд використаних мов програмування

Мова програмування Java

Java є потужною мовою програмування, яка має широке застосування в різних галузях. Вона використовується для розробки різноманітних додатків, включаючи настільні та мобільні додатки, обробку великих обсягів даних, вбудовані системи і багато іншого. За даними Oracle, компанії, що розробляє та підтримує Java, ця мова програмування працює на трьох мільярдах пристроїв у всьому світі, що свідчить про її велику популярність і поширеність [3].

Завдячуючи своїй універсальності та безкоштовності, вона дозволяє створювати локалізоване та розподілене програмне забезпечення. Використання Java охоплює різноманітні сфери, включаючи:

– розробка ігор: багато популярних мобільних, комп'ютерних та відеоігор побудовані з використанням Java. Вона використовується як для традиційних ігор, так і для ігор з використанням передових технологій, таких як машинне навчання та віртуальна реальність;

– хмарні обчислення: Java часто називають "Write Once and Run Anywhere" (WORA), оскільки вона може працювати на різних платформах. Це робить її ідеальним вибором для розробки децентралізованих хмарних додатків, які працюють на різних базових системах;

– великі дані: Java використовується для обробки великих обсягів даних в режимі реального часу. Вона надає потужні механізми для роботи зі складними наборами даних та виконання розрахунків над великими обсягами інформації;

– штучний інтелект: Java пропонує багато бібліотек для машинного навчання. Її стабільність та швидкодія роблять її ідеальним інструментом для розробки додатків штучного інтелекту, включаючи обробку природної мови та глибоке навчання.

Java є популярною мовою програмування, завдяки своїм зручним можливостям використання. Деякі з причин, чому розробники продовжують обирати Java більш, ніж інші мови програмування, включають:

– висока якість освітніх ресурсів: Java існує протягом тривалого часу, що забезпечує наявність багатьох різноманітних навчальних матеріалів для початківців. Доступна детальна документація, вичерпні книги та навчальні курси, які допомагають розробникам в процесі навчання. Початківці можуть почати з основ Java (Java Core) перед переходом до більш складних аспектів (Advanced Java);

– багатий набір вбудованих функцій та бібліотек: при використанні Java розробникам не потрібно писати кожну функцію з нуля. Java надає велику

екосистему вбудованих функцій і бібліотек, які полегшують розробку різноманітних програм;

– активна підтримка спільноти: Java має велику активну спільноту користувачів, яка може надати підтримку розробникам при зустрічі з програмними проблемами. Платформа Java також підтримується і регулярно оновлюється;

– інструменти для розвитку якості: Java надає різноманітні інструменти для автоматизованого редагування, налагодження, тестування, розгортання та керування змінами. Ці інструменти роблять програмування на Java ефективним та економічним;

– багатоплатформність: код, написаний на Java, може працювати на різних базових платформах, таких як Windows, Linux, iOS або Android, без необхідності в переписуванні. Це надає значну перевагу у сучасному світі, коли необхідно запускати програми на різних пристроях;

– безпека: Java дозволяє завантажувати ненадійний код через мережу та запускати його в контрольованому безпечному середовищі, де він не може завдати шкоди хост-системі. Ненадійний код не може заражати хост-систему вірусами або читати/записувати файли на жорсткому диску. Рівні та обмеження безпеки в Java також можна легко налаштувати [4].

Мова програмування JavaScript

JavaScript - це мова сценаріїв, що дозволяє реалізувати складну функціональність на веб-сторінках. Вона дозволяє веб-сторінкам бути більш живими та динамічними, ніж просто відображати статичну інформацію. За допомогою JavaScript можна створювати оновлення контенту, інтерактивні карти, анімовану 2D/3D графіку, музичні плеєри з відео тощо [5].

JavaScript був введений у 1995 році як засіб для вбудовування програм у веб-сторінки в браузері Netscape Navigator. З того часу інші популярні графічні веб-браузери також прийняли цю мову. Це сприяло появі сучасних веб-програм, з якими можна взаємодіяти безпосередньо, без необхідності перезавантажувати сторінку після кожної дії. Крім того, JavaScript використовується на традиційних

веб-сайтах для надання різноманітної інтерактивності та інтелектуальних можливостей.

Оскільки мова JavaScript постійно розвивається, браузери мають постійно оновлюватись, щоб підтримувати нові функції. Старі версії браузерів, можуть не підтримувати всі сучасні можливості. Розробники мови стараються уникати внесення змін, які можуть порушити роботу існуючих програм, тому нові браузери здатні запускати старі програми.

JavaScript використовується не лише у веб-браузерах. Деякі бази даних, такі як MongoDB і CouchDB, використовують JavaScript як мову сценаріїв і запитів. Кілька фреймворків програмування для робочих столів і серверів, зокрема Node.js, надають середовище для виконання JavaScript поза межами браузера [6].

Декларативна мова програмування SQL

Структурована мова запитів (SQL) є стандартизованою мовою програмування, яка використовується для управління реляційними базами даних та виконання різноманітних операцій над даними в них. SQL була створена в 1970-х роках і зараз використовується не лише адміністраторами баз даних, але й розробниками, які пишуть сценарії інтеграції даних, та аналітиками даних, які проводять аналітичні запити.

SQL використовується для наступних цілей:

– зміна структури таблиць та індексів бази даних;

– додавання, оновлення та видалення рядків даних;

– отримання підмножини інформації з систем керування реляційними базами даних (РСУБД), яку можна використовувати для обробки транзакцій, аналітичних програм та інших програм, які взаємодіють з реляційною базою даних.

Запити SQL та інші операції складаються з команд, які записуються у вигляді операторів та об'єднуються в програми, що дозволяють користувачам додавати, змінювати або отримувати дані з таблиць бази даних.

Таблиця є основною одиницею бази даних і складається з рядків і стовпців даних. Кожна таблиця містить записи, які зберігаються в рядках таблиці. Таблиці є найпоширенішим типом об'єктів або структур бази даних, що містять або посилаються на дані в реляційній базі даних. Інші типи об'єктів бази даних включають представлення (логічне представлення даних з однієї або кількох таблиць), індекси (прискорюють функції пошуку) та звіти (дані, отримані з однієї або кількох таблиць, як правило, вибрані підмножини даних на основі критеріїв пошуку).

Кожен стовпець у таблиці відповідає категорії даних, наприклад, ім'я або адреса клієнта, і кожен рядок містить значення даних для відповідного стовпця.

Реляційні бази даних названі так через те, що вони складаються з таблиць, які пов'язані між собою. Наприклад, SQL-база даних, що використовується для обслуговування клієнтів, може містити таблицю з іменами та адресами клієнтів, а також інші таблиці з інформацією про конкретні покупки, коди продуктів та контакти клієнтів. Таблиця, яка використовується для відстеження контактів клієнтів, часто містить унікальний ідентифікатор клієнта, відомий як ключ або первинний ключ, для посилання на запис клієнта в окремій таблиці, що містить дані про клієнта, такі як ім'я та контактна інформація.

SQL стала фактично стандартною мовою програмування для реляційних баз даних після їхньої появи наприкінці 1970-х і на початку 1980-х років [7].

2.3.2. Огляд використаних технологій

React

У веб-орієнтованому додатку, який розробляється у межах цієї кваліфікаційної роботи, для розробки інтерфейсу користувача було використано бібліотеку React. React - це бібліотека, створена Facebook, яка базується на компонентах. Вона дозволяє створювати різноманітні інтерфейси користувача та управляти станами компонентів.

Завдяки кільком компонентам і інструментам можна розробляти складні програми, не втрачаючи стану в DOM. Крім того, React можна використовувати для розробки мобільних додатків з використанням React Native - відкритої «native» бібліотеки для мобільних додатків.

React це не зовсім «фреймворк». Це не повне рішення, і часто знадобиться використовувати багато інших бібліотек з React, щоб створити будь-яке рішення.

Під час роботи з фреймворком для розробника вже прийнято багато розумних дизайнерських рішень, що дає вам чіткий шлях, щоб зосередитися на написанні хорошої логіки на рівні програми. Однак фреймворки мають деякі недоліки. Для досвідчених розробників, які працюють з великими кодовими базами, ці недоліки іноді можуть перешкоджати роботі [8].

React дозволяє створювати різноманітні інтерфейси користувача та управляти станами компонентів. React використовується лише для конкретної частини роботи. Він не містить всіх інструментів, які зазвичай знаходяться в традиційних фреймворках JavaScript. Багато рішень стосовно вибору інструментів з екосистеми залишаються на розсуд розробника. Крім того, постійно з'являються нові інструменти, а старі виходять з вжитку. React часто залучає дуже велику кількість різних бібліотек [9].

Згідно зі статистикою «Найбільш використовувані веб-фреймворки серед розробників у всьому світі», яка була приведена за 2022 рік [10], React посів друге місце, як можна побачити на рис. 2.7.

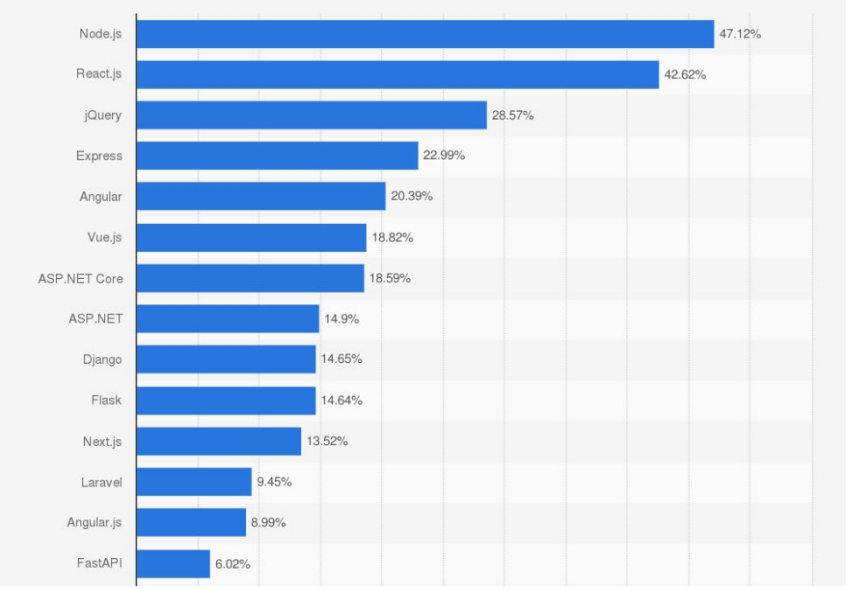

Рис. 2.1. Статистика популярності React

Ця технологія має багато переваг, що роблять її однією з найпопулярніших в світі веб-розробки. Ось деякі з них:

– велика спільнота: у комерційній розробці це один з найважливіших факторів, коли ми вирішуємо використовувати наші бібліотеки, інструменти та інше. Якщо певна технологія має велику кількість прихильників, це полегшує і прискорює процес розробки. Якщо розробник зіткнеться з будь-якою помилкою, він може запитати на публічному форумі, і є велика ймовірність, що серед мільйонів інших розробників React, деякі з них вже розв'язували цю помилку раніше. Завдяки великій спільноті React також має безліч сторонніх бібліотек та посібників;

– зворотна сумісність: однією з найбільших переваг React є те, що він не вносить критичних змін. Після кожного оновлення API React майже не змінюється. Навіть якщо планується ввести значні зміни, буде висвітлено попередження протягом тривалого часу, що дозволить легко перенести ваш код;

– багаторазові компоненти: розробники React можуть створювати багаторазові компоненти. Це дозволяє створювати невеликі фрагменти інтерфейсу користувача та включати їх у будь-якому місці програми. Ви можете поєднувати багато менших компонентів, щоб створити більший складний

інтерфейс користувача. Досить часто розробники створюють додаток, використовуючи, наприклад, готові компоненти інтерфейсу користувача, такі як MaterialUI;

- повні структури SSR/SSG:
  - SSR – «рендеринг» на стороні сервера;
  - SSG - Генератор на стороні сервера;

У світі програмування багато технологій побудовано на React. Серед усіх цих речей є фреймворки, які дозволяють створювати серверні додатки React або просто генерувати файли HTML на основі коду React. Прикладом SSG є Gatsby, який полегшує створення швидких і дружніх до SEO веб-сайтів на React. Іншим корисним фреймворком є Next.js. З ним розробники отримують переваги SSG і можливість створювати сторінки, що «рендеряться» на стороні сервера з використанням React, що полегшує створення динамічних веб-додатків, зручних для SEO;

- віртуальний DOM: Virtual DOM (Document Object Model) відповідає за підтримку абстракції React і оновлення цієї абстракції в реальному DOM (те, що насправді відображається на екрані користувача). Це покращує продуктивність і швидкість програми, оскільки реалізація віртуального DOM React порівнює абстракцію програми між змінами та повторно відображає лише змінені частини програми [11].

Загалом, переваги React полягають у його ефективності, продуктивності, надійності та гнучкості.

Spring Boot

Бізнес-логіка веб-орієнтованого додатку з цієї кваліфікаційної роботи розроблена за допомогою Spring Boot. Spring є відомою платформою з відкритим кодом для розробки корпоративних програм. Вона була створена для спрощення процесу розробки додатків на стеку технологій Java EE від Oracle, який на той момент був важким у використанні.

Цей фреймворк має надійну модель програмування та налаштування. На відміно від багатьох інших фреймворків, він фокусується на кількох головних

аспектах функціональності додатків і надає широкий спектр додаткових можливостей для них.

Однією з основних переваг Spring Framework є використання шаблону Dependency Injection (DI). DI значно спрощує реалізацію функціональних можливостей, необхідних додаткам, і дозволяє розробляти слабко зв'язані, більш загальні класи.

Spring Boot же є фреймворком з відкритим вихідним кодом, призначеним для полегшення розробки з використанням Spring. Він автоматизує процес налаштування та прискорює створення та розгортання додатків Spring.

Хоча основна структура Spring спрямована на гнучкість, Spring Boot зосереджується на скороченні коду та спрощенні розробки веб-додатків. З використанням анотацій та шаблонної конфігурації, цей інструмент допомагає зменшити час, необхідний для розробки програм. Він дозволяє створювати автономні додатки з докладанням менших зусиль для налаштування.

Переваги Spring Boot

Згідно з Stackshare [12], три найпопулярніші особливості Spring Boot визначаються як наступні:

– потужний та зручний;

– простота налаштування;

– підтримка мови програмування Java.

У загальному, Spring Boot сприяє спрощенню роботи зі Spring для розробників. Як саме це досягається:

– легкий розвиток - усі функції фреймворку спрямовані на прискорення та спрощення розробки програм;

– відсутність необхідності у розгортанні файлів .war - розробник може упакувати свою програму як виконуваний файл .jar, не потребуючи розгортання .war файлів;

– автономні програми - розробник може запакувати свої сервіси як окремі автономні програми;

– вбудовування Tomcat, Jetty або Undertow - розробник може вбудувати сервери Tomcat, Jetty або Undertow безпосередньо у програму;

– відсутність конфігурації XML - автоматична конфігурація усуває необхідність вручну налаштовувати XML-файли;

– зменшення обсягу вихідного коду - фреймворк допомагає зменшити кількість коду, який потрібно писати для розробки Spring-програм;

– додаткові функціональні можливості - фреймворк містить корисні функції для тестування, підтримки Maven та Gradle, вбудованого веб-сервера для кожної програми та інше;

– легкий старт - це особливо зручно для початківців, оскільки він автоматично налаштовує багато речей, які їм потрібні;

– просте налаштування та управління - автоматичне налаштування та просте керування базами даних та залежностями дозволяють швидко налаштувати програму;

– спільнота - велика спільнота користувачів та наявність багатьох посібників полегшують процес ознайомлення з фреймворком.

Загалом, фреймворк Spring Boot робить неймовірну роботу у спрощенні розробки з використанням Spring. Він допомагає керувати залежностями та дозволяє користувачам запускати програми безпосередньо з командного рядка. Він також не вимагає зовнішнього контейнера програм та суттєво зменшує кількість коду, необхідного для налаштування [13].

MySQL

На початку, у 1994 році, шведська компанія під назвою MySQL AB була відповідальна за розробку MySQL. Пізніше, в 2008 році, американська технологічна компанія Sun Microsystems придбала MySQL AB та стала її повним власником. У 2010 році американський технологічний гігант Oracle придбав Sun Microsystems, і з того часу MySQL фактично перейшла під контроль компанії Oracle.

Загалом, MySQL є РСУБД з відкритим вихідним кодом та моделлю клієнт-сервер. РСУБД - це програмне забезпечення або сервіс, який використовується для створення та управління базами даних, заснованих на реляційній моделі.

MySQL, хоча і не єдине РСУБД на ринку, але вважається одним з найпопулярніших, поступаючись лише Oracle Database за такими ключовими показниками, як кількість згадок у пошукових результатах, професійних профілях на LinkedIn та частотою технічних дискусій на Інтернет-форумах. Існує кілька причин, чому багато великих технологічних компаній віддають перевагу MySQL, що ще більше підтверджує його місце у сучасному світі:

– гнучкість і простота використання: MySQL надає можливість змінювати вихідний код відповідно до потреб розробника, і йому не потрібно платити за цю можливість, включаючи оновлення до покращеної комерційної версії. Процес встановлення відносно простий і займає не більше 30 хвилин;

– висока продуктивність: MySQL підтримує широкий спектр кластерних серверів. Незалежно від того, чи зберігає розробник великі обсяги даних електронної комерції, чи проводить серйозний бізнес-аналіз, MySQL може надати оптимальну швидкість роботи;

– стандарт у галузі: MySQL використовується в індустрії протягом багатьох років, що означає наявність великої кількості ресурсів для кваліфікованих розробників. Користувачі MySQL можуть розраховувати на швидку розробку програмного забезпечення та доступ до експертів, готових працювати за більш прийнятну ціну, якщо знадобиться додаткова підтримка;

– безпека: при виборі правильного програмного забезпечення РСУБД безпека даних користувача є найважливішою. MySQL встановлює високі стандарти безпеки завдяки системі керування привілеями доступу та обліковими записами користувачів. Вона надає можливості автентифікації на основі хоста і шифрування паролів.

Отже, MySQL визнана і поширена як надійна СУБД, яка має простоту використання, високу продуктивність, широку підтримку галузевих стандартів та високий рівень безпеки [14].

TensorFlow

У межах цієї кваліфікаційної роботи був використаний TensorFlow, як засіб для створення сітки обличчя (facemesh) з анотаціями. Завдяки цьому функціоналу реалізоване відстеження обличчя та подальше накладання зображення окулярів на нього.

TensorFlow - це відкрита платформа для машинного навчання, яка має гнучку екосистему інструментів, бібліотек і ресурсів спільноти. Вона дозволяє дослідникам розвивати новітні технології машинного навчання, а розробникам - легко створювати та розгортати програми машинного навчання. TensorFlow був розроблений дослідниками та інженерами, які працюють у команді Google Brain в організації Google Machine Intelligence Research для машинного навчання та дослідження глибоких нейронних мереж. Вона має стабільні API Python і C++, а також API негарантованої зворотної сумісності для інших мов. TensorFlow поєднує різні моделі та алгоритми машинного навчання і глибокого навчання і робить їх придатними [15].

TensorFlow пропонує кілька рівнів абстракції для легкого створення моделі, тому розробник може обрати підходящий рівень для своїх потреб. TensorFlow надає API високого рівня Keras для створення та навчання моделей, що полегшує початок роботи з TensorFlow і машинним навчанням. TensorFlow завжди забезпечує прямий шлях до Robust ML, незалежно від того, яку мову чи платформу використовує розробник. TensorFlow забезпечує гнучкість і контроль завдяки таким функціям, як Keras Functional API та Model Subclassing API для створення складних топологій [16].

Axios API

У веб-орієнтованому додатку був використаний Axios API для спрощення виконання запитів на сервер та отримання до них відповідей. Axios API - це HTTP-клієнт, заснований на Promise для Node.js та браузера. Цей API було обрано, так як він значно спрощую виконання запитів з браузеру та робить код більш зрозумілішим для розробника.

Його особливості:

– виконує XMLHttpRequests із браузера;
– виконує http-запити з node.js;
– підтримує Promise API;
– перехоплює запит і відповідь;
– скасування запиту та інше.

Загалом Axios є корисним інструментом для створення HTTP-запитів від програм JavaScript, що працюють на сервері, а також у веб-браузерах. Він надає простий API для створення HTTP-запитів і підтримує Promise API, що полегшує роботу з асинхронним кодом [17].

Bootstrap

Дизайн же сайту веб-орієнтованого додатку, який розробляється у межах цієї кваліфікаційної роботи, виконаний з використанням CSS та допоміжних інструментів таких як Bootstrap. Bootstrap - це безкоштовна платформа веб-розробки з відкритим кодом, яка була створена для спрощення процесу розробки адаптивних веб-сайтів, спрямованих на мобільні пристрої. Вона надає набір синтаксису для дизайну шаблонів.

Основною метою Bootstrap є створення адаптивних веб-сайтів, що добре працюють на будь-якому розмірі екрану. Він забезпечує оптимальну роботу всіх елементів інтерфейсу сайту незалежно від пристрою.

