НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК __004.42___

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
зі спеціальності 126 «Інформаційні системи та технології»
на тему:
«Інформаційна система підтримки діяльності мережі барбершопів»

Виконав:
студент 2 курсу, групи ІК-21мп
Сарнавський Владислав Анатолійович _____

Керівник:
доцент кафедри ІСТ, к.т.н., доцент.,
Крилов Євген Володимирович _____

Рецензент:
доцент кафедри ІПІ, к.т.н., доцент.
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Сарнавський Владислав Анатолійович

1. Тема дисертації: «Інформаційна система підтримки діяльності мережі барбершопів», науковий керівник дисертації Крилов Євген Володимирович, к.т.н., доцент, затверджені наказом по університету від «07» 11 2023 р. № 5168-с.

2. Термін подання студентом дисертації «08» 01 2024 р.

3. Об’єкт дослідження: інформаційна система для діяльності мережі барбершопів (процес створення, функціональність та ефективність).

4. Вихідні дані: технічна література, теоретичні дані.

5. Перелік завдань, які потрібно розробити: аналіз предметної області, формування вимог до системи, огляд технологій, власне розробка системи, тестування, стартап-проект.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: схема функціоналу інформаційної системи, діаграма прецедентів, схема процесу аутентифікації користувачів в розробленій інформаційній системі, схема нереляційної бази даних MongoDB, схема алгоритму визначення доступного часу в розробленій інформаційній системі, схема зв’язку між

клієнтами та барбершопом у створеній інформаційній системі, схема рекламно-маркетингових засобів у розробленій інформаційній системі, структурна схема роботи створеної інформаційної системи.

7. Орієнтовний перелік публікацій: тези на III Міжнародна науково-практична конференція «Collective Thinking: Unifying Scientific Approaches in Multifaceted Research», назва статті «Інформаційна система барбершопу на основі стека технологій MERN».

9. Дата видачі завдання 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Затвердження теми	01.09.23 – 07.09.23	
2.	Постановка задачі	07.09.23 – 14.09.23	
3.	Вивчення та аналіз предметної області	14.09.23 – 01.10.23	
4.	Підбір технологій для розробки додатку	01.10.23 – 07.10.23	
5.	Власне розробка інформаційної системи	07.10.23 – 01.11.23	
6.	Тестування інформаційної системи	01.11.23 – 07.11.23	
7.	Виправлення помилок	07.11.23 – 14.11.23	
8.	Оформлення роботи	14.11.23 – 14.12.23	
9.	Подання теми на попередній захист	18.12.23	
10.	Подання теми на основний захист		

Студент _____ Владислав Сарнавський

Науковий керівник _____ Євген Крилов

РЕФЕРАТ

Інформаційна система підтримки діяльності мережі барбершопів: 131 с., 30 табл., 62 рис., 9 дод., 25 джерел.

БАРБЕРШОП, БАРБЕР, ІНФОРМАЦІЙНА СИСТЕМА, СТЕК ТЕХНОЛОГІЙ «MERN», MONGODB, EXPRESS.JS, REACT, NODE.JS, РОЗРОБКА, ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ.

Реалізація інформаційної системи для барбершопів є доцільною та актуальною задачею в наш час, адже кожен відвідував сферу краси. У даній роботі спочатку був проведений аналіз предметної області та формування вимог, що забезпечило ефективне проектування системи, виявило ризики, проблематику області.

Після цього був огляд засобів реалізації та власне розробка інформаційної системи. Ці етапи допомогли підібрати оптимальні технології та інструменти розробки, для легкого створення додатка.

Кінцеві етапи – тестування та впровадження системи. Тестування визначає, аналізує та виправляє помилки, тим самим підвищуючи якість та впевненість бездоганної роботи інформаційної системи перед релізом.

Результатом даної роботи є створена інформаційна система для мережі барбершопів на стеці технологій «MERN», де MongoDB – відповідає за гнучке зберігання даних, Express.js – реалізує логіку на стороні сервера та обмін даними, React – створює зрозумілий та швидкий інтерфейс користувача, Node.js – зв’язує всі компоненти стека разом та спрощує виконання серверного коду. Всі ці технології в сукупності забезпечують швидку, гнучку та ефективну роботу

Розроблена система може бути далі модифікована та розширена для ще більшого покращення процесу управління діяльності та обслуговування клієнтів барбершопу.

ABSTRACT

Information system for supporting barbershops chain operations: 131 p., 30 table, 62 figure, 9 appendix, 25 sources.

BARBERSHOP, BARBER, INFORMATION SYSTEM, "MERN" TECHNOLOGY STACK, MONGODB, EXPRESS.JS, REACT, NODE.JS, UTVECKLING, TESTER, IMPLEMENTERING.

Creating an information system for barbershops is an appropriate and relevant task nowadays, because everyone has visited the beauty industry. In this work, first, the analysis of the subject area and the formation of requirements was carried out, which ensured effective system design, identified risks and problems of the area.