Bootstrap доступний у двох версіях - з попередньо скомпільованим кодом і з вихідним кодом. Досвідчені розробники віддають перевагу останній, оскільки вона дозволяє налаштовувати стилі відповідно до власних проектів.

Простота використання - це одна з переваг Bootstrap. Завдяки його популярності існує багато навчальних матеріалів і онлайн-форумів, які допоможуть розробнику почати роботу. Однією з причин, чому Bootstrap є таким популярним серед веб-розробників і веб-дизайнерів, є його проста файлова структура. Файли легко доступні для редагування з базовими знаннями HTML, CSS і JS.

«Responsive Grid» - Bootstrap постачається з визначеною наперед системою сітки, що звільняє розробника від її створення з нуля. Вона складається з рядків

і стовпців, що дозволяє створювати сітку всередині існуючої сітки, замість використання медіа-запитів у CSS-файлі.

Крім того, «responsive grid» Bootstrap спрощує введення даних, оскільки містить багато медіа-запитів, які дозволяють визначати спеціальні контрольні точки для кожного стовпця залежно від потреб веб-проекту.

Зазвичай вистачає стандартних налаштувань. Після створення сітки все, що потрібно зробити, це додати вміст до контейнерів. Грід-система Bootstrap має два класи контейнерів для кращої адаптації настільних і мобільних проектів - фіксований контейнер (.container) і .container-fluid. Перший клас контейнерів надає фіксовану ширину контейнера, тоді як другий пропонує повну ширину контейнера, здатного адаптуватися до розмірів екрана[18].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Структура і алгоритми системи

Структура веб-орієнтованого додатку

Система веб-орієнтованого додатку інтернет магазину оптики складається з таких елементів:

– Головна сторінка з інформацією про поточні пропозиції та магазин. Компоненти:
  – Home.js;
  – Banner.js;
– Сторінка каталогу з товарами. Компоненти:
  – Catalog.js;
  – Products.js;
  – Filters.js;
  – Pagination.js;
  – BasicDropdown.js;
– Інформаційна сторінка товару. Компоненти:

- Product.js;
- AddToCart.js;
- TryProduct.js
- TryProductFacemesh.js

– Сторінка кошику. Компоненти:

- Cart.js;
- CartItem.js;
- CartAmountToggle.js;

– Сторінка реєстрації та авторизації користувача. Компоненти:

- LoginForm.js;
- Registration.js;

– Сторінка кабінету користувача з персональними даними та замовленнями. Компоненти:

- PersonalPage.js;
- PersonalData.js;
- EditPersonalData.js
- OrderData.js;
- OrderDataItem.js

– Сторінка кабінету адміністратора. Компоненти:

- AdminPersonalPage.js
- AdminProducts.js
- AdminOrders.js
- AdminOrderDataItem.js
- AdminDropzone.js

– На всіх сторінках наявні «хедер» та «футер». Компоненти:

- Header.js;
- Footer.js.

– Інше:

- PageNotFound.js
- Error.js

- CartReducer.js
- usePagination.js
- FacemeshUtils.js
- CartContext.js
- Pagination.js
- Layout.js
- Loading.js

За отримання запитів, їх обробку та відправку відповідей відповідають такі класи Rest-контролери:

- UserController
- EyeglassController
- EyglassFilterController
- OrderController
- CabinetController

Алгоритм роботи веб-орієнтованого додатку

Сайт веб-орієнтованого додатку інтернет магазину оптики є динамічним, реалізованим за допомогою React, JavaScript та JQuery. Особливість React в тому, що компоненти, змінюючи стан, виконують рендер окремо один від одного, що зменшує навантаження на мережу та прискорює швидкодію додатку.

Маршрутизація сайту налаштована за допомогою компоненту BrowserRouter, який складається з адресів всіх сторінок:

```
<Route path="/catalog/male/:currentPage?:limit?:shape?" element={<Catalog />}></Route>
<Route path="/catalog/female/:currentPage?:limit?:shape?" element={<Catalog />}></Route>
<Route path="/catalog/kids/:currentPage?:limit?:shape?" element={<Catalog />}></Route>
<Route path="/login" element={<LoginForm />}></Route>
<Route path="/reg" element={<RegistrationForm />}></Route>
<Route path="/product/:id?" element={<Product />}></Route>
<Route path="/cart" element={<Cart />}></Route>
<Route exact path="/cabinet/personal" element={<PersonalPage showPersonal/>}></Route>
```

<Route exact path="/cabinet/orders" element={<PersonalPage showOrders/>}></Route>

<Route path="/tryproduct" element={<TryProduct />}></Route>

Веб-орієнтований додаток інтернет магазину оптики побудований за шаблоном MVC (Model-View-Controller). MVC дозволяє відокремити логіку додатку від його візуального представлення, що спрощує розробку, тестування та підтримку програмного забезпечення. Крім того, він підтримує принцип одноразового використання коду та забезпечує модульність та повторне використання компонентів додатку.

Тобто модель додатку розділена на три основні компоненти:

– модель (Model): Складається з сутностей бази даних;

– представлення (View): Відповідає за візуалізацію даних та взаємодію з користувачем. Цю роль займає додаток розроблений на React;

– контролер (Controller): Відповідає за обробку вхідних даних від користувача та взаємодію з моделлю та представленням. Задля реалізацію контролеру було використано API, яке розроблене на Spring Boot.

На рис. 2.2 наведено архітектуру додатку.

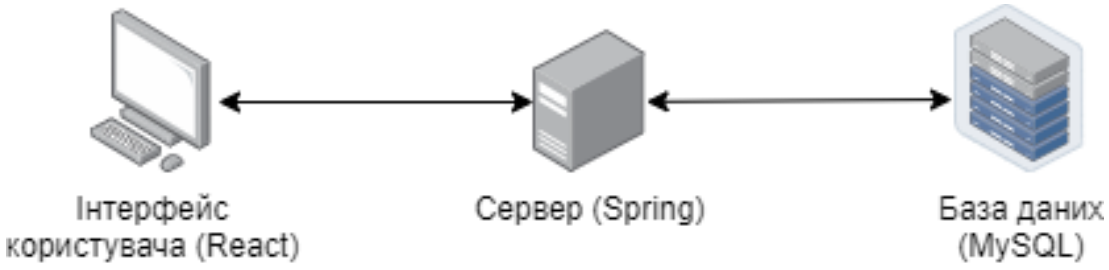

Рис. 2.2. Трирівнева архітектура

Для розробки інтерфейсу користувача було використано бібліотеку React, яка надає швидке та зручне спілкування користувача з додатком магазину.

Алгоритм функціювання обміну даних між різними шарами додатку застосовує REST, архітектурний стиль для побудови web-додатків, які використовують протокол HTTP для передачі даних між клієнтом та сервером. REST дозволяє взаємодіяти з сервером через стандартні HTTP-методи, такі як GET, POST, PUT та DELETE. Послідовність виконання алгоритму:

Використовуючи Axios API, з інтерфейсу користувача відправляється запит на сервер Spring, де виконується його подальша обробка. Приведемо приклад таких запитів:

GET-запити:

– отримання списку товарі з використанням пагінації;

– отримання товару за його ідентифікатором та кольором;

– сортування товару за його ознаками.

POST-запити:

– реєстрація нового користувача;

– вхід до системи вже існуючого користувача;

– створення замовлення.

Метод обробки запиту задіє Spring Boot JPA (рис. 2.3) – це специфікація Java для керування реляційними даними в програмах Java. Вона дозволяє отримати доступ до даних і зберігати їх між об’єктом/класом Java і реляційною базою даних.

```
1 usage
public interface EyeglassRepository extends JpaRepository<EyeglassView, Integer> {
    1 usage
    @Query("SELECT e FROM EyeglassView e WHERE e.id = :id AND e.color != :color")
    List<EyeglassView> findAltColorsById(@Param("id") int id, @Param("color") String color);

    1 usage
    EyeglassView findByIdAndColor(int id, String color);

    List<EyeglassView> findAll();

    @Query("SELECT e FROM EyeglassView e WHERE e.position = 1")
    List<EyeglassView> findAllMain();
```

Рис. 2.3. JPA репозиторій

Отриманий об’єкт з бази даних MySQL чи повідомлення повертається у відповіді до запиту на інтерфейс користувача. React використовую JSX (JavaScript XML) – це розширення React, яке дозволяє писати код JavaScript, який виглядає як HTML, що дає можливість з легкістю використовувати отриманий об’єкт разом з HTML розміткою.

Як наочний приклад обрано сторінку каталогу, яка показана на рис. 2.4.

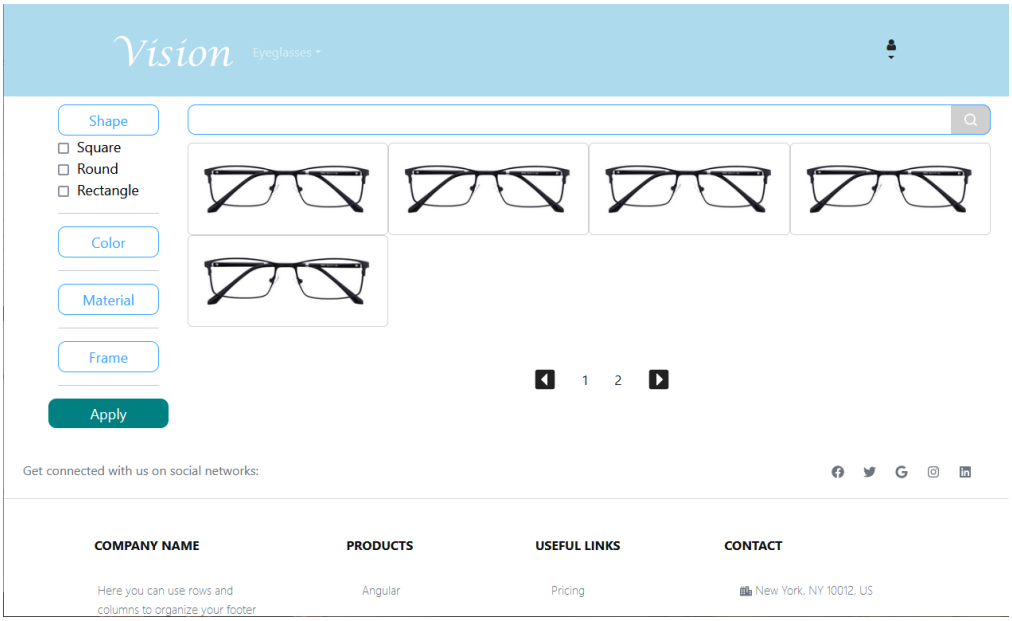

Рис. 2.4. Сторінка каталогу

Як було зазначено на початку розділу ця сторінка складається з компонентів списку товарів (Products.js) та списку фільтрів (Filters.js). Ці компоненті поєднані у компоненті Catalog.js, який повертає форму з дією (action), яка відправляє запит по адресу /catalog. При першому заході на сторінку каталогу користувач переходить за адресою /catalog. Компонент Filters.js (рис. 2.5) повертає список фільтрів відправляючи GET-запит за допомогою AxiosAPI.

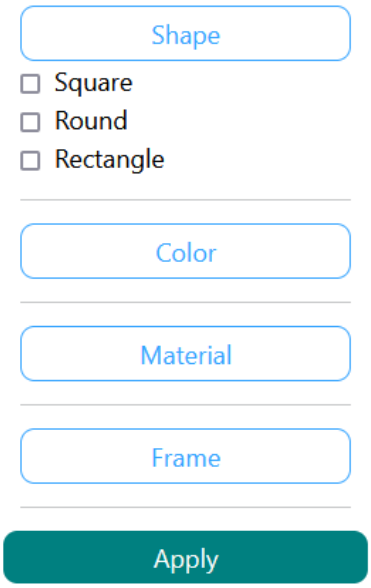

Рис. 2.5. Компонент фільтрів

Приклад запиту:

```
const getShapes = async () => {
    try {
      const response = await api.get(
        "http://localhost:8080/allShapes"
      );
      setShapes(response.data);
    } catch (err) {
      setShapes(null);
    } finally {

    }
};
```

Компонент Products.js (рис. 2.6) повертає список товарів, отримуючи поточну адресу каталогу з параметрами /catalog?currentPage=1&limit=5. Натискаючи кнопки «Застосувати фільтри» чи «Знайти» форма оброблюється. Для застосування фільтрів додаються ще параметри:

– shape – форма окулярів;
– color – колір окулярів;
– material – матеріал окулярів;
– frame – оправа окулярів;
– name – назва окулярів.

Рис. 2.6. Компонент товарів

Приклад запиту:

```
const getData = async () => {
        try {
          const response = await api.get(
            getCurrentURL() – функція застосування параметрів фільтрів до адреси
              );
          setData(response.data);
        } catch (err) {
          setData(null);
        } finally {
          setLoading(false);
        }
      };
      getData();
```

Запит /catalog?currentPage=1&limit=5 з фільтрами чи без них відправляється на сервер Spring, де оброблюється відповідним методом-ендпоінтом. Приклад методу для категорії чоловічих окулярів:

```
@CrossOrigin(origins = "http://localhost:8080")
  @GetMapping(path = "/catalog/male")
  public @ResponseBody Page<List<EyeglassView>> getEyeglassesMale(@RequestParam
Map<String, String> allParams) {
        if (allParams.containsKey("shape") || allParams.containsKey("color") ||
allParams.containsKey("material")
            || allParams.containsKey("frame") || allParams.containsKey("name")) {
          Pageable firstPageWithTwoElements =
PageRequest.of(Integer.parseInt(allParams.get("currentPage")),
Integer.parseInt(allParams.get("limit")));
          return
eyeglassRepository.findByCategoryByShapeAndColorAndMaterialAndFrameAndName(firstPage
WithTwoElements, EyeglassCategory.MALE,
              allParams.get("shape"),
              allParams.get("color"),
              allParams.get("material"),
```

```
                allParams.get("frame"),
                allParams.get("name"));
        }
        Pageable firstPageWithTwoElements = PageRequest.of(Integer.parseInt(allParams.get("currentPage")),
Integer.parseInt(allParams.get("limit")));
        return eyeglassRepository.findAllByCategoryMain(firstPageWithTwoElements, EyeglassCategory.MALE);
    }
```

Метод шукає відповідні товари за отриманими параметрами, використовуючи репозиторій та повертає список (List) товарів з бази даних. Команда, яка виконується:

```
@Query("SELECT e FROM EyeglassView e WHERE (:category is null or e.category = :category)" +
        "and (:shape is null or e.shape = :shape)" +
        "and (:color is null or e.color = :color)" +
        "and (:material is null or e.material = :material)" +
        "and (:frame is null or e.frame = :frame)" +
        "and (:name is null or e.name LIKE %:name%)")
```

Отриманий список повертається у відповідь на запит до інтерфейсу користувача, де за допомогою JSX виводиться на сторінку:

```
{data.content.map((eyeglass, index) => (
    ...
        <img src={require("../../images/" + eyeglass.img)} alt={eyeglass.name} />
        <div className="txtcontent">
        <div className="simpletxt">
        <h3 className="name">{eyeglass.name}</h3>
        <p>{eyeglass.description}</p>
        <h4 className="price">{eyeglass.price}&euro;</h4>
        <Link to={"/product?id=" + eyeglass.id + "&color=" + eyeglass.color} className="text-left text-reset">READ MORE</Link><br />
        <div className="wishtxt">
        <p className="paragraph1"> Add to Wishlist <span className="glyphicon glyphicon-heart"></span> </p>
```

```
                <p className="paragraph2">Compare <span className="icon"><img
src="image/compicon.png" alt="compicon" /></span></p>
                </div>
                </div>
            ...
        ))}
```

Алгоритм накладання зображення окулярів на обличчя користувача

Користувач потрапляє на сторінку, де погоджується дати доступ до камери та натискає кнопку «Почати», тоді запускається таймер з інтервалом в 1000 мс і TensorFlow, бібліотека для машинного навчання, повертає масив орієнтирів обличчя з камери. Для створення масиву орієнтирів використано пакет Face Landmarks Detection з TensorFlow, який надає моделі для запуску розпізнавання обличчя та відстеження його орієнтирів у реальному часі. Цей пакет дає змогу отримати передбачені 486 3D орієнтирів обличчя (ключових точок), щоб визначити приблизну геометрію поверхні людських обличь, ці точки можна побачити на рис. 2.7. Якщо довжина масиву більше 0, то виконується визначення необхідних координат, а саме середина між очима, ліва та права координата контуру обличчя.

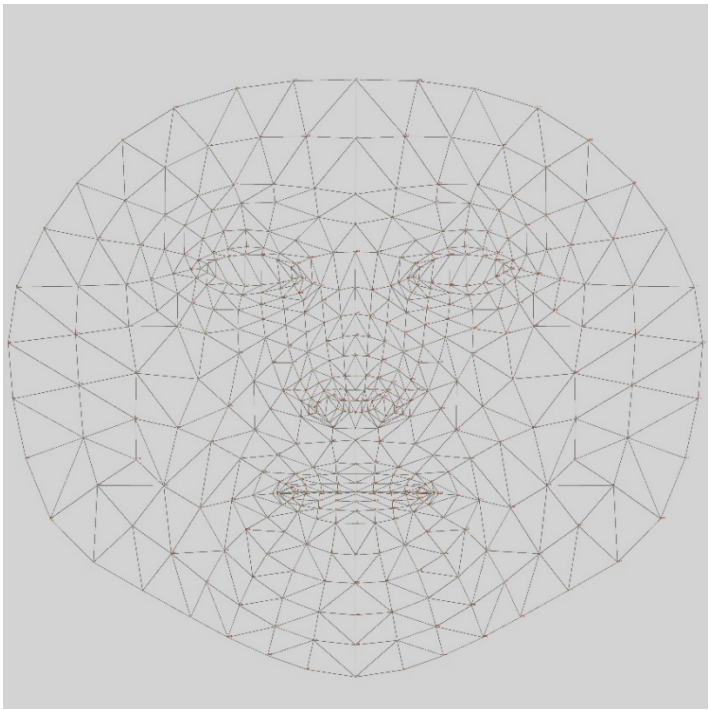

Рис. 2.7. Діаграма ключових точок

Маючи координати точки, яка знаходиться посередині між очима, визначається центр в який буде накладено зображення окулярів. Ліва та права координатні точки, а саме відстань між цими точками, використовуються для визначення ширини зображення окулярів. Функція, яка відповідальна за розрахунок, має заздалегідь визначені початкові параметри висоти та ширини, базуючись на цих параметрах зображення окулярів змінює масштаб в залежності від відстані користувача до камери. Опис математичних формул, використаних для розрахунків, наведено у підрозділі 2.2

Задля наочного прикладу інформацію, яка оброблюється функцією, було залоговано до консолі, що можна побачити на рисунку 2.8

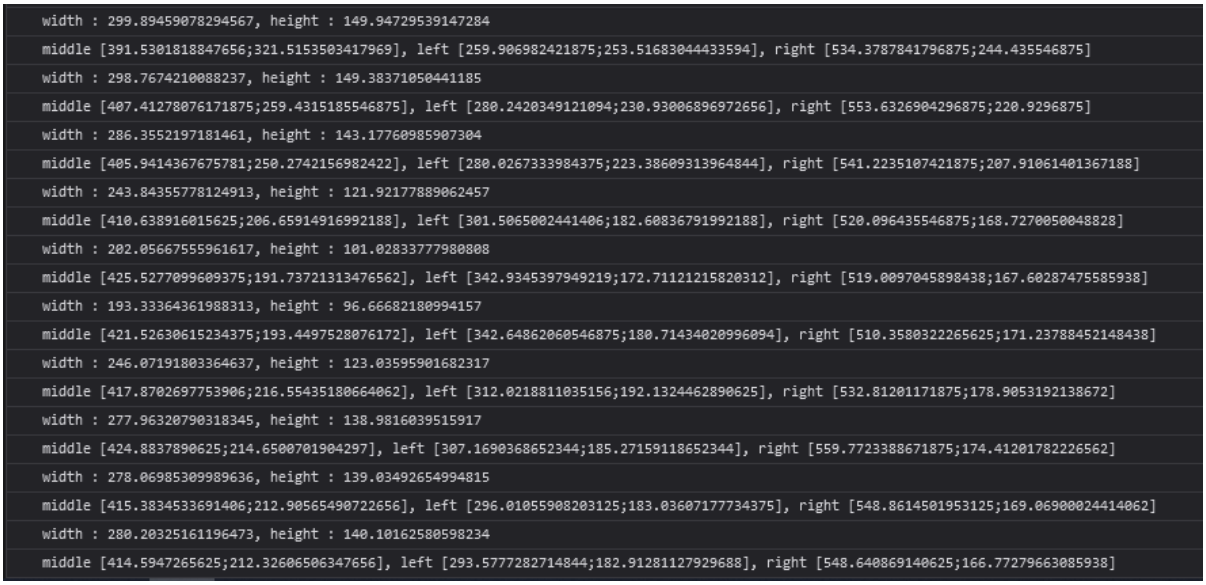

Рис. 2.8. Лог функції обробки накладання

2.4.2. Структура бази даних

База даних, використана у проекті кваліфікаційної роботи, є реляційною. Перед початком проекту було визначено основні сутності та їх відношення, що будуть реалізовані у вигляді таблиць бази даних. Основними сутностями є товари (eyeglass, eyeglass_color), замовлення (user_order, user_details_order, order_item) та користувачі системи (user). Інші таблиці (shape, color, material,

frame та payment_method) є допоміжними і містять менш значну інформацію для підтримки цілісності основних сутностей.

Сутності бази даних та їх атрибути

Таблиці eyeglass, eyeglass_color, eyeglass_image, eyeglass_stock, discount відповідають за зберігання інформації про товари магазину.

eyeglass – зберігає інформацію про товари:

- id – первинний ключ, ідентифікатор товару;
- name – назва товару;
- description – опис товару;
- price – ціна товару;
- shape_id – ідентифікатор форми товару;
- material_id – ідентифікатор матеріалу товару;
- frame_id – ідентифікатор оправи товару;
- category – категорія товару (чоловіча, жіноча чи дитяча).

eyeglass_color – кольори окулярів:

- eyeglass_id – складовий первинний ключ, ідентифікатор товару;
- color_id – складовий первинний ключ, ідентифікатор кольору;
- position – складовий первинний ключ, позиція у каталозі (позиція 1 більш пріоритетна та виводиться у каталозі, інші виводяться у порядку зростання на сторінці продукту);
- tryproduct_img – зображення товару для накладення на обличчя.

eyeglass_image – зображення окулярів:

- id – первинний ключ, ідентифікатор товару;
- eyeglass_color_id – ідентифікатор товару за кольором;
- img – зображення товару;
- img_position – позиція зображення на сторінці продукту.

eyeglass_stock – кількість окулярів на складі:

- eyeglass_id – первинний ключ, ідентифікатор товару;
- stock – кількість на складі;

discount – знижка на товар:

– id – первинний ключ, ідентифікатор знижка;
– eyeglass_id – ідентифікатор товару;
– color_id – ідентифікатор кольору;
– discount_price – ціна зі знижкою;
– date_of_opening – дата створення знижки;
– date_of_closing – дата завершення знижки.

Таблиці user, user_order, user_details_order, order_item відповідають за зберігання даних користувача та його замовлень. user – користувачі:

– id – первинний ключ, ідентифікатор користувача;
– first_name – ім’я користувача;
– last_name – прізвище користувача;
– email – електронна пошта користувача;
– phone_number – номер телефону користувача;
– password – пароль користувача;
– avatar – аватар користувача;
– user_role – роль користувача у системі.

user_order – замовлення користувачів:

– id – первинний ключ, ідентифікатор замовлення;
– date_of_creation – дата створення замовлення;
– date_of_closing – дата завершення замовлення;
– total_price – повна сума для оплати замовлення;
– order_status – статус замовлення;

user_details_order – дані про користувача, який отримує замовлення

– order_id – складовий первинний ключ, ідентифікатор замовлення;
– user_id – складовий первинний ключ, ідентифікатор користувача;
– first_name – ім’я користувача, який отримує замовлення;
– last_name – прізвище користувача, який отримує замовлення;
– phone_number – телефонний номер користувача, який отримує замовлення;
– email – електронна пошта користувача, який отримує замовлення;

– payment_method – спосіб оплати замовлення

order_item – зміст замовлення;

– order_id – складовий первинний ключ, ідентифікатор замовлення;

– eyeglass_id – складовий первинний ключ, ідентифікатор товару;

– color_id – складовий первинний ключ, ідентифікатор кольору товару;

– amount – кількість позиції товару у замовленні;

– price – ціна товару, яку має сплатити користувач;

Таблиці shape, color, material, frame, payment_method відповідають за зберігання наявних характеристик товарів, які знадобляться при фільтрації каталогу:

shape – форма окулярів:

– id – первинний ключ, ідентифікатор форми товару;

– name – найменування форми;

color – колір окулярів:

– id – первинний ключ, ідентифікатор кольору товару;

– name – найменування кольору;

material – матеріал окулярів:

– id – первинний ключ, ідентифікатор матеріалу товару;

– name – найменування матеріалу;

frame – оправа окулярів:

– id – первинний ключ, ідентифікатор оправи товару;

– name – найменування оправи;

vendor – постачальник окулярів:

– id – первинний ключ, ідентифікатор постачальника;

– name – найменування постачальника;

payment_method – метод оплати замовлення:

– id – первинний ключ, ідентифікатор способу оплати замовлення;

– payment_method – найменування способу оплати.

Представлення eyeglass_view, eyeglass_popularity_view, order_view та discount_view використані для більш зрозумілішої передачі даних до клієнтської частини.

На рис. 2.9 наведено ER-діаграму бази даних, яка використовується у системі

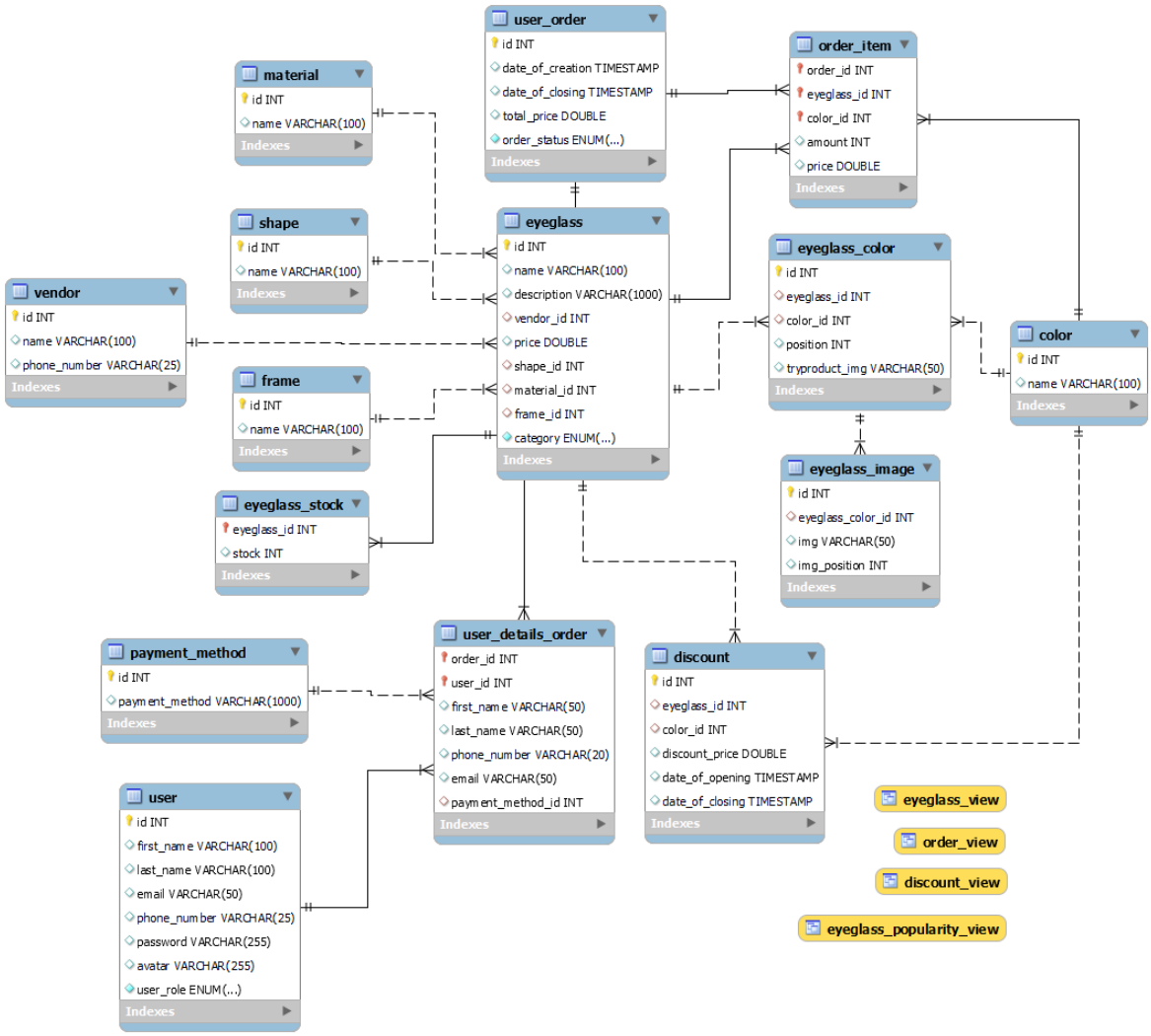

Рис. 2.9. ER-діаграма

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення інтернет магазину отримує вхідні дані шляхом завантаження інформації з бази даних, текстового вводу та локального сховища (якщо потрібно).

Вхідні дані:

– список товарів;
– список фільтрів;
– список замовлень користувача;
– інформація користувача;
– операції адміністратора з товарами.

Вхідні дані користувача організовуються через текстовий ввід інформації користувачем у відповідні поля. Нижче наведено приклад вводу інформації:

Серед вхідних даних від користувача вимагається введення своєї електронної пошти та паролю на спеціальній формі (рис. 2.10) при вході у систему:

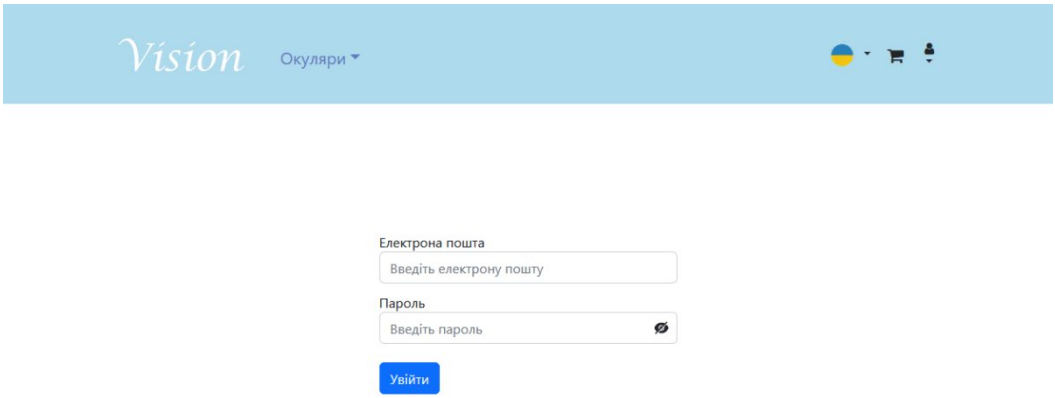

Рис. 2.10. Сторінка входу

При реєстрації нового користувача у систему, додатково до цього, додається поля введення ім’я, прізвища та номеру телефону, як це показано на рис. 2.11.

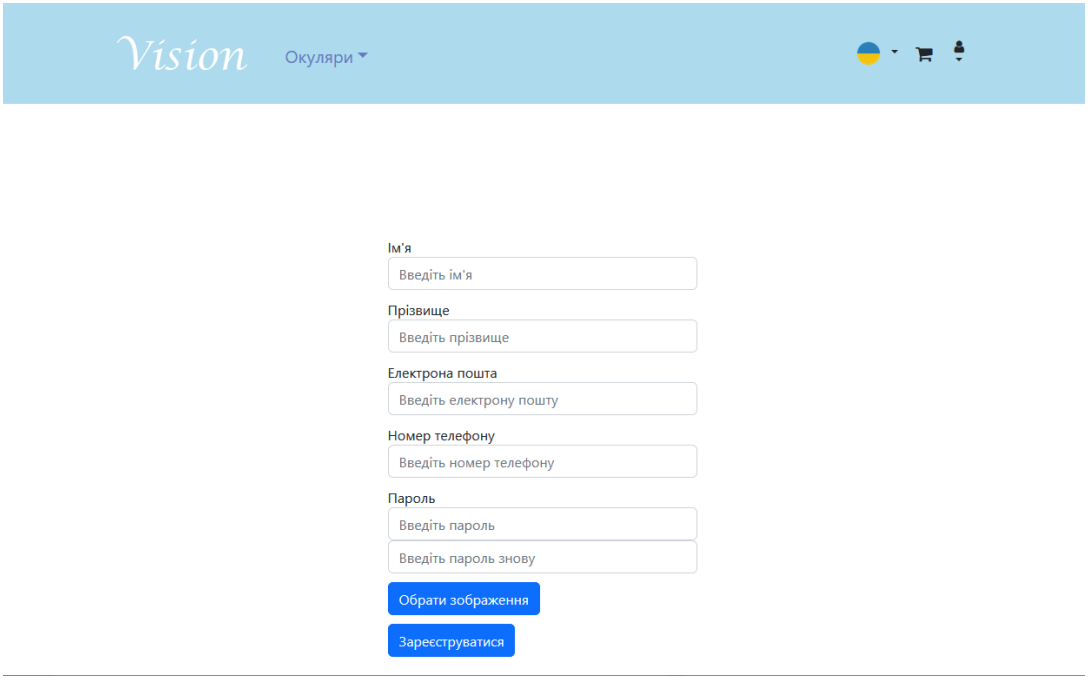

Рис. 2.11. Сторінка реєстрації

Вихідні дані включають в себе:

– сторінка каталогу;

– сторінка продукту;

– сторінка замовлень.

Кошик з товарами знаходиться у локальному сховищі та бере на вхід інформацію про товар та його бажану кількість, що було додано користувачем. На вихід формується запит до серверу Spring, де формується замовлення та вноситься до бази даних

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Веб-орієнтовану інформаційну систему було протестована на персональному ЕОМ з такими характеристиками та периферією:

– процесор: AMD Ryzen 5 2600 3.4GHz 16MB sAM4;

– материнська плата: MSI B450 Tomahawk (sAM4, AMD B450, PCI-Ex16);

– модулі пам’яті: Kingston Fury DDR4-2666 16384 MB PC4-21300 (Kit of 2x8192);

– система охолодження: be quiet! Dark Rock Slim;

– SSD: Samsung 970 Evo series 500GB M.2;

– Блок живлення: be quiet! System Power 9 600W;

– монітор: 27" LG UltraGear;

– Комп’ютерна миша;

– Клавіатура.

2.6.2. Використані програмні засоби

Під час розробки цього застосунку були використані наступні програмні засоби:

– Visual Studio Code;

– IntelliJ IDEA;

– MySQL Workbench 8.0;

– Node.js;

– Figma.

Visual Studio Code

Visual Studio Code (рис. 2.12) – це редактор коду з відкритим вихідним кодом, розроблений компанією Microsoft, який спеціально оптимізований для розробки та налагодження сучасних веб- та хмарних програм. Він представляє собою легкий, але потужний редактор вихідного коду, доступний для операційних систем Windows, macOS і Linux. У ньому реалізовано багато корисних функцій, включаючи підтримку налагодження, підсвічування синтаксису, інтелектуальне автодоповнення, фрагменти коду, можливість рефакторингу коду та вбудований Git. Крім того, Visual Studio Code надає

можливість встановлення розширень, що дозволяють додавати нові мови програмування, теми, налагоджувачі і інтеграцію з додатковими сервісами [19].

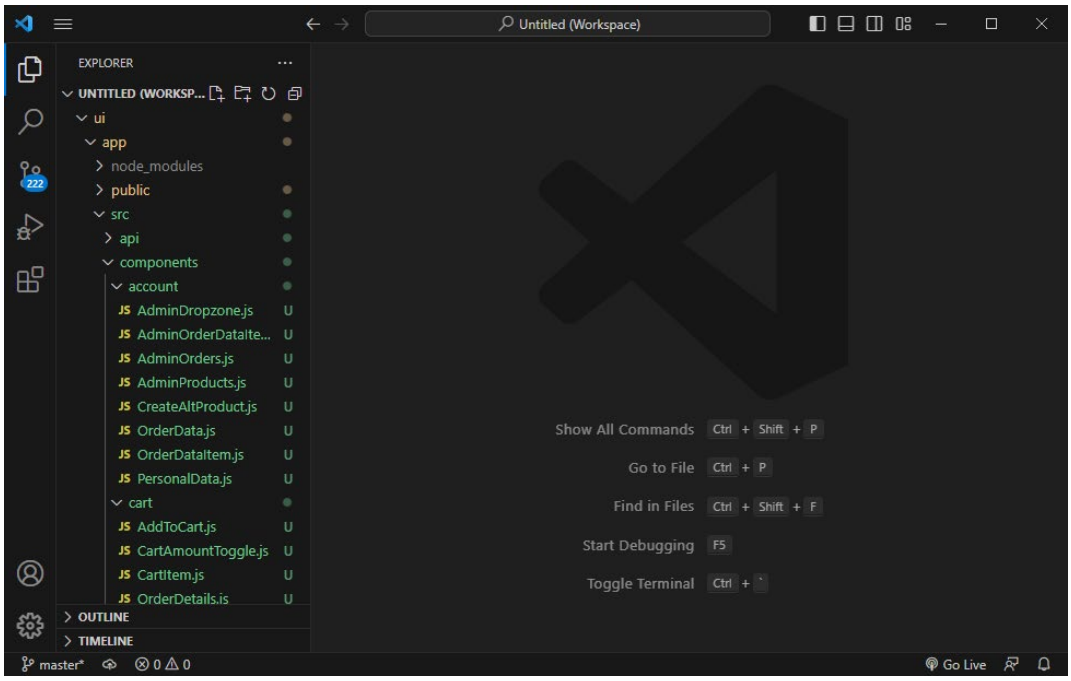

Рис. 2.12. Visual Studio Code

IntelliJ IDEA

IntelliJ IDEA (рис. 2.13) є провідною інтегрованою середовище розробки (IDE) для Java і Kotlin, яке сприяє підвищенню продуктивності розробників за допомогою розширеного набору функцій. IntelliJ IDEA забезпечує розробників набором інструментів, що допомагають підвищити продуктивність, забезпечують надійність рефакторингу, спрощують навігацію по коду та забезпечують інтегровані засоби для розробки, веб-підтримки та корпоративного розвитку.

IntelliJ IDEA відоме своєю високою продуктивністю, інтегрованими інструментами для роботи з фреймворками та технологіями, такими як Spring, Hibernate, Maven, Gradle, Android та багато інших [20].

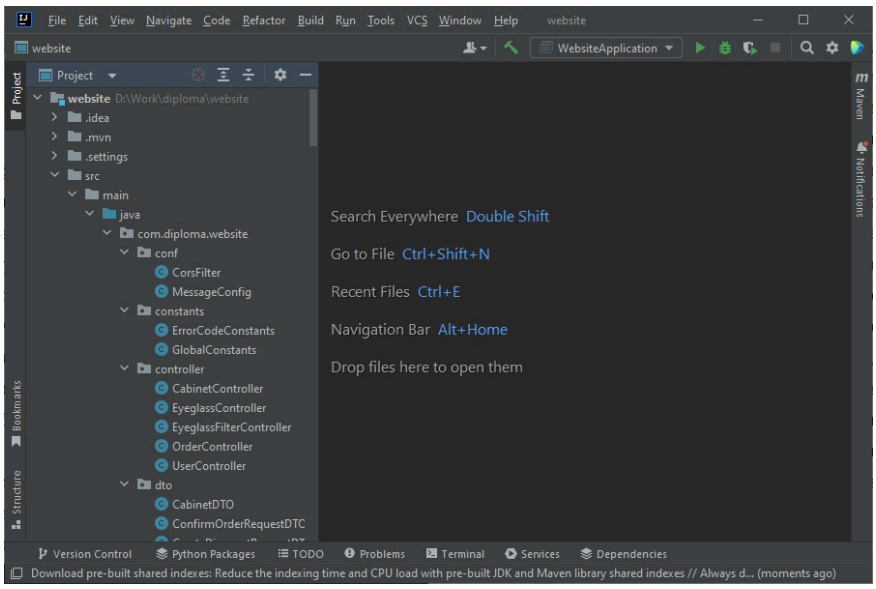

Рис. 2.13. IntelliJ IDEA

MySQL Workbench

MySQL Workbench (рис. 2.14) є візуальним інструментом, призначеним для архітекторів баз даних, розробників та адміністраторів. Цей інструмент надає засоби для моделювання даних, написання SQL-запитів і розширення можливостей для керування базою даних, такі як налаштування сервера, адміністрування користувачів, створення резервних копій і багато іншого. MySQL Workbench доступний для операційних систем Windows, Linux і Mac OS X [21].

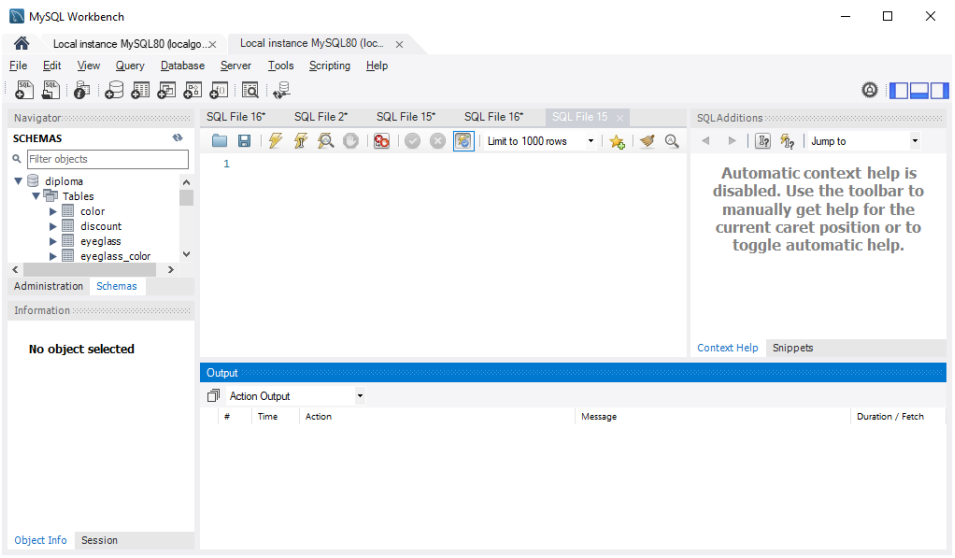

Рис. 2.14. MySQL Workbench

Node.js

Node.js - це бібліотека з відкритим вихідним кодом та міжплатформне середовище виконання JavaScript, яке дозволяє запускати веб-додатки поза веб-браузером. Вона була розроблена Раяном Далем у 2009 році, а остання версія - 15.14 - вийшла у квітні 2021 року. Розробники використовують Node.js для створення серверних веб-додатків, і воно ідеально підходить для додатків, які мають великі обчислювальні навантаження або потребують обробки великого обсягу даних, оскільки воно використовує асинхронну модель, що базується на подіях [22].

Воно приймає важливу роль при створені веб-орієнтованого додатку з цієї кваліфікаційної роботи, так як надає велику кількість пакетів і бібліотек, доступних через менеджер пакетів npm, що дозволяє легко використовувати готові рішення та сторонні бібліотеки для розширення функціональності додатку.

Figma

Figma – це інструмент для спільного проектування інтерфейсу користувача, який працює в браузері і дозволяє користувачам спільно працювати над створенням яскравих та інтерактивних прототипів. З моменту свого випуску в 2016 році Figma стала популярним рішенням як у сфері веб-дизайну, так і серед онлайн-спільнот. Вона дозволяє користувачам співпрацювати, ділитися шаблонами, дизайнами та віджетами з мільйонами користувачів по всьому світу [23].

2.6.3. Виклик та завантаження програми

У процесі розробки допускається використання ПК, для комерційного застосування знадобиться серверне обладнання, як можна придбати чи орендувати. Для виклику та завантаження необхідно виконати наступні дії:

– встановити на ЕОМ Node.js для запуску сервера UI на React, MySQL для бази даних, Spring вже має вбудований Apache Tomcat сервер;

– налаштувати та запустити програмне забезпечення;
– якщо використовуються локальний сервер, то після запуску достатньо ввести до пошуку у браузері URL: localhost:3000/, так як Node.js запускає сервер UI, використовуючи порт 3000 за замовчуванням. Spring використовує порт 8080, а MySQL – 3306;
– Якщо використовуються хост-сервери, то знадобиться використовувати домене ім’я для виклику застосунків, беручи до уваги звернення UI до «ендпоінтів»;
– Після введення дійсної адреси у вікні браузеру буде відкрито веб-орієнтований додаток.

2.6.4. Опис інтерфейсу користувача

Веб-орієнтований додаток має головну сторінку (рис. 2.15). При вході на сайт користувач побачить її першою.

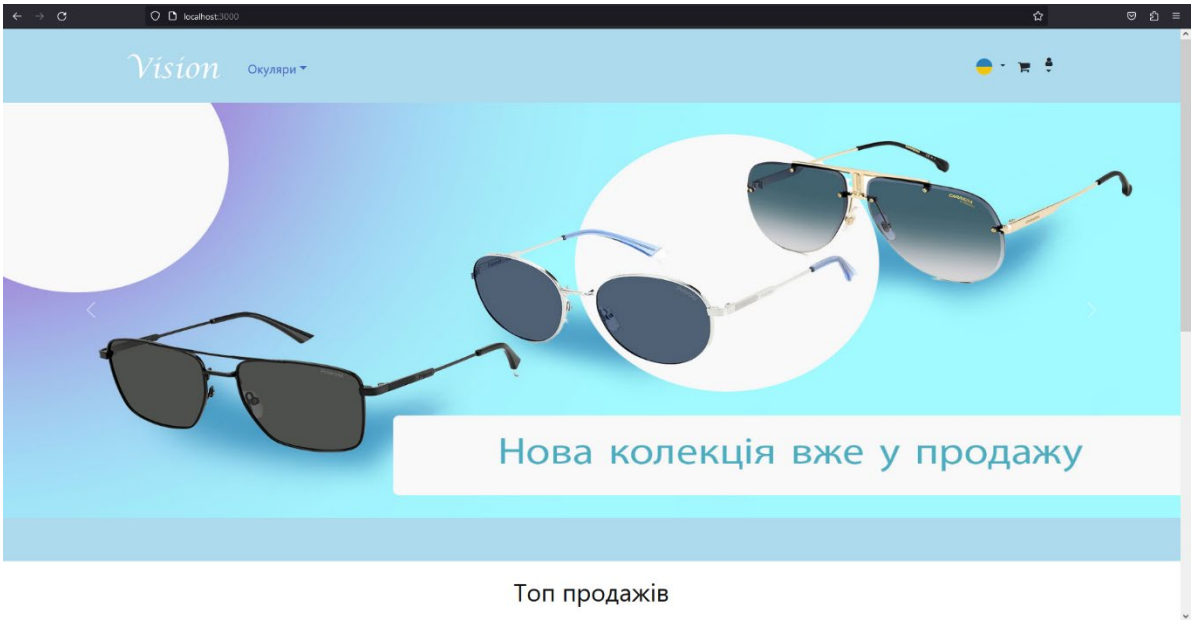

Рис. 2.15. Головна сторінка

Головна сторінка містить у своїй структурі баннер, товари, які продаються найбільше, та акційні пропозиції (рис. 2.16).

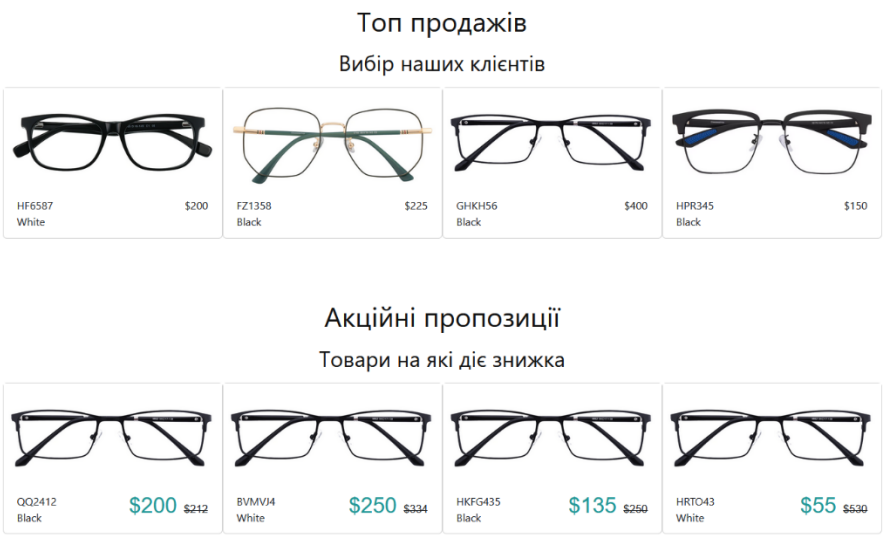

Рис. 2.16. Зміст головної сторінки

Локалізацію сайту можна змінювати у «хедері» (рис. 2.17 – 2.18). Вибір цього параметру вплине на мову тексту та формат ціни.

Рис. 2.17. Вибір локалізації

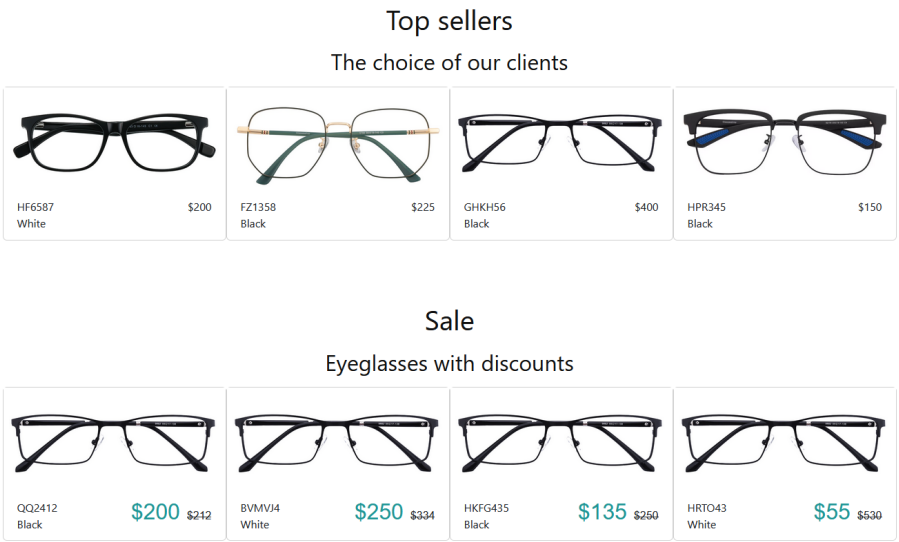

Рис. 2.18. Англійська локалізація

На сайті реалізована можливість реєстрації та авторизації користувача до системи (рис. 2.19). Якщо користувач неавторизований, у «хедері» сайту можна побачити значок користувача та натисканням на нього відкрити випадаюче меню.

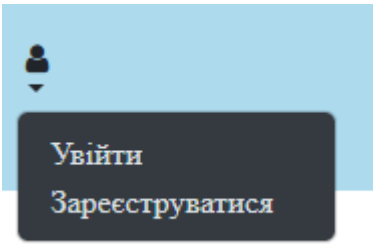

Рис. 2.19. Випадаюче меню авторизації

Після авторизації користувач побачить свій аватар, електрону пошту та кнопку «Вийти» (рис. 2.20).

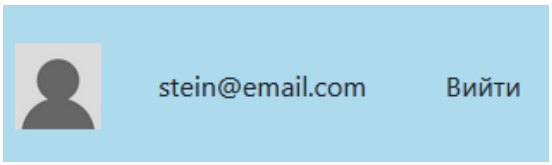

Рис. 2.20. «Хедер» після авторизації

Сторінка реєстрації містить дані користувача та аватар (рис. 2.21).

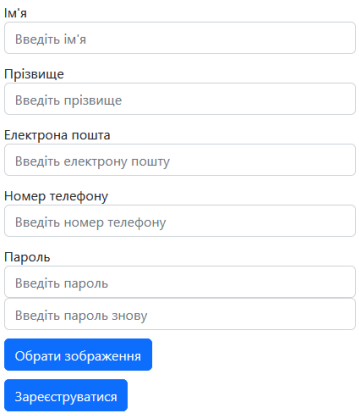

Рис. 2.21. Форма реєстрації

Якщо поля реєстрації не заповнені чи не відповідають вимогам, то користувач отримує повідомлення з помилкою. Наприклад, помилка невірного формату паролю (рис. 2.22).

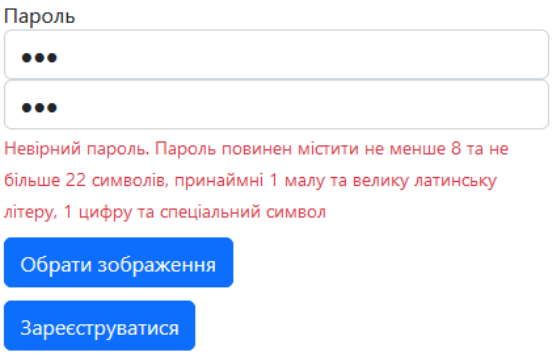

Рис. 2.22. Валідація

Сторінка авторизації (рис. 2.23) містить поле електронної пошти та паролю. Також як і при реєстраціє поля форми валідуються. Для того, щоб побачити уведений пароль, користувач може натиснути на значок ока.

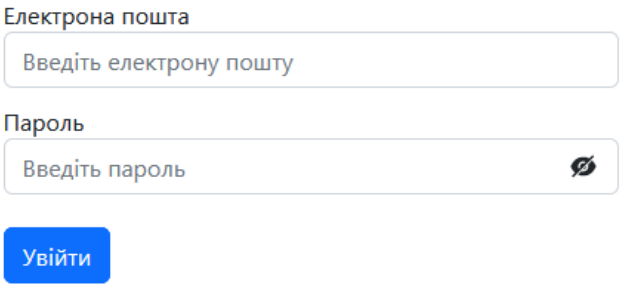

Рис. 2.23. Форма авторизації

У «хедері», натискаючи на «Окуляри» (рис. 2.24), користувач може обрати необхідну йому категорію товарів: чоловічі, жіночі та дитячі окуляри.

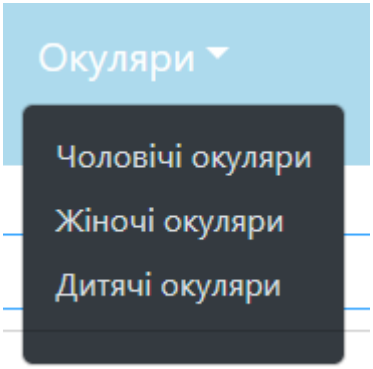

Рис. 2.24. Випадаюче меню каталогу

Сторінка каталогу містить список наявних окулярів за певною категорією, пагінацію, строку пошуку за назвою та фільтри (рис. 2.25).

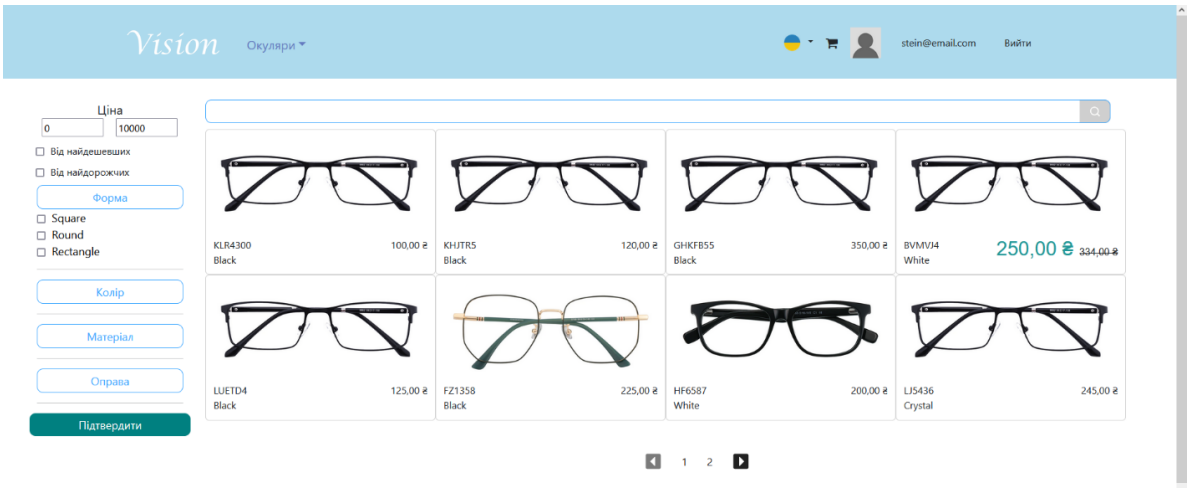

Рис. 2.25. Каталог

На рис. 2.26 було наведено приклад сортування по ціні від найдешевших товарів до найдорожчих. Кожен елемент фільтру записується до параметрів URL (рис. 2.27), тому, скопіювавши URL та вставивши його заново, користувач, якщо буде необхідно, може зберегти результати свого пошуку.

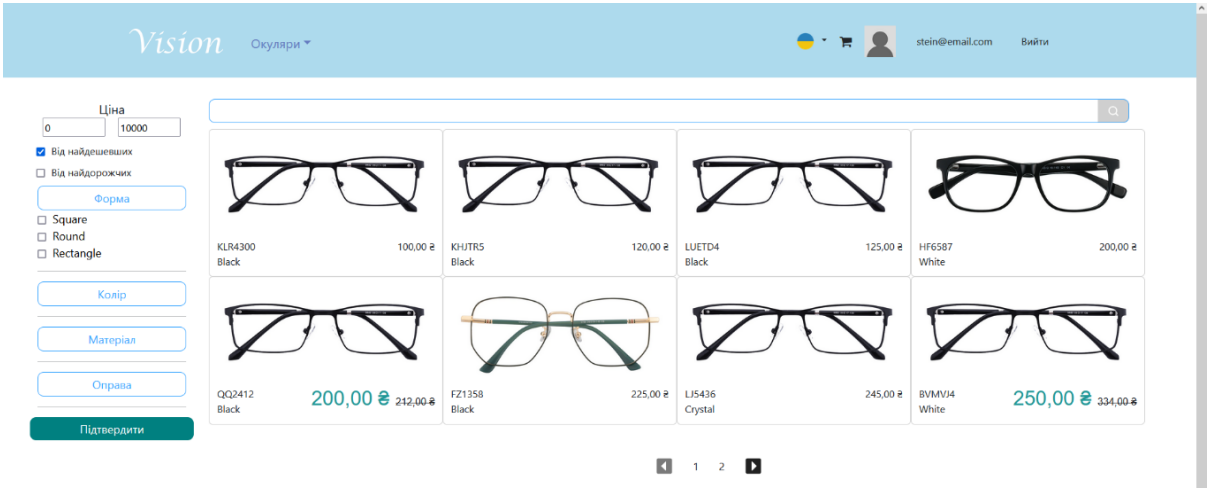

Рис. 2.26. Сортування

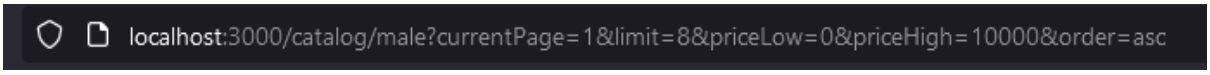

Рис. 2.27. URL каталогу

Щоб перейти до сторінки товару, потрібно навести курсор на карточку товару та натиснути кнопку «Детальніше» (рис. 2.28). Сторінка товару містить всю його необхідну інформацію (рис. 2.29). Між зображеннями можна перемикатися просто, якщо навести курсор. З права можна обрати необхідну кількість товарів та додати їх до кошика, якщо цього товару не буде на складі користувач побачить напис «Закінчився». Якщо на товар діє скидка, то користувач додатково побачить дату її закінчення (рис. 2.30).

Рис. 2.28. Картка товару

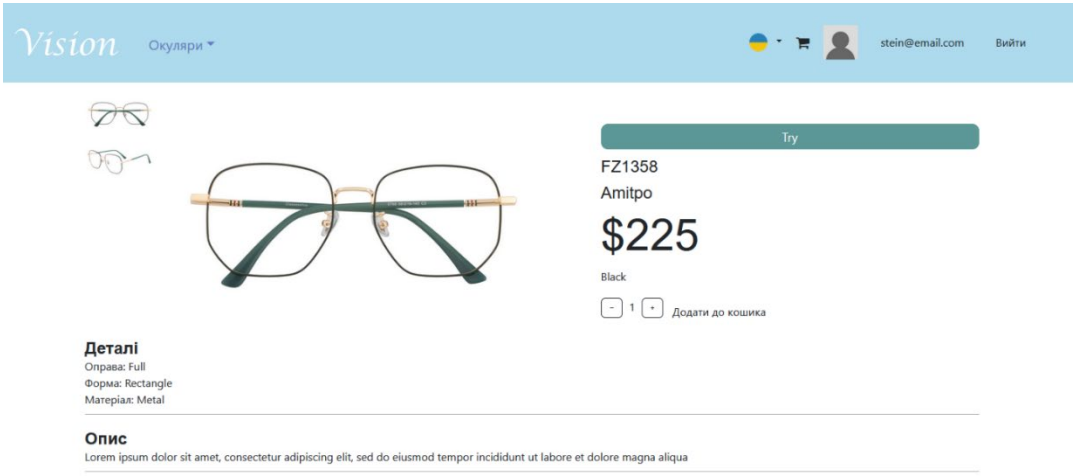

Рис. 2.29. Сторінка товару

Рис. 2.30. Акційне повідомлення

У окулярів можлива наявність альтернативних кольорів. Якщо такий товар наявний, то на його сторінці з’являється можливість обрати інший колір під його ціною (рис. 2.31).

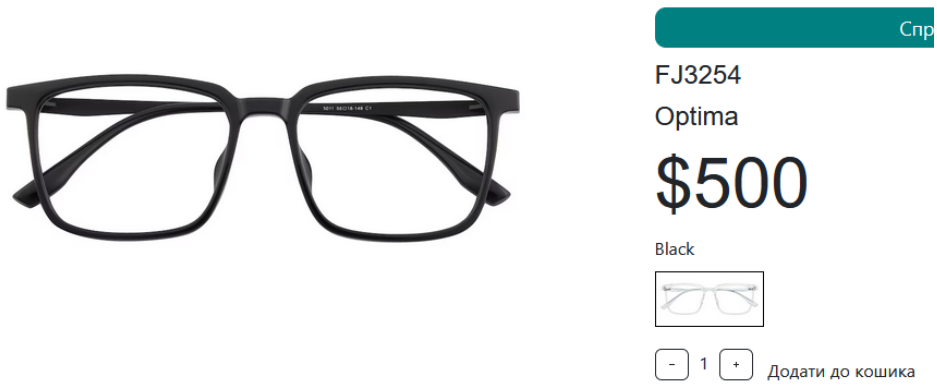

Рис. 2.31. Акційне повідомлення

Кнопка «Спробувати» на сторінці товару надає можливість користувачеві віртуально приміряти товар двома способами: завантаживши своє фото чи відстеженням лиця у реальному часі.

Перший спосіб (рис. 2.32) надає можливість перетягувати окуляри у необхідне місце та змінювати їх розмір. Кнопка «Очистити» прибирає зображення

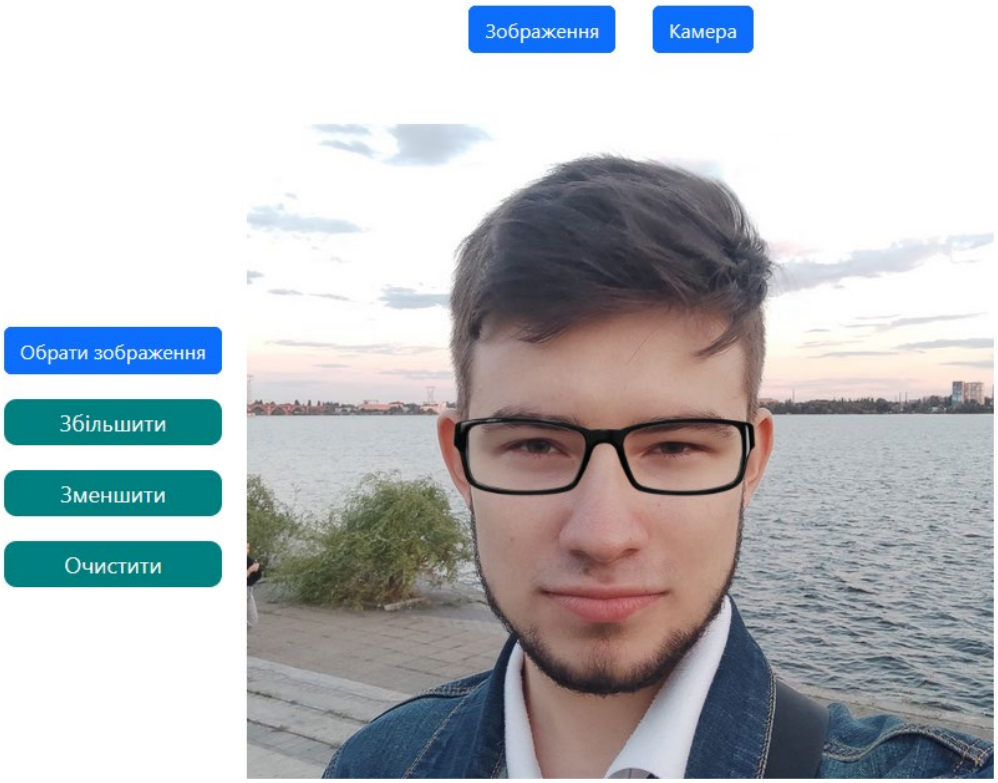

Рис. 2.32. Перший спосіб приміряти окуляри

Другий спосіб (рис. 2.33) відстежує положення обличчя та прикріплює у цьому місці зображення окулярів. Кнопка «Зупинити» - зупиняє малювання окулярів та залишає їх у фіксованому положенні. Кнопка «Очистити» прибирає окуляри.

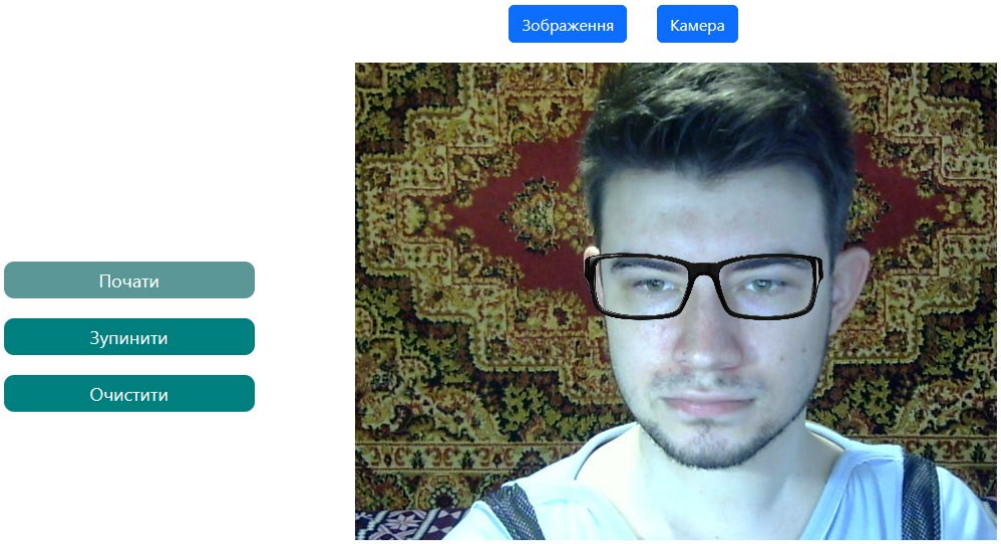

Рис. 2.33. Другий спосіб приміряти окуляри

Кошик (рис. 2.34) містить товари, які додав користувач, та зберігається у локальному сховищі, тому їм можуть користуватися навіть неавторизовані користувачі, але тоді їм буде запропоновано увійти/зареєструватися після натискання кнопки «Продовжити». Авторизований користувач зможе вести дані отримувача замовлення та обрати зручний йому метод оплати (рис. 2.35).

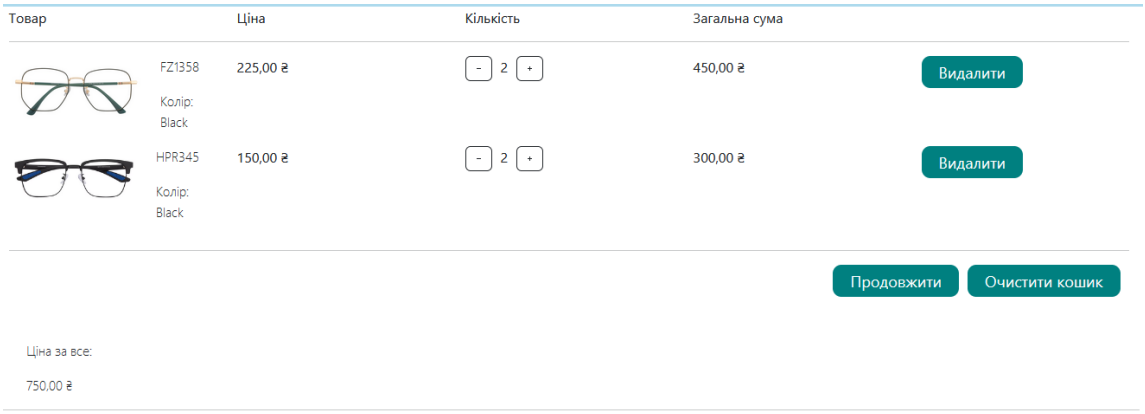

Рис. 2.34. Кошик

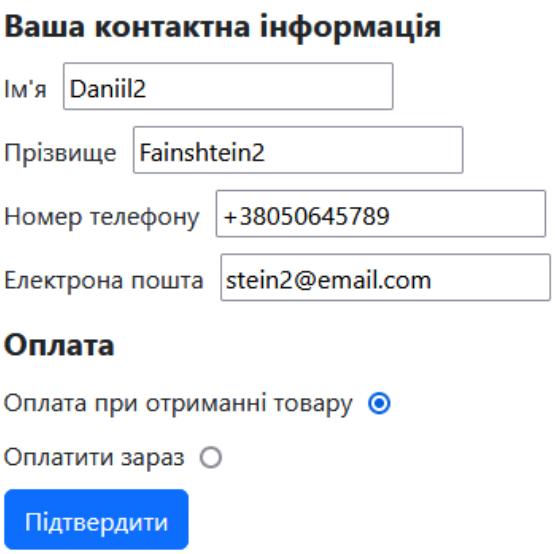

Рис. 2.35. Дані отримувача замовлення

Персональний кабінет користувача містить два розділи: Персональна інформація (рис. 2.36) та Управління замовленнями (рис. 2.37). Персональну інформацію користувач може редагувати (рис. 2.38). В управлінні замовленнями знаходяться товари, які замовив користувач.

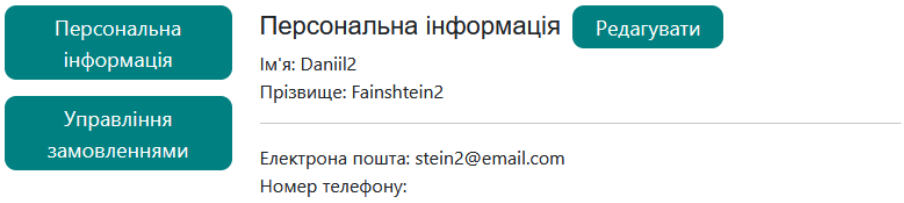

Рис. 2.36. Розділ «Персональна інформація»

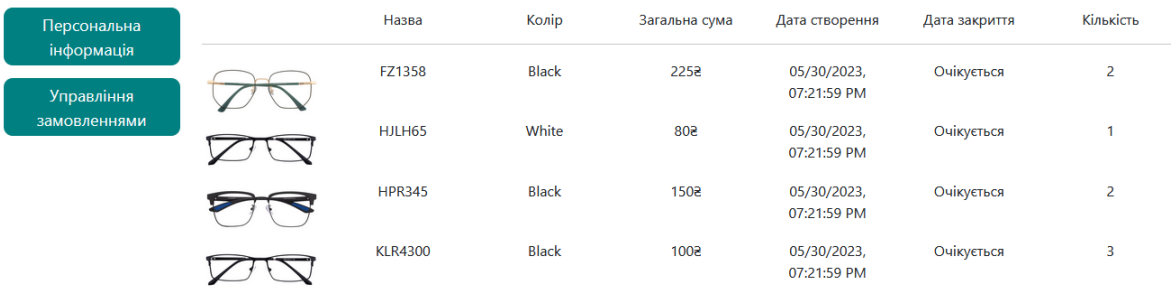

Рис. 2.37. Розділ «Управління замовленнями»

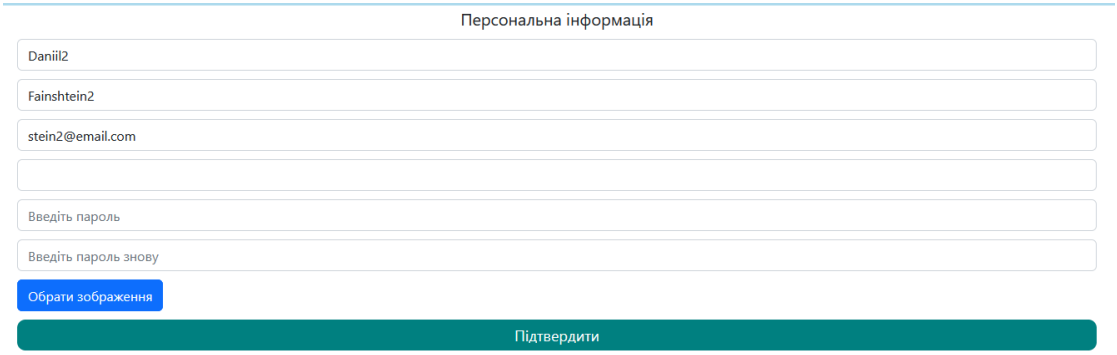

Рис. 2.38. Редагування персональної інформації

Адміністратор у своєму кабінеті має таку саму сторінку персональних даних, як і у користувача, а також ще два розділи, які відрізняються від розділів користувача (рис. 2.39).

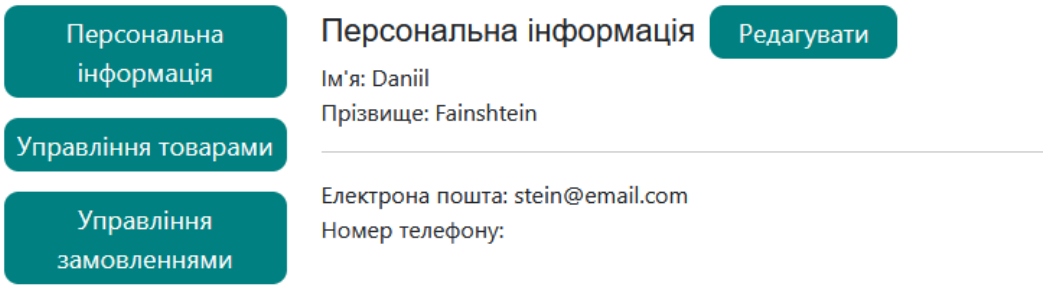

Рис. 2.39. Розділ «Персональна інформація» адміністратора

Розділ «Управління товарами» містить ще п’ять вкладок: Створити, Оновити, Створити альтернативний колір, Створити знижку та Видалити.

У розділі «Створити» (рис. 2.40) адміністратор створює товар, вводячи всі необхідні дані, такі як: назва, опис, постачальник, форма, матеріал, колір, оправа, категорія, ціна, кількість на складі та зображення. Додавання зображень реалізовано у вигляді drag’n’drop та їх порядок на сторінці можна налаштувати перетягнувши зображення до необхідної позиції (рис. 2.41).

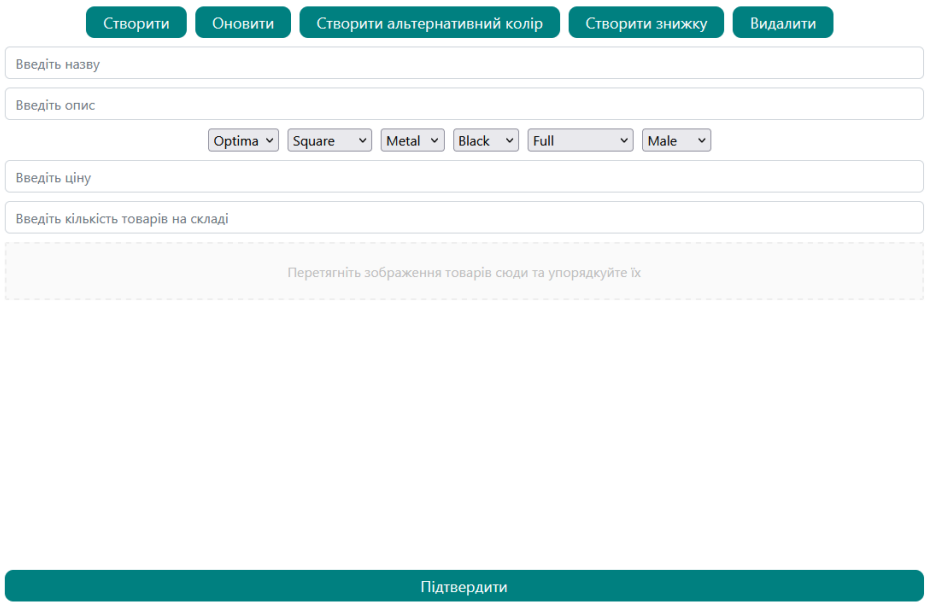

Рис. 2.40. Розділ «Створити»

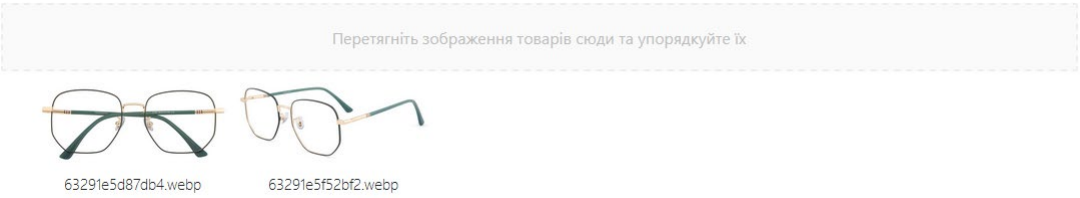

Рис. 2.41. drag’n’drop для зображень

У розділі «Оновити» (рис. 2.42) адміністратор шукає та оновлює інформацію про необхідний товар.

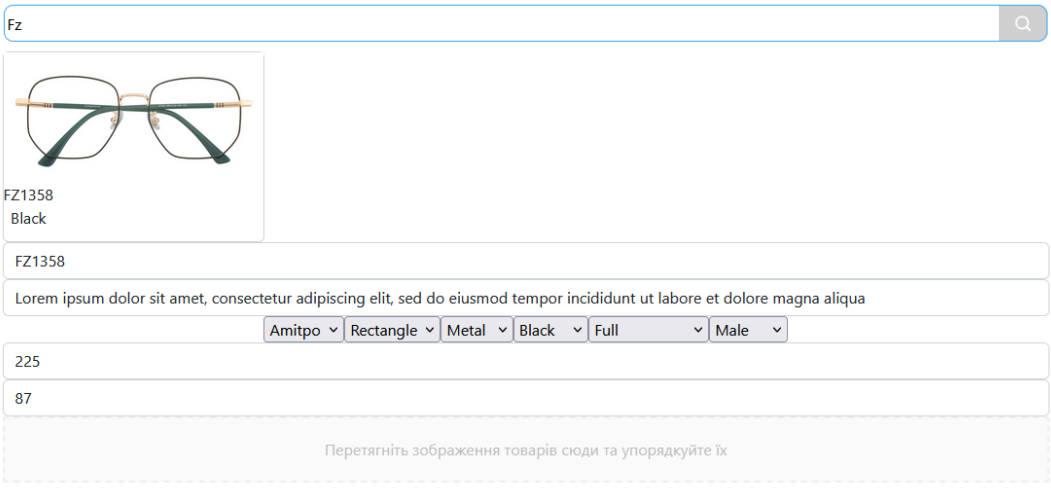

Рис. 2.42. Розділ «Оновити»

У розділі «Створити альтернативний колір» (рис. 2.43) адміністратор створює додатковий колір для вже існуючого товару та завантажує відповідно нові зображення.

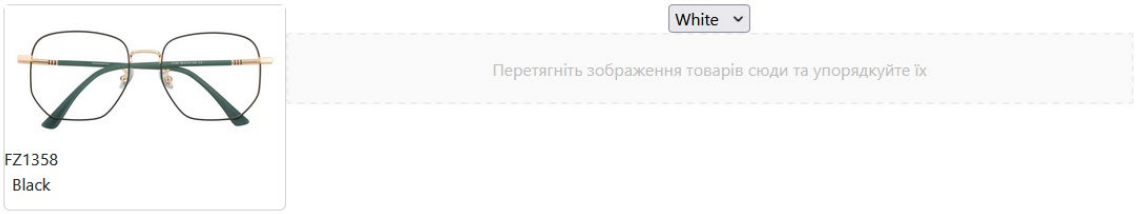

Рис. 2.43. Розділ «Створити альтернативний колір»

У розділі «Створити знижку» (рис. 2.44) адміністратор створює знижку на товар вводячи нову ціну та дату закінчення знижки.

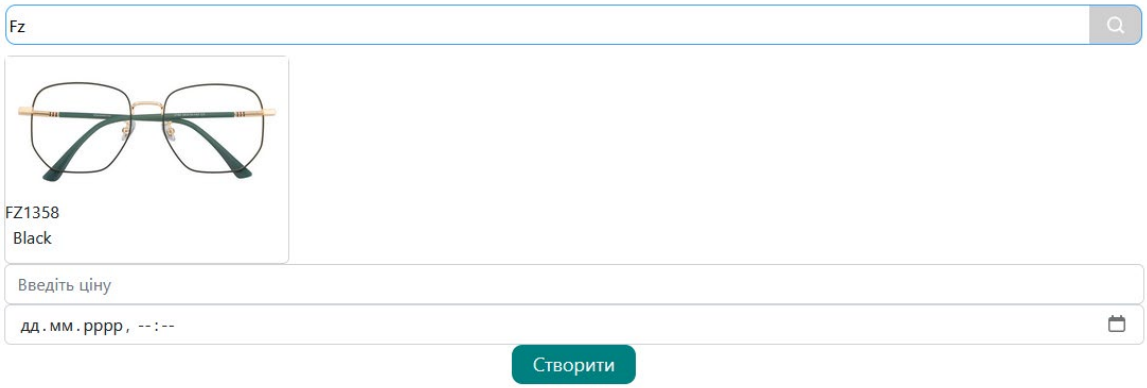

Рис. 2.44. Розділ «Створити знижку»

У розділі «Видалити» адміністратор видаляє необхідний товар з каталогу

Адміністратор у розділі «Управління замовленнями» (рис. 2.45) може підтвердити необхідне замовлення, що буде відображено користувачеві.

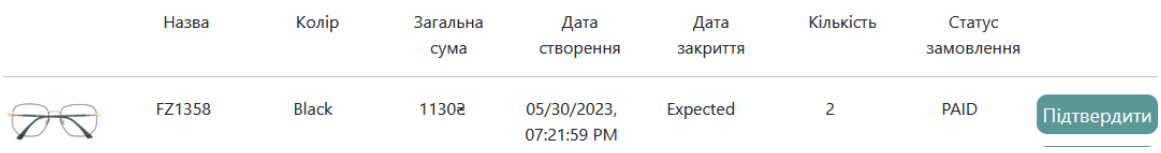

Рис. 2.45. Розділ «Управління замовленнями»

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

- передбачуване число операторів програми – 3000;
- коефіцієнт складності програми – 1,6;
- коефіцієнт корекції програми в ході її розробки – 0,1;
- годинна заробітна плата програміста – 210,1 грн/год;

Середня годинна зарплата Junior Full-Stack розробника в Україні була отримана використовуючи дані з «Української спільноти програмістів (DOU)»[24]. Станом на 2023 рік зарплата Junior Full-Stack розробника становить приблизно 1000 грн. При курсі валют НБУ на початок червня 2023 року один американський долар дорівнює 36,98 грн, тому середня зарплата в гривнях дорівнює 36 980 грн. При стандартному графіку (176 годин/місяць) зарплата за годину буде становити близько 210,1 грн.

- коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,25;
- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3;
- вартість машино-години ЕОМ – 20.1 грн/год.

Оскільки для цього проекту потрібна значна потужність ПК для роботи серверу та підтримки зберігання значної кількість даних на сервері баз даних, тому електроспоживання комп’ютера на годину дорівнює приблизно 300Вт. Ціна за 1 кВт/год при споживані понад 250 кВт на місяць станом на 1 червня 2023 становить 1.68 грн[25]. Тобто вартість години роботи комп’ютера дорівнює 0.3 * 1.68 = 0,504 грн.

Амортизація комп’ютерного обладнання (не враховуючи оновлення) становить:

$$А = \frac{Ф - Л}{ТВ} * 100\%, \quad (3.1)$$

де Ф – первісна вартість обладнання, що дорівнює 30000 грн,

Л – ліквідаційна вартість, що дорівнює вартості вивозу сміття – 109.71 грн,

ТВ – термін використання.

Термін використання, враховуючи, що у місяці приблизно 176 робочих годин і 10 місяців витрачених на розробку кваліфікаційної роботи становить:

$$176 * 10 = 1760 \text{ год},$$

Амортизація комп’ютерного обладнання за формулою (3.1) становить:

$$А = \frac{30000 - 109.71}{1760} * 100\% \approx 17{,}04 \text{ грн},$$

Ремонт та обслуговування ЕОМ склало 4000 грн, що при розподілі на 10 місяців становить:

$$\frac{4500}{1760} \approx 2.56 \text{ грн/год},$$

Загальна вартість машино-години ЕОМ склала:

$$0.504 + 17.04 + 2.56 \approx 20.1 \text{ грн/год},$$

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:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}, \quad (3.2)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

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;

$t_{\text{отл}}$ - витрати праці на налагодження програми на ЕОМ;

$t_{\text{д}}$ - витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p) = q * C * (1 + p), \quad (3.3)$$

де q – передбачуване число операторів (3000);

C – коефіцієнт складності програми (1,6);

p – коефіцієнт кореляції програми в ході її розробки (0,1).

За формулою (3.3) умовне число операторів складає:

$$Q = 3000 * 1{,}6 * (1 + 0{,}1) = 5280,$$

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

$$t_u = \frac{Q*B}{(75..85)*k}, \text{ людино-годин,} \quad (3.4)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,25),

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1,3).

Витрати праці на вивчення опису задачі за формулою (3.4) складають:

$$t_u = \frac{5280*1,25}{85*1,3} \approx 59,7, \text{ людино-годин,}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.5)$$

Виходячи з формули (3.5), витрати праці на розробку алгоритму рішення задачі складають:

$$t_a = \frac{5280}{25*1,3} \approx 162,5, \text{ людино-годин,}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.6)$$

За формулою (3.6) витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{5280}{25*1,3} \approx 162,5, \text{ людино-годин,}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)*k}, \text{ людино-годин,} \quad (3.7)$$

Використовуючи формулу (3.7) витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання дорівнюють:

$$t_{\text{отл}} = \frac{5280}{5*1,3} \approx 812,3, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1,5 * t_{\text{отл}}, \text{ людино-годин,} \quad (3.8)$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.8) складають:

$$t_{\text{отл}}^{k} = 1,5 * 812,3 \approx 1218,4, \text{ людино-годин,}$$

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{ людино-годин,} \quad (3.9)$$

де $t_{\text{др}}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{Q}{(15..20)*k}, \text{ людино-годин,} \quad (3.10)$$

$t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{\text{до}} = 0,75 * t_{\text{др}}, \text{ людино-годин,} \quad (3.11)$$

За формулами (3.9) – (3.11) витрати праці на підготовку документації складають:

$$t_{\text{др}} = \frac{5280}{20*1,3} \approx 203,1, \text{ людино-годин},$$

$$t_{\text{до}} = 0,75 * 203,1 \approx 152,3, \text{ людино-годин},$$

$$t_{\text{д}} = 203,1 + 152,3 = 355,4, \text{ людино-годин},$$

Розрахувавши всі показники, отримаємо трудомісткість розробки програмного забезпечення за формулою (3.2):

$$t = 50 + 59,7 + 162,5 + 162,5 + 812,3 + 355,4 = 1602,4, \text{ людино-годин.}$$

3.2. Витрати на створення програмного забезпечення

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

$$К_{\text{ПО}} = З_{\text{ЗП}} + З_{\text{МВ}}, \text{ грн.} \quad (3.12)$$

Заробітна плата виконавців визначається за формулою:

$$З_{\text{ЗП}} = t * С_{\text{ПР}}, \text{ грн,} \quad (3.13)$$

де: t - загальна трудомісткість, людино-годин (1602,4),

$С_{\text{ПР}}$ - середня годинна заробітна плата програміста, грн/година (210,1).

Враховуючи те, що середня годинна зарплата Junior Full-Stack розробника становить 210,1 грн/год, за формулою (3.13) заробітна плата виконавців:

$$З_{ЗП} = 1602{,}4 * 210{,}1 \approx 336\,664, \text{ грн},$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{Мч}, \text{ грн}, \quad (3.14)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год. (812,3),

$С_{Мч}$ - вартість машино-години ЕОМ, грн/год (20,1).

За формулою (3.14) вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = 812{,}3 * 20{,}1 \approx 16\,327, \text{ грн},$$

Виходячи з цього, за формулою (3.12) витрати на створення програмного забезпечення:

$$К_{ПО} = 336\,664 + 16\,327 = 352\,991, \text{ грн},$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{ міс}, \quad (3.15)$$

де B_k - число виконавців (1),

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

За формулою (3.15) очікуваний період створення програмного забезпечення:

$$T = \frac{1602,4}{1*176} \approx 9,1 \text{ міс.}$$

ВИСНОВКИ

Метою кваліфікаційної роботи було поставлено: створити веб-орієнтовану інформаційну систему з використанням бібліотеки React, призначену для управління та відображення інтернет-магазину оптики, яка би підвищила ефективність роботи магазину за рахунок діяльності в сфері електронної комерції.

Актуальність теми пов’язана із розширенням ринку електронної комерції та, беручи до уваги сучасні умови війни, з потребою у дистанційних засобах здійснення торгівлі, що є вигідним як для покупця, так і для продавця.

У цій інформаційній системі було розроблено інструмент для віртуального приміряння окулярів, як за допомогою завантаження фото, так і у реальному часі, з відстежуванням обличчя, що забезпечує зручний та покращений вибір товару. Такий функціонал не був наявний у вже готових існуючих рішеннях.

Зважаючи на те, що ІС була написана, використовуючи сучасні та популярні засоби розробки такі, як React та Spring, вона буде мати довгострокову та якісну технічну підтримку. А зручний та простий інтерфейс, який має веб-орієнтований додаток, розроблений у рамках кваліфікаційної роботи, допоможе користувачам, які мало знайомі з комп’ютером отримати кращий досвід у його використані.

Практичне призначення полягає у розробці веб-орієнтованого додатку, що сприяє ефективному функціонуванню інтернет-магазину шляхом поліпшення взаємодії з клієнтами та розширення аудиторії завдяки можливості здійснювати онлайн-торгівлю. Додаток забезпечує процеси замовлення, зберігання даних клієнтів та управління товарами. Крім того, розроблена система надає можливість візуально оцінити товар шляхом віртуального приміряння, швидко та зручно знаходити потрібні товари, надавати повну інформацію про них та використовувати систему знижок для збільшення обсягів продажів. В результаті створення такої системи компанія підвищить свою конкурентоспроможність та забезпечить успішну діяльність у сфері електронної комерції.

У процесі реалізації даного проекту були виконані наступні задачі:

– проаналізовано предметну галузь та інші наявні готові рішення
– створено алгоритм функціонування додатку
– створено базу даних для зберігання інформації
– створено API сайту для обробки запитів
– створено інтерфейс для сайту, який забезпечує взаємодію додатку з користувачем

Користувальницька частина була розроблена з використанням бібліотеки React, для якої були використані мови розмітки HTML, CSS та мова програмування JavaScript. Бізнес-логіка веб-додатку була реалізована на фреймворку Spring та мові програмування Java. Як СУБД було використано MySQL. Архітектура проекту використовує шаблон MVC та Rest API

Для розробки програмного продукту, призначеного для комерційної діяльності онлайн у сфері оптики, необхідно витратити 1602,4 людино-годин. З урахуванням цих даних можна припустити, що очікувана тривалість розробки продукту складатиме 9,1 місяців при стандартному робочому тижні та робочому місяці. Вартість цього програмного продукту складатиме 352 991 грн, включаючи заробітну плату фахівців, витрати на ЕОМ та витрати на його придбання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Люксоптика / URL: https://luxoptica.ua/ua/. дата звернення: 25.04.2023.
2. Dnipro_Optika / URL: https://dnipro-optika.com.ua. дата звернення: 25.04.2023.
3. Learn Java Programming / URL: https://www.programiz.com/java-programming. дата звернення: 10.05.2023.
4. What Is Java? / URL: https://aws.amazon.com/what-is/java/?nc1=h_ls. дата звернення: 10.05.2023.
5. What is JavaScript? / URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. дата звернення: 10.05.2023.
6. Haverbeke M. Eloquent JavaScript Eloquent JavaScript 3rd edition (2018) / M. Haverbeke. – San Francisco: No Starch Press, 2018. – 472 p.
7. Structured Query Language (SQL) / Peter Loshin / URL: https://www.techtarget.com/searchdatamanagement/definition/SQL. дата звернення: 10.05.2023.
8. Buna S. React Succinctly / S. Buna. – ebook, 2019. – 119 p. / Режим доступа: https://www.syncfusion.com/succinctly-free-ebooks/react-succinctly. дата звернення: 12.05.2023.
9. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux 1st Edition. – Sebastopol: O'Reilly Media, 2018. – 350 p.
10. Most used web frameworks among developers worldwide, as of 2022 / URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/. дата звернення: 12.05.2023.
11. Pros and Cons of React / Cezary Goralski / URL: https://thecodest.co/blog/pros-and-cons-of-react/. дата звернення: 12.05.2023.
12. Spring Boot / URL: https://stackshare.io/spring-boot#pros. дата звернення: 12.05.2023.

13. Pros and Cons of Using Spring Boot / URL: https://bambooagile.eu/insights/pros-and-cons-of-using-spring-boot/. дата звернення: 12.05.2023.

14. What is MySQL: MySQL Explained For Beginners / Richard B. / URL: https://www.hostinger.com/tutorials/what-is-mysql. дата звернення: 13.05.2023.

15. TensorFlow GitHub / URL: https://github.com/tensorflow/tensorflow дата звернення: 13.05.2023.

16. Why TensorFlow / URL: https://www.tensorflow.org/about дата звернення: 13.05.2023.

17. Getting Started. Promise based HTTP client for the browser and node.js / URL: https://axios-http.com/docs/intro. дата звернення: 13.05.2023.

18. What Is Bootstrap? / Jordana A. / URL: https://www.hostinger.com/tutorials/what-is-bootstrap/. дата звернення: 13.05.2023.

19. Visual Studio Code / https://code.visualstudio.com. дата звернення: 18.05.2023.

20. What is IntelliJ IDEA? / URL: https://www.jetbrains.com/idea/features/. дата звернення: 18.05.2023.

21. MySQL Workbench / URL: https://www.mysql.com/products/workbench/. дата звернення: 18.05.2023.

22. What is Node.js: A Comprehensive Guide / URL: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs. дата звернення: 18.05.2023.

23. What is Figma? / URL: https://www.nobledesktop.com/learn/figma/what-is-figma. дата звернення: 18.05.2023.

24. Зарплати українських розробників — зима 2023 / URL: https://dou.ua/lenta/articles/salary-report-devs-winter-2023/. дата звернення: 29.05.2023.

25. Тариф на електроенергію в залежності від місячних обсягів споживання / URL: https://yasno.com.ua/b2c-tariffs. дата звернення: 29.05.2023.

ДОДАТОК А

КОД ПРОГРАМИ

React

App.js // Маршрутизація

```
import React, { useState } from 'react';
import './App.css';
import Layout from './components/Layout';
import { Routes, Route } from 'react-router-dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import Catalog from './pages/Catalog';
import LoginForm from './pages/LoginForm';
import RegistrationForm from './pages/RegistrationForm';
import Product from './pages/Product';
import Cart from './pages/Cart';
import PersonalPage from './pages/PersonalPage';
import TryProduct from './pages/TryProduct';
import TryProductFacemesh from './components/catalog/TryProductFacemesh';
import Loading from './components/Loading';
import PageNotFound from './pages/PageNotFound';
import LocaleContext from './localization/LocaleContext';
import i18n from "i18next";
import "./localization/i18n";
import { Suspense } from 'react';
import AdminPersonalPage from './pages/AdminPersonalPage';
import EditPersonalData from './pages/EditPersonalData';
import Error from './pages/Error';

function App() {
  const [locale, setLocale] = useState(i18n.language);

  i18n.on('languageChanged', (lng) => setLocale(i18n.language));

  return (

    <LocaleContext.Provider value={{ locale, setLocale }}>
      <Suspense fallback={<Loading />}>
        <div className="App">
          <Header />
          <Routes>
            <Route path="/" element={<Layout />}>
              <Route path="/" element={<Home />}></Route>
              <Route path="/catalog/male/:currentPage?:limit?:shape?" element={<Catalog />}></Route>
              <Route path="/catalog/female/:currentPage?:limit?:shape?" element={<Catalog />}></Route>
              <Route path="/catalog/kids/:currentPage?:limit?:shape?" element={<Catalog />}></Route>
              <Route path="/login" element={<LoginForm />}></Route>
              <Route path="/reg" element={<RegistrationForm />}></Route>
              <Route path="/product/:id?" element={<Product />}></Route>
              <Route path="/cart" element={<Cart />}></Route>
              <Route exact path="/cabinet/personal" element={<PersonalPage showPersonal />}></Route>
              <Route exact path="/cabinet/orders" element={<PersonalPage showOrders />}></Route>
              <Route exact path="/cabinet/adminpersonal" element={<AdminPersonalPage showPersonal
/>}></Route>
              <Route exact path="/cabinet/adminproducts" element={<AdminPersonalPage showProducts
/>}></Route>
              <Route exact path="/cabinet/adminorders/getAllOrders/:currentPage?:limit?"
element={<AdminPersonalPage showOrders />}></Route>
              <Route path="/product/tryproduct" element={<TryProduct />}></Route>
              <Route path="/tryproductface" element={<TryProductFacemesh />}></Route>
```

```
            <Route path="/cabinet/personal/edit" element={<EditPersonalData />}></Route>

            <Route path="/error" element={<Error/>}/>

            <Route path="*" element={<PageNotFound />} />
           </Route>
          </Routes>
          <Footer />
         </div>
     </Suspense>
   </LocaleContext.Provider>

  );
}

export default App;
```

Home.js // Головна сторінка

```
import React, { useEffect, useState } from 'react'
import { Link, useNavigate, Navigate } from "react-router-dom";
import Banner from '../components/Banner';
import api from '../api/axiosConfig';
import { useTranslation } from 'react-i18next';

const Home = () => {
  const navigate = useNavigate();
  const [populars, setPopulars] = useState([]);
  const [discounts, setDiscounts] = useState([]);
  const [loading, setLoading] = useState(true);
  const { t } = useTranslation();
  useEffect(() => {
    const getPopular = async () => {
      try {
        const response = await api.get(
          "http://localhost:8080/popular"
        );
        setPopulars(response.data);
      } catch (err) {
        setPopulars(null);
        navigate("/error", { err });
      } finally {
        setLoading(false);
      }
    };
    const getDiscounts = async () => {
      try {
        const response = await api.get(
          "http://localhost:8080/getDiscounts"
        );
        setDiscounts(response.data);
      } catch (err) {
        setDiscounts(null);
      } finally {
        setLoading(false);
      }
    };

    getPopular();
    getDiscounts();
  }, []);
  function handleCheck(val) {
    return discounts.some(item => val.id === item.eyeglassId);
  }
```

```
function findArrayElementByTitle(id) {
  var result = discounts.find(obj => {
    return obj.eyeglassId === id
  })
  console.log(result);
  return result.discountPrice;
}
if (loading) {
  return <p>Loading...</p>;
}
return (
  <div>
    <Banner />
    <div className="appeal-section">
      <div className="container">
        <h1> {t('appeal_section_header')} </h1>
        <h2> {t('appeal_section_text')}</h2>
        {populars.map((popular, index) => (
          <div className="col-xs-12 col-sm-6 col-md-4 col-lg-3 appeal_products card" key={index}>
            <div className="txthover">
              <img src={require("../images/products/" + popular.img)} alt={popular.name} />
              <div className="product_card_readmore txtcontent">
                <div className="simpletxt">
                  <Link to={"/product?id=" + popular.eyeglassId + "&color=" + popular.color}
className="readmore text-left">{t('read_more')}</Link><br />
                </div>
                <div className="stars2">
                  <div className="glyphicon glyphicon-star"></div>
                  <div className="glyphicon glyphicon-star"></div>
                  <div className="glyphicon glyphicon-star"></div>
                </div>
              </div>
            </div>
            <div className="product_card_signature">
              <div className="sale_card_signature_name">{popular.name}<br />{popular.color}</div>
              <div className="sale_card_signature_price">
                {handleCheck(popular)
                ?
                <div>
                  {console.log(findArrayElementByTitle(popular.eyeglassId))}
                  <span
className="catalog_product_price">${findArrayElementByTitle(popular.eyeglassId)}</span>
                    <span className="catalog_product_discount_price">${popular.price}</span>
                </div>
                :
                <div>
                  <span className="">${popular.price}</span>
                </div>
              }
            </div>
          </div>
        </div>
      ))}
    </div>
  </div>
  <div className="discont_section">
    <div className="container">
      <h1>{t('sale')}</h1>
      <h2>{t('sale_section_text')}</h2>
      <div>
        {discounts.map((discount, index) => (
          <div className="col-xs-12 col-sm-6 col-md-4 col-lg-3 appeal_products card" key={index}>
            <div className="txthover">
```

```
                    <img src={require("../images/products/" + discount.img)} alt={discount.name} />
                    <div className="product_card_readmore txtcontent">
                     <div className="simpletxt">
                      <Link to={"/product?id=" + discount.eyeglassId + "&color=" + discount.color}
className="readmore text-left">{t('read_more')}</Link><br />
                     </div>
                     <div className="stars2">
                      <div className="glyphicon glyphicon-star"></div>
                      <div className="glyphicon glyphicon-star"></div>
                      <div className="glyphicon glyphicon-star"></div>
                     </div>
                    </div>
                   </div>
                   <div className="product_card_signature">
                    <div className="sale_card_signature_name">{discount.name}<br />{discount.color}</div>
                    <div className="sale_card_signature_price">
                     <div>
                      <span className="catalog_product_price">${discount.discountPrice}</span>
                      <span className="catalog_product_discount_price">${discount.originalPrice}</span>
                     </div>
                    </div>
                   </div>
                  </div>
                 ))}
                </div>
               </div>
              </div>
             </div>
            )
           }

           export default Home

           Header.js // «Хедер»
           import React, { useState, useEffect, useContext } from 'react'
           import { Link, useLocation, useNavigate } from "react-router-dom";
           import Container from 'react-bootstrap/Container';
           import Nav from 'react-bootstrap/Nav';
           import Navbar from 'react-bootstrap/Navbar';
           import NavDropdown from 'react-bootstrap/NavDropdown';
           import Button from 'react-bootstrap/esm/Button';
           import api from '../api/axiosConfig';
           import UkraineFlagIco from '../images/ua_flag.png';
           import USAFlagIco from '../images/usa_flag.png';
           import i18n from "i18next";
           import { useTranslation } from 'react-i18next';
           import LocaleContext from '../localization/LocaleContext';

           const Header = () => {
            const { locale } = useContext(LocaleContext);
            const { t } = useTranslation();
            const navigate = useNavigate();
            const [user, setUser] = useState();
            const [avatar, setAvatar] = useState();
            const [nextLink, setNextLink] = useState("");
            const [loading, setLoading] = useState(true);
            const location = useLocation();
            useEffect(() => {
             setUser(JSON.parse(localStorage.getItem('user')));
            }, [location.state]);
            useEffect(() => {
             const getAvatar = async () => {
              try {
```

```
        const response = await api.get(
          "http://localhost:8080/getAvatar?email=" + user.username, { responseType: 'blob' });
        setAvatar(URL.createObjectURL(response.data));
       } catch (err) {
        setAvatar(null);
       } finally {
        setLoading(false);
       }
      };

      getAvatar();
    }, [user]);
    async function next() {
      let token = user.token;
      try {
        const response = await api.get(
          "http://localhost:8080/cabinet", {
          headers: {
            'Accept': 'application/json, text/plain, */*',
            'Content-Type': 'application/json',
            'Authorization': "Bearer " + token,
          }
        }
        );
        navigate(response.data.link);
      } catch (err) {

      } finally {
        setLoading(false);
      }
    }
    function changeLocale(l) {
      if (locale !== l) {
        i18n.changeLanguage(l);
      }
    }
    if (loading) {
      return <p>Loading...</p>;
    }
    return (
      <div className="header_section">
        <div className="navbar-custom">
          <Navbar variant="dark" expand="lg">
            <Container fluid>
              <Navbar.Brand><Link to="/" className="text-left text-reset"><img
src={require("../images/logo_Vision.png")} width="150px" /></Link></Navbar.Brand>
              <Navbar.Toggle aria-controls="navbar-dark-example" />
              <Navbar.Collapse id="navbar-dark-example">
                <Nav>
                  <NavDropdown id="nav-dropdown-dark-example" className="header_product_list"
title={t('eyeglasses')} menuVariant="dark">
                    <NavDropdown.Item
href="/catalog/male?currentPage=1&limit=5">{t('mens_eyeglasses')}</NavDropdown.Item>
                    <NavDropdown.Item href="/catalog/female?currentPage=1&limit=5">
{t('womens_eyeglasses')}</NavDropdown.Item>
                    <NavDropdown.Item href="/catalog/kids?currentPage=1&limit=5">
{t('kids_eyeglasses')}</NavDropdown.Item>
                    <NavDropdown.Divider />
                  </NavDropdown>
                </Nav>
              </Navbar.Collapse>
              <Navbar.Collapse className="justify-content-end">
                <NavDropdown id="nav-dropdown-dark-example" className="navbar_flag_dropdown"
```

```
                    title={
                      locale === "en" ?
                        <img className="locale_flag" src={USAFlagIco} />
                        :
                        <img className="locale_flag" src={UkraineFlagIco} />
                    }
                    menuVariant="dark"
                  >
                    <NavDropdown.Item onClick={() => changeLocale('en')}><div className="locale_item"><img
className="locale_flag" src={USAFlagIco} /><div>EN</div></div> </NavDropdown.Item>
                    <NavDropdown.Item onClick={() => changeLocale('uk-Ua')}><img className="locale_flag"
src={UkraineFlagIco} />UK</NavDropdown.Item>
                  </NavDropdown>
                  <Link to="/cart" className="navbar_flag_dropdown text-left text-reset"><i className="fa-lg fa-
shopping-cart cart_icon"></i></Link>
                  {(user) ?
                    <div className="navbar_end">
                      <Nav.Link href="#link"><img className="avatar" src={avatar} width="30px" /></Nav.Link>
                      <button className="btn btn-link text-reset" onClick={next}>{user.username}</button>
                      <Nav.Link href="/"><Button variant="contained" onClick={() => { localStorage.clear(); setUser()
}}>{t('Log out')}</Button></Nav.Link>
                    </div>
                    :
                    <i className="fa fa-fw fa-user">
                      <NavDropdown id="nav-dropdown-dark-example" menuVariant="dark">
                        <NavDropdown.Item href="/login">{t('sign_in')}</NavDropdown.Item>
                        <NavDropdown.Item href="/reg">{t('sign_up')}</NavDropdown.Item>
                      </NavDropdown>
                    </i>
                  }
                </Navbar.Collapse>
              </Container>
            </Navbar>
          </div>
        </div>
      )
    }

    export default Header

    Products.js // Компонент відповідальний за вивід товарів у каталог
    import React, { useEffect, useState } from 'react'
    import { Link, useSearchParams } from "react-router-dom";
    import api from '../../api/axiosConfig';
    import Pagination from './Pagination';
    import { useTranslation } from 'react-i18next';

    const Products = () => {
      let PageSize = 8;
      const { t } = useTranslation();
      const [searchParams, setSearchParams] = useSearchParams();
      function getCurrentURL() {
        let updatedSearchParams = new URLSearchParams(searchParams.toString());
        setCurrentPage(updatedSearchParams.get('currentPage') * 1);
        updatedSearchParams.set('limit', PageSize);
        if (updatedSearchParams.has('name') &&
          (updatedSearchParams.get('name') === "" || updatedSearchParams.get('name') === undefined)) {
          updatedSearchParams.delete('name');
        }
        setSearchParams(updatedSearchParams.toString());
        let currentURL = window.location.href;
        let processedURL = currentURL.replace(3000, 8080);
        let replacement = "";
```

```
    if (currentPage === undefined) {
      replacement = "currentPage=" + 1;
    } else {
      replacement = "currentPage=" + (currentPage - 1);
    }
    updatedSearchParams.set('currentPage', currentPage + 1);
    processedURL = processedURL.substring(0, processedURL.indexOf("currentPage=")) + replacement
      + processedURL.substring(processedURL.indexOf("currentPage=") + replacement.length);
    return processedURL;
}
function setCurrentPageToURL(page) {
    let updatedSearchParams = new URLSearchParams(searchParams.toString());
    updatedSearchParams.set('currentPage', page);
    updatedSearchParams.set('limit', PageSize);
    if (updatedSearchParams.has('name') &&
      (updatedSearchParams.get('name') === "" || updatedSearchParams.get('name') === undefined)) {
      updatedSearchParams.delete('name');
    }
    setSearchParams(updatedSearchParams.toString());
    setCurrentPage(page);
}
const [currentPage, setCurrentPage] = useState();
const [data, setData] = useState([]);
const [discounts, setDiscounts] = useState([]);
const [loading, setLoading] = useState(true);
useEffect(() => {
    const getDiscounts = async () => {
      try {
        const response = await api.get(
          "http://localhost:8080/getDiscounts"
        );
        setDiscounts(response.data);
      } catch (err) {
        setDiscounts(null);
      } finally {
      }
    };

    getDiscounts();
}, []);
useEffect(() => {
    const getData = async () => {
      try {
        const response = await api.get(
          getCurrentURL()
        );
        setData(response.data);
      } catch (err) {
        setData(null);
      } finally {
        setLoading(false);
      }
    };

    getData();
}, [currentPage]);
function handleCheck(val) {
    return discounts.some(item => val.id === item.eyeglassId);
}
function findArrayElementByTitle(id) {
    var result = discounts.find(obj => {
      return obj.eyeglassId === id
    })
```

```
            return result.discountPrice;
        }
        if (loading) {
            return <p>Loading...</p>;
        }
        return (
            <div>
                <div className="row catalog_row">
                    <div className="catalog_search">
                        <input className="catalog_search_field" type="textarea" name="name" />
                        <input type="submit" className="search_button" value="" />
                    </div>
                    {data.content.map((eyeglass, index) => (
                        <div className="col-xs-12 col-sm-6 col-md-4 col-lg-3 product_card card" key={index}>
                            <div className="txthover">
                                <img src={require("../../images/products/" + eyeglass.img)} alt={eyeglass.name} />
                                <div className="product_card_readmore txtcontent">
                                    <div className="simpletxt">
                                        <Link to={"/product?id=" + eyeglass.id + "&color=" + eyeglass.color}
className="readmore text-left">{t('read_more')}</Link><br />
                                    </div>
                                    <div className="stars2">
                                        <div className="glyphicon glyphicon-star"></div>
                                        <div className="glyphicon glyphicon-star"></div>
                                        <div className="glyphicon glyphicon-star"></div>
                                    </div>
                                </div>
                            </div>
                            <div className="product_card_signature">
                                <div                    className="product_card_signature_name">{eyeglass.name}<br
/>{eyeglass.color}</div>
                                <div className="product_card_signature_price">
                                    {handleCheck(eyeglass)
                                        ?<div>
                                            <span        className="catalog_product_price">{t('price_format',        {        value:
findArrayElementByTitle(eyeglass.id) })}</span>
                                            <span    className="catalog_product_discount_price">{t('price_format',    {    value:
eyeglass.price })}</span>
                                        </div>
                                        :<div><span className="">{t('price_format', { value: eyeglass.price })}</span></div>
                                    }
                                </div>
                            </div>
                        </div>
                    ))}
                </div>
                <div className="row pagination_section_row">
                    <div className="pagination_section">
                        <Pagination                    className="pagination-bar"                    currentPage={currentPage}
totalCount={data.totalElements} pageSize={PageSize} onPageChange={page => setCurrentPageToURL(page)}/>
                    </div>
                </div>
            </div>
        )
    }

    export default Products

    Catalog.js // Сторінка каталогу
    import React from 'react'
    import { useSearchParams } from "react-router-dom";
    import Filters from '../components/catalog/Filters';
    import Products from '../components/catalog/Products';
```

```
const Catalog = () => {
   const [searchParams, setSearchParams] = useSearchParams();
   function getAction() {
      let currentURL = window.location.href;
      let action = currentURL.substring(currentURL.indexOf("/"), currentURL.indexOf("?"));
      return action;
   }
   const handleSubmit = (e) => {
      let updatedSearchParams = new URLSearchParams(searchParams.toString());
      if (updatedSearchParams.has('currentPage')) {
         updatedSearchParams.set('currentPage', "1");
      }
      setSearchParams(updatedSearchParams.toString());
   }
   return (
      <div className="catalog_container container-fluid">
         <form action={getAction()} method="GET" onSubmit={handleSubmit}>
            <div className="catalog_section">
               <div className="row m-0 w-100">
                  <div className="filters_col col-2">
                     <Filters />
                  </div>
                  <div className="col">
                     <div className="row m-0">
                        <Products />
                     </div>
                  </div>
               </div>
            </div>
         </form>
      </div>
   )
}

export default Catalog
```

Filters.js // Компонент фільтрів для каталогу

```
import React, { useEffect, useState } from 'react'
import { useSearchParams } from "react-router-dom";
import BasicDropdown from "./BasicDropdown";
import api from '../../api/axiosConfig';
import { useTranslation } from 'react-i18next';

const Filters = () => {
  const { t } = useTranslation();
  const [searchParams, setSearchParams] = useSearchParams();
  const [shapes, setShapes] = useState([]);
  const [colors, setColors] = useState([]);
  const [materials, setMaterials] = useState([]);
  const [frames, setFrames] = useState([]);
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    const getShapes = async () => {
      try {
        const response = await api.get(
          "http://localhost:8080/allShapes"
        );
        setShapes(response.data);
      } catch (err) {
        setShapes(null);
      } finally {
```

```
    }
  };
  const getColors = async () => {
    try {
      const response = await api.get(
        "http://localhost:8080/allColors"
      );
      setColors(response.data);
    } catch (err) {
      setColors(null);
    } finally {

    }
  };
  const getMaterials = async () => {
    try {
      const response = await api.get(
        "http://localhost:8080/allMaterials"
      );
      setMaterials(response.data);
    } catch (err) {
      setMaterials(null);
    } finally {

    }
  };
  const getFrames = async () => {
    try {
      const response = await api.get(
        "http://localhost:8080/allFrames"
      );
      setFrames(response.data);
    } catch (err) {
      setFrames(null);
    } finally {

    }
  };
  try {
    getShapes();
    getColors();
    getMaterials();
    getFrames();

  } finally {
    setLoading(false);
  }
}, []);
function handleOrder(order) {
  if (order === "asc") {
    if (document.getElementById("priceAsc").checked === true) {
      document.getElementById("priceDesc").disabled = true;
      document.getElementById("priceAsc").disabled = false;
    } else {
      document.getElementById("priceDesc").disabled = false;
      document.getElementById("priceAsc").disabled = false;
    }
  } else if (order === "desc") {
    if (document.getElementById("priceDesc").checked === true) {
      document.getElementById("priceAsc").disabled = true;
      document.getElementById("priceDesc").disabled = false;
    } else {
```

```
            document.getElementById("priceAsc").disabled = false;
            document.getElementById("priceDesc").disabled = false;
          }
        }
      }
      if (loading) {
        return <p>Loading...</p>;
      }
      return (
        <div>
          <div className="left-filters d-none d-lg-block">
            <div className="sticky-top">
              <div className="collapse mobile-collapse d-lg-block" id="Filters">
                <div className="col-12">
                  <ul className="filter-wrap">
                    <input type="hidden" name="currentPage" value={searchParams.get("currentPage")} />
                    <input type="hidden" name="limit" value={searchParams.get("limit")} />
                    <li className="filters_price position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                      <div className="filters_price_title">{t('price')}</div>
                      <div className="price_limits">
                        {searchParams.has("priceLow") ? <div className="price_limit"><input name="priceLow" defaultValue={searchParams.get("priceLow")} /></div> : <div className="price_limit"><input name="priceLow" defaultValue="0" /></div>}
                        {searchParams.has("priceLow") ? <div className="price_limit"><input name="priceHigh" defaultValue={searchParams.get("priceHigh")} /></div> : <div className="price_limit"><input name="priceHigh" defaultValue="10000" /></div>}
                      </div>
                    </li>
                    <li className="filters_price_order position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                      <label className="checkbox-inline p-lg-0 pb-3" htmlFor="priceAsc">
                        {searchParams.has("order") && searchParams.get("order") === "asc" ? <input id="priceAsc" type="checkbox" className="form-checkbox mt-n1" name="order" value="asc" onChange={() => handleOrder("asc")} defaultChecked /> : <input id="priceAsc" type="checkbox" className="form-checkbox mt-n1" name="order" value="asc" onChange={() => handleOrder("asc")} />}
                        {t('ascending')}
                      </label>
                    </li>
                    <li className="filters_price_order position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                      <label className="checkbox-inline p-lg-0 pb-3" htmlFor="priceDesc">
                        {searchParams.has("order") && searchParams.get("order") === "desc" ? <input id="priceDesc" type="checkbox" className="form-checkbox mt-n1" name="order" value="desc" onChange={() => handleOrder("desc")} defaultChecked /> : <input id="priceDesc" type="checkbox" className="form-checkbox mt-n1" name="order" value="desc" onChange={() => handleOrder("desc")} />}
                        {t('descending')}
                      </label>
                    </li>
                    <li className="position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                      <div className="filter-down filter-down-shape">
                        <div className="row m-0">
                          <BasicDropdown name={t('shape')} itemName={"shape"} items={shapes} isActive={true} param={searchParams.has("shape") ? searchParams.getAll("shape") : null} />
                        </div>
                      </div>
                    </li>
                    <li className="position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                      <div className="filter-down filter-down-shape filter_item">
                        <div className="row m-0">
                          {searchParams.has("color") ? <BasicDropdown name={t('color')} itemName={"color"} items={colors} isActive={true} param={searchParams.getAll("color")} /> : <BasicDropdown name={t('color')} itemName={"color"} items={colors} isActive={false} param={null} />}
```

```
                    </div>
                   </div>
                  </li>
                  <li className="position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                   <div className="filter-down filter-down-shape">
                    <div className="row m-0">
                     {searchParams.has("material") ? <BasicDropdown name={t('material')} itemName={"material"}
items={materials}  isActive={true}  param={searchParams.getAll("material")}  />  :  <BasicDropdown
name={t('material')} itemName={"material"} items={materials} isActive={false} param={null} />}
                    </div>
                   </div>
                  </li>
                  <li className="position-relative col-12 col-lg-auto p-0 filter-item active filter-item-shape">
                   <div className="filter-down filter-down-shape">
                    <div className="row m-0">
                     {searchParams.has("frame")  ?  <BasicDropdown  name={t('frame')}  itemName="frame"
items={frames} isActive={true} param={searchParams.getAll("frame")} /> : <BasicDropdown name={t('frame')}
itemName={"frame"} items={frames} isActive={false} param={null} />}
                    </div>
                   </div>
                  </li>
                  <input type="submit" className="apply_filters_button" value={t('confirm')} />
                 </ul>
                </div>
               </div>
              </div>
             </div>
            </div>
           )
          }

          export default Filters

          TryProduct.js // Сторінка «Спробувати» на сторінці продукту
          import React, { useEffect, useState, useContext } from 'react'
          import { useNavigate } from "react-router-dom";
          import $ from 'jquery';
          import 'jquery-ui-bundle';
          import 'jquery-ui-bundle/jquery-ui.css';
          import TryProductFacemesh from "../components/catalog/TryProductFacemesh";
          import { useTranslation } from 'react-i18next';
          import api from '../api/axiosConfig';
          import LocaleContext from '../localization/LocaleContext';

          const TryProduct = () => {
            const navigate = useNavigate();
            const { locale } = useContext(LocaleContext);
            let mousePos = { x: undefined, y: undefined };
            const [open, setOpen] = React.useState(false);
            const [tryProductImg, setTryProductImg] = useState();
            const handleOpen = () => setOpen(true);
            const handleClose = () => setOpen(false);
            const [image, setImage] = useState([]);
            const { t } = useTranslation();
            function getCurrentURL() {
              let currentURL = window.location.href;
              currentURL = currentURL.replace("3000", "8080");
              return currentURL;
            }
            useEffect(() => {
              const getData = async () => {
                try {
                  const response = await api.get(
```

```
                getCurrentURL(), {
                  headers: {
                    'Accept-Language': locale,
                  }
                }
              );
              setTryProductImg(await response.data);
            } catch (err) {
              navigate("/error", { state: { error: err.response.data.message } })
            }
          };
          getData();
        }, []);
        function handleImage() {
          document.getElementById("camera_section").style.display = "none";
          document.getElementById("image_section").style.display = "flex";
        }
        function handleCamera() {
          document.getElementById("image_section").style.display = "none";
          document.getElementById("camera_section").style.display = "flex";
        }
        function handleReset() {
          setImage([]);
        }
        function handlePlus() {
          let el = document.getElementById("dragme");
          let oldWidth = el.offsetWidth;
          console.log(oldWidth);
          document.getElementById("dragme").style.width = oldWidth + 10 + "px";
        }
        function handleMinus() {
          let el = document.getElementById("dragme");
          let oldWidth = el.offsetWidth;
          document.getElementById("dragme").style.width = oldWidth - 10 + "px";
        }
        useEffect(() => {
          var dragme = document.getElementsByClassName("dragme");
          console.log(dragme);
          for (var i = 0; i < dragme.length; i++) {
            $(dragme[i]).draggable();
          }
        }, [image]);
        return (
          <div id="characters" className="container" >
            <div className="row">
              <div className="tryproduct_section">
                <div className="tryproduct_choise_controls">
                  <button className="btn btn-primary" onClick={() => handleImage()}>{t('image')}</button>
                  <button className="btn btn-primary" onClick={() => handleCamera()}>{t('camera')}</button>
                </div>
                <div id="image_section" className="image_section">
                  <div className="tryproduct_image_contorls">
                    <label htmlFor="tryproduct_image_img" className="btn btn-
primary">{t('select_image')}</label>
                    <input id="tryproduct_image_img" className="sign_up_avatar" type="file"
onChange={(event) => setImage(event.target.files[0])} />
                    <button className="apply_filters_button" onClick={() =>
handlePlus()}>{t('increase_image_size')}</button>
                    <button className="apply_filters_button" onClick={() =>
handleMinus()}>{t('decrease_image_size')}</button>
                    <button className="apply_filters_button" onClick={() =>
handleReset()}>{t('clear')}</button>
                  </div>
```

```
                        {image.length === 0 ? <div><img className="tryproduct_image_empty"
src={require("../images/tryprofile.jpg")} /></div> : <div><img className="tryproduct_image"
src={URL.createObjectURL(image)} /></div>}
                        {image.length === 0 ? "" : <div id="dragme" className="dragme follow_image"><img
src={require("../images/products/" + tryProductImg)} /></div>}
                    </div>
                    <div id="camera_section" className="camera_section">
                      <TryProductFacemesh tryProductImg={tryProductImg} />
                    </div>
                  </div>
                </div>
              </div>
            )
          }
          export default TryProduct

          FacemeshUtils.js // Функція обчислення положення та розміру зображення окулярів у вкладці «Створити»
          // Initial
          let oldImWidth = 300;
          let oldHeight = 150;

          export const drawImage = (predictions, ctx, tryProductImg) => {
            if (predictions.length > 0) {
              predictions.forEach((prediction) => {
                const middle = prediction.annotations.midwayBetweenEyes;
                const silhouette = prediction.annotations.silhouette;
                // Middle face
                const x = middle[0][0];
                const y = middle[0][1];
                // Left face point
                const xS = silhouette[30][0];
                const yS = silhouette[30][1];
                // Right face point
                const xSr = silhouette[6][0];
                const ySr = silhouette[6][1];
                // Width
                const imWigth = (Math.sqrt(Math.pow(xSr + 25 - xS, 2) + Math.pow(ySr + 25 - yS, 2)));
                // Height (Scaling)
                let newheight = oldHeight * imWigth / oldImWidth;
                console.log("width : " + imWigth + ", height : " + newheight );
                console.log("middle [" + x + ";" + y + "], left [" + xS + ";" + yS +"]" + ", right [" + xSr + ";" + ySr + "]");
                // Image
                var img = new Image();
                img.onload = function () {
                  let imageHeight = newheight;
                  // Center point of image
                  let centerX = imWigth / 2;
                  let centerY = imageHeight / 2;
                  let newX = x - centerX;
                  let newY = y - centerY;
                  // Draw image
                  ctx.drawImage(img, newX, newY, imWigth, imageHeight);
                  // For scaling
                  oldImWidth = imWigth;
                  oldHeight = imageHeight;
                };
                img.src = require("../../images/products/" + tryProductImg.tryProductImg);
              });
            }
          };
```

UserController.java // Клас-контролер для користувачів сайту

```
package com.diploma.website.controller;

import com.diploma.website.constants.GlobalConstants;
import com.diploma.website.dto.RegistrationRequestDTO;
import com.diploma.website.dto.UserLoginRequestDTO;
import com.diploma.website.dto.UserLoginResponseDTO;
import com.diploma.website.exceptions.ErrorCode;
import com.diploma.website.exceptions.user.service.UserServiceException;
import com.diploma.website.model.CustomUserDetails;
import com.diploma.website.model.JwtAuthentication;
import com.diploma.website.model.User;
import com.diploma.website.model.UserRole;
import com.diploma.website.service.user.AuthService;
import com.diploma.website.service.ImageService;
import com.diploma.website.service.user.UserService;
import com.diploma.website.utils.JwtUtils;
import com.diploma.website.utils.UserUtils;
import com.fasterxml.jackson.databind.node.ObjectNode;
import jakarta.security.auth.message.AuthException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.Collection;
import java.util.UUID;

@RestController
public class UserController {
    private static final String USER_EMAIL_PROP = "email";
    private static final String USER_PASSWORD_PROP = "password";

    private final JwtUtils jwtUtils;

    private final UserService userService;

    private final ImageService imageService;

    private final MessageSource messageSource;

    private final AuthService authService;

    @Autowired
    public UserController(UserService userService, AuthService authService, JwtUtils jwtUtils, MessageSource
messageSource, ImageService imageService) {
        this.userService = userService;
        this.authService = authService;
        this.jwtUtils = jwtUtils;
        this.messageSource = messageSource;
        this.imageService = imageService;
    }
```

```
    @CrossOrigin(origins = "http://localhost:8080")
    @PostMapping(path = "/login")
    public ResponseEntity<Object> login(@RequestBody ObjectNode userCred) {
        String email = userCred.get(USER_EMAIL_PROP).asText();
        String password = userCred.get(USER_PASSWORD_PROP).asText();
        final UserLoginResponseDTO token = authService.login(new UserLoginRequestDTO(email, password));
        return ResponseEntity.ok(token);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @PostMapping(path = "/user")
    public ResponseEntity<UserLoginResponseDTO> getUser(int id) {
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication.getId() == id) {
            User user = userService.findUserById(id);
            CustomUserDetails g = new CustomUserDetails(user);
            final String accessToken = jwtUtils.generateAccessToken(g);
            final UserLoginResponseDTO token = new UserLoginResponseDTO(g, accessToken);
            return ResponseEntity.status(HttpStatus.OK).body(token);
        }
        throw new UserServiceException(ErrorCode.ACCESS_DENIED);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @PostMapping(path = "/reg", consumes = {MediaType.MULTIPART_FORM_DATA_VALUE})
    public ResponseEntity<String> registration(@ModelAttribute RegistrationRequestDTO
registrationRequestDTO) throws IOException {
        if (userService.findUserByEmail(registrationRequestDTO.getEmail()) != null) {
            throw new UserServiceException(ErrorCode.USER_ALREADY_EXIST);
        }
        MultipartFile img = registrationRequestDTO.getImg();
        String strongPassword;
        if (registrationRequestDTO.getPassword().equals(registrationRequestDTO.getPasswordRe())) {
            strongPassword = UserUtils.encryptPassword(registrationRequestDTO.getPassword());
        } else {
            throw new UserServiceException(ErrorCode.PASSWORDS_DO_NOT_MATCH);
        }
        String avatar = "default_user_avatar.png";
        if (img != null) {
            avatar = imageService.uploadImage(GlobalConstants.AVATAR_DIR, img);
        }
        User user = new User(registrationRequestDTO.getFirstName(), registrationRequestDTO.getLastName(),
                registrationRequestDTO.getEmail(), registrationRequestDTO.getPhoneNumber(),
                strongPassword, avatar, UserRole.USER);
        User createdUser = userService.saveUser(user);
        if (createdUser != null) {
            return                                                                                    new
ResponseEntity<>(messageSource.getMessage(GlobalConstants.REGISTRATION_COMPLETED_SUCCESSFULLY
,
                    null, LocaleContextHolder.getLocale()), HttpStatus.CREATED);
        }
        throw new UserServiceException(ErrorCode.REGISTRATION_ERROR);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/getAvatar", produces = {MediaType.IMAGE_JPEG_VALUE,
MediaType.IMAGE_PNG_VALUE})
    public @ResponseBody byte[] getUserAvatar(@RequestParam String email) {
        User user = userService.findUserByEmail(email);
        try {
            File initialFile = new File(GlobalConstants.AVATAR_DIR + "\\" + user.getAvatar());
            InputStream is = new FileInputStream(initialFile);
            BufferedImage img = ImageIO.read(is);
```

```
            ByteArrayOutputStream bao = new ByteArrayOutputStream();
            ImageIO.write(img, imageService.getExtensionByStringHandling(user.getAvatar()).get(), bao);
            return bao.toByteArray();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/getRole")
    public ResponseEntity<UserRole> getRole() throws AuthException {
        JwtAuthentication authentication = authService.getAuthInfo();
        Collection<? extends GrantedAuthority> g = authentication.getAuthorities();
        UserRole g2 = authentication.getUserRole();
        if (authentication.getUserRole() != null) {
            return new ResponseEntity<>(authentication.getUserRole(), HttpStatus.OK);
        } else {
            throw new AuthException("No role");
        }
    }
}
```

EyeglassController.java // Клас-контролер для товарів магазину

```
package com.diploma.website.controller;

import com.diploma.website.dto.EyeglassDiscountDTO;
import com.diploma.website.exceptions.ErrorCode;
import com.diploma.website.exceptions.user.service.UserServiceException;
import com.diploma.website.mapper.EyeglassMapper;
import com.diploma.website.model.*;
import com.diploma.website.service.catalog.DiscountViewService;
import com.diploma.website.service.catalog.EyeglassImageService;
import com.diploma.website.service.catalog.EyeglassPopularityViewService;
import com.diploma.website.service.catalog.EyeglassViewService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class EyeglassController {

    private final EyeglassImageService eyeglassImageService;
    ;
    private final EyeglassViewService eyeglassViewService;
    private final EyeglassPopularityViewService eyeglassPopularityViewService;
    private final DiscountViewService discountViewService;

    @Autowired
    public EyeglassController(EyeglassImageService eyeglassImageService,
                              EyeglassViewService eyeglassViewService,
                              EyeglassPopularityViewService eyeglassPopularityViewService,
                              DiscountViewService discountViewService) {
        this.eyeglassImageService = eyeglassImageService;
        this.eyeglassViewService = eyeglassViewService;
        this.discountViewService = discountViewService;
        this.eyeglassPopularityViewService = eyeglassPopularityViewService;
    }
```

```
    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/popular")
    public @ResponseBody List<EyeglassPopularityView> getPopular() {
        List<EyeglassPopularityView> list = eyeglassPopularityViewService.findAllEyeglassPopulars();
        return list;
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/catalog/male")
    public @ResponseBody Page<List<EyeglassView>> getEyeglassesMale(@RequestParam int currentPage,
@RequestParam int limit,
                                                    @RequestParam(required = false) String priceLow,
                                                    @RequestParam(required = false) String priceHigh,
                                                    @RequestParam(required = false) List<String> shape,
                                                    @RequestParam(required = false) List<String> color,
                                                    @RequestParam(required = false) List<String> material,
                                                    @RequestParam(required = false) List<String> frame,
                                                    @RequestParam(required = false) String name,
                                                    @RequestParam(required = false) String order) {
        Pageable firstPageWithTwoElements;
        if (order != null && order.equals("asc")) {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit, Sort.by("price").ascending());
        } else if (order != null && order.equals("desc")) {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit, Sort.by("price").descending());
        } else {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit);
        }
        if (priceLow != null || priceHigh != null || shape != null || color != null || material != null
                || frame != null || name != null) {
            return
eyeglassViewService.findEyeglassViewsByCategoryByShapeAndColorAndMaterialAndFrameAndNameIn(firstPageW
ithTwoElements, EyeglassCategory.MALE,
                    priceLow, priceHigh, shape, color, material, frame, name);
        }
        return          eyeglassViewService.findAllEyeglassViewsByCategoryMain(firstPageWithTwoElements,
EyeglassCategory.MALE);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/catalog/female")
    public @ResponseBody Page<List<EyeglassView>> getEyeglassesFemale(@RequestParam int currentPage,
@RequestParam int limit,
                                                    @RequestParam(required = false) String priceLow,
                                                    @RequestParam(required = false) String priceHigh,
                                                    @RequestParam(required = false) List<String> shape,
                                                    @RequestParam(required = false) List<String> color,
                                                    @RequestParam(required = false) List<String> material,
                                                    @RequestParam(required = false) List<String> frame,
                                                    @RequestParam(required = false) String name,
                                                    @RequestParam(required = false) String order) {
        Pageable firstPageWithTwoElements;
        if (order != null && order.equals("asc")) {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit, Sort.by("price").ascending());
        } else if (order != null && order.equals("desc")) {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit, Sort.by("price").descending());
        } else {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit);
        }
        if (priceLow != null || priceHigh != null || shape != null || color != null || material != null
                || frame != null || name != null) {
            return
eyeglassViewService.findEyeglassViewsByCategoryByShapeAndColorAndMaterialAndFrameAndNameIn(firstPageW
ithTwoElements, EyeglassCategory.FEMALE,
```

```
                priceLow, priceHigh, shape, color, material, frame, name);
        }
        return eyeglassViewService.findAllEyeglassViewsByCategoryMain(firstPageWithTwoElements,
EyeglassCategory.FEMALE);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/catalog/kids")
    public @ResponseBody Page<List<EyeglassView>> getEyeglassesKids(@RequestParam int currentPage,
@RequestParam int limit,
                                                    @RequestParam(required = false) String priceLow,
                                                    @RequestParam(required = false) String priceHigh,
                                                    @RequestParam(required = false) List<String> shape,
                                                    @RequestParam(required = false) List<String> color,
                                                    @RequestParam(required = false) List<String> material,
                                                    @RequestParam(required = false) List<String> frame,
                                                    @RequestParam(required = false) String name,
                                                    @RequestParam(required = false) String order) {

        Pageable firstPageWithTwoElements;
        if (order != null && order.equals("asc")) {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit, Sort.by("price").ascending());
        } else if (order != null && order.equals("desc")) {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit, Sort.by("price").descending());
        } else {
            firstPageWithTwoElements = PageRequest.of(currentPage, limit);
        }
        if (priceLow != null || priceHigh != null || shape != null || color != null || material != null
                || frame != null || name != null) {
            return
eyeglassViewService.findEyeglassViewsByCategoryByShapeAndColorAndMaterialAndFrameAndNameIn(firstPageW
ithTwoElements, EyeglassCategory.KIDS,
                priceLow, priceHigh, shape, color, material, frame, name);
        }
        return eyeglassViewService.findAllEyeglassViewsByCategoryMain(firstPageWithTwoElements,
EyeglassCategory.KIDS);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/product")
    public @ResponseBody EyeglassDiscountDTO getEyeglass(@RequestParam int id, @RequestParam String
color) {
        EyeglassView eyeglassViews = eyeglassViewService.findEyeglassViewByIdAndColor(id, color);
        DiscountView discount = discountViewService.findDiscountViewByEyeglassId(eyeglassViews.getId());
        EyeglassDiscountDTO eyeglassDiscountDTO;
        if (discount == null) {
            eyeglassDiscountDTO = EyeglassMapper.INSTANCE.toEyeglassDiscountDTO(eyeglassViews,
eyeglassViews.getPrice(), eyeglassViews.getPrice(), null, null);
        } else {
            eyeglassDiscountDTO = EyeglassMapper.INSTANCE.toEyeglassDiscountDTO(eyeglassViews,
discount.getOriginalPrice(), discount.getDiscountPrice(),
                discount.getDateOfOpening(), discount.getDateOfClosing());
        }
        return eyeglassDiscountDTO;
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/product/tryproduct")
    public @ResponseBody String getTryproduct(@RequestParam int id, @RequestParam String color) {
        EyeglassView eyeglassView = eyeglassViewService.findEyeglassViewByIdAndColor(id, color);
        if (eyeglassView == null) {
            throw new UserServiceException(ErrorCode.IMAGE_NOT_FOUND);
        }
```

```
        return eyeglassView.getTryproductImg();
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/images")
    public @ResponseBody List<EyeglassImage> getImages(@RequestParam int id, @RequestParam String
color) {
        EyeglassView eyeglassViews = eyeglassViewService.findEyeglassViewByIdAndColor(id, color);
        return
eyeglassImageService.findByEyeglassColorIdOrderByImgPositionAsc(eyeglassViews.getEyeglassColorId());
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/productAltColors")
    public @ResponseBody Iterable<EyeglassView> getEyeglassAltColors(@RequestParam int id,
@RequestParam String color) {
        return eyeglassViewService.findEyeglassViewsAltColorsById(id, color);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/getDiscounts")
    public @ResponseBody List<DiscountView> getDiscounts() {
        return discountViewService.findAllActiveDiscountViews();
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/getProductsByName")
    public @ResponseBody List<EyeglassView> getEyeglasses(@RequestParam String name) {
        return eyeglassViewService.findEyeglassViewsByName(name);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/getFirstProductsByName")
    public @ResponseBody List<EyeglassView> getFirstProductsByName(@RequestParam String name) {
        return eyeglassViewService.findEyeglassViewsFirstByName(name);
    }
}
```

OrderController.java // Клас-контролер для замовлень

```
package com.diploma.website.controller;

import com.diploma.website.dto.ConfirmOrderRequestDTO;
import com.diploma.website.model.*;
import com.diploma.website.service.catalog.EyeglassStockService;
import com.diploma.website.service.order.OrderItemService;
import com.diploma.website.service.order.OrderViewService;
import com.diploma.website.service.order.UserDetailsOrderService;
import com.diploma.website.service.order.UserOrderService;
import com.diploma.website.service.user.AuthService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.sql.Timestamp;
import java.util.List;

@RestController
public class OrderController {

    private final EyeglassStockService eyeglassStockService;
```

```
    private final AuthService authService;
    private final UserOrderService userOrderService;
    private final OrderItemService orderItemService;
    private final UserDetailsOrderService userDetailsOrderService;
    private final OrderViewService orderViewService;

    @Autowired
    public OrderController(AuthService authService, EyeglassStockService eyeglassStockService,
                           UserOrderService userOrderService, OrderItemService orderItemService,
                           UserDetailsOrderService userDetailsOrderService, OrderViewService orderViewService) {
        this.authService = authService;
        this.eyeglassStockService = eyeglassStockService;
        this.userOrderService = userOrderService;
        this.orderItemService = orderItemService;
        this.userDetailsOrderService = userDetailsOrderService;
        this.orderViewService = orderViewService;
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @PostMapping(path = "/makeOrder")
    public ResponseEntity<UserOrder> makeOrder(@RequestBody OrderPayload order) {
        int userId = order.getUserId();
        String firstName = order.getFirstName();
        String lastName = order.getLastName();
        String phoneNumber = order.getPhoneNumber();
        String email = order.getEmail();
        int paymentMethod = order.getPaymentMethod();
        Timestamp dateOfCreation = new Timestamp(System.currentTimeMillis());
        double totalPrice = order.getTotalPrice();
        List<OrderItemRow> orderItemsRow = order.getOrderItems();
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication.getId() != userId) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
        }
        UserOrder userOrder = new UserOrder(dateOfCreation, null, totalPrice, OrderStatus.REQUESTED);
        UserOrder createdUserOrder = userOrderService.saveUserOrder(userOrder);
        UserDetailsOrder userDetailsOrder = new UserDetailsOrder(createdUserOrder.getId(), userId,
                firstName, lastName, phoneNumber, email, paymentMethod);
        userDetailsOrderService.saveUserDetailsOrder(userDetailsOrder);
        for (OrderItemRow item : orderItemsRow) {
            OrderItem orderItem = new OrderItem(createdUserOrder.getId(), item.getEyeglassId(),
                    item.getColorId(), item.getAmount(), item.getPrice());
            orderItemService.saveOrderItem(orderItem);
            EyeglassStock eyeglassStock = eyeglassStockService.findEyeglassStockById(item.getEyeglassId());
            eyeglassStock.setStock(eyeglassStock.getStock() - item.getAmount());
            eyeglassStockService.saveEyeglassStock(eyeglassStock);
        }
        return ResponseEntity.status(HttpStatus.CREATED).body(createdUserOrder);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/getOrder")
    public @ResponseBody Object getOrder(@RequestParam int currentPage, @RequestParam int limit,
@RequestParam int userId) {
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication.getId() == userId) {
            Pageable firstPageWithTwoElements = PageRequest.of(currentPage, limit);
            return orderViewService.findOrderViewsByUserId(firstPageWithTwoElements, userId);
        }
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
    }

    @CrossOrigin(origins = "http://localhost:8080")
```

```
    @GetMapping(path = "/cabinet/adminorders/getRequestedOrders")
    public @ResponseBody Object getRequestedOrders(@RequestParam int currentPage, @RequestParam int
limit) {
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication == null) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
        }
        if (authentication.getUserRole() == UserRole.ADMIN) {
            Pageable firstPageWithTwoElements = PageRequest.of(currentPage, limit);
            return                    orderViewService.findOrderViewsByOrderStatus(firstPageWithTwoElements,
OrderStatus.REQUESTED.toString());
        }
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/cabinet/adminorders/getPaidOrders")
    public @ResponseBody Object getPaidOrders(@RequestParam int currentPage, @RequestParam int limit) {
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication == null) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
        }
        if (authentication.getUserRole() == UserRole.ADMIN) {
            Pageable firstPageWithTwoElements = PageRequest.of(currentPage, limit);
            return                    orderViewService.findOrderViewsByOrderStatus(firstPageWithTwoElements,
OrderStatus.PAID.toString());
        }
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @GetMapping(path = "/cabinet/adminorders/getAllOrders")
    public @ResponseBody Object getAllOrders(@RequestParam int currentPage, @RequestParam int limit) {
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication == null) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
        }
        if (authentication.getUserRole() == UserRole.ADMIN) {
            Pageable firstPageWithTwoElements = PageRequest.of(currentPage, limit);
            return orderViewService.findAllOrderViewsPag(firstPageWithTwoElements);
        }
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);
    }

    @CrossOrigin(origins = "http://localhost:8080")
    @PostMapping(path = "/confirmOrder")
    public @ResponseBody Object confirmOrder(@RequestBody ConfirmOrderRequestDTO order) {
        JwtAuthentication authentication = authService.getAuthInfo();
        if (authentication == null) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
        }
        if (authentication.getUserRole() == UserRole.ADMIN) {
            UserOrder userOrder = userOrderService.findUserOrderById(order.getOrderId());
            userOrder.setOrderStatus(OrderStatus.PAID);
            userOrderService.saveUserOrder(userOrder);
            return ResponseEntity.status(HttpStatus.OK).body(null);
        }
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна_робота_Файнштейн.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна_робота_Файнштейн.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Код_Файнштейн.docx	Код програми, який не ввійшов до пояснювальної записки
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація_Файнштейн.pptx	Презентація кваліфікаційної роботи