After that, there was a review of the means of implementation and the actual development of the information system. These stages helped to choose optimal technologies and development tools for easy application creation.

The final stages are testing and implementation of the system. Testing identifies, analyzes and corrects errors, thereby increasing the quality and confidence of the perfect operation of the information system before the release.

The result of this work is the creation of an information system for a network of barbershops on the MERN technology stack, where MongoDB is responsible for flexible data storage, Express.js implements server-side logic and data exchange, React creates a clear and fast user interface, Node. js – ties all components of the stack together and simplifies the execution of server code. All these technologies together provide fast, flexible and efficient work

The developed system can be further modified and expanded to further improve the process of business management and customer service of the barbershop.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

BDD – Behavior-Driven Development

CLI – Command Line Interface

CSS – Cascading Style Sheets

DB – Database

DOM – Document Object Model

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IDE – Integrated Development Environment

JS – JavaScript

JSON – JavaScript Object Notation

JPA – Java Persistence API

JWT – JSON Web Token

NPM – Node Packet Manager

OS – Operating System

OOP – Object-oriented Programming

SQL – Structured Query Language

TDD – Test-Driven Development

UI – User Interface

URL – Uniform Resource Locator

VCS – Version Control System

VM – Virtual Machine

ІС – інформаційна система

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ВСТУП

Сучасний світ бурхливо розвивається та адаптується до нових реалій життя, особливо в інформаційному напрямі. В наші дні інформаційні технології мають великий вплив та використовуються в усіх процесах життя людини, полегшуючи й автоматизуючи їх.

Інформаційні технології – це методи та засоби для створення, зберігання, обробки та передачі різноманітної інформації. Тобто, це найголовніший спосіб комунікації, який є загальнодоступним та розповсюдженим. Через інтернет кожен може викликати собі таксі, придбати товари, зробити запис до лікаря, орендувати номер в готелі, знайти репетитора з іноземних мов, роблячи це просто зі свого девайсу, який підключений до Всесвітньої павутини.

Тому, в наш час розробляється багато інформаційних систем, які допомагають реалізувати, покращувати ці повсякденні процеси та послуги, використовуючи сучасні технології, нові інструменти розробки, ефективні алгоритми тощо.

Особливо, інформаційні системи будуть корисні у сфері краси, а саме – мережі барбершопів. Кожен хоче бути доглянутим та мати гарний вигляд, але є проблеми щодо вибору барбешопу, процедури, зачіски, спеціалізованих товарів для догляду, часу прийому тощо. Тобто, ця область має безліч процесів, які можна автоматизувати й покращити. Тому кожен барбешоп не може обійтись без впровадження інформаційної системи.

Інформаційна система полегшить управління бізнесом: планування графіків персоналу, аналіз ефективності використання ресурсів, відстеження та покращення послуг, надсилання спеціальних рекламних пропозицій клієнтам, фінансову звітність, зворотний зв’язок з клієнтами тощо. Тобто, від функціонування системи буде залежати внутрішній процес роботи барбершопу. А зовнішній вигляд та широкий функціонал системи привабить клієнтів, що в своєю чергою підвищить прибуток.

Отже, актуальністю цієї роботи є впровадження інформаційної системи в мережу барбершопів дозволить: підвищити ефективність обслуговування клієнтів, змінити та покращити підхід управління, оптимізувати графіки роботи персоналу та, найголовніше, підвищити конкурентоспроможність.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В даному розділі міститься інформація про мету, постановка задачі та опис проблеми інформаційної системи для мережі барбершопів на стеці технологій «MERN», також загальне поняття інформаційної системи, їх властивості, види та співвідношення з інформаційними технологіями.

1.1 Мета та постановка задачі

Мета роботи являє собою створення «респонсивної» інформаційної системи, яка підтримує, оптимізує, покращує та автоматизує внутрішню роботу барбершопу.

Основною задачею роботи є розрекламування мережі барбершопів, автоматизація онлайн-запису на послугу, придбання онлайн косметичних товарів тощо. Це все створить нову базу клієнтів, тим самим підвищить прибуток та конкурентоспроможність мережі барбершопів.

1.2 Проблематика предметної області

З кожним днем все більше і більше зростає сфера інформаційних технологій, також збільшуються й користувачі цих технологій. Тому, завдяки цьому зростанню, збільшується попит і на створення спеціальних інформаційних систем, які допомагають оптимізувати та автоматизувати процес роботи,

Проблематика для потенційного клієнта полягає у потрібності послуг від барбершопу (це може бути запис на візит до барбера, купівля певних спеціалізованих товарів для догляду за волоссям тощо). Тому щоб отримати послугу, він повинен використати інформаційну систему, яка забезпечить швидкість та ефективність у виконанні послуг барбершопом.

Інформаційна система має мати широкий функціонал, швидкість роботи, мінімалістичний дизайн, надійність, безпеку, підтримку, легку інтеграцію з іншими

сервісами, гнучкість та інтуїтивно-зрозумілий інтерфейс для легкої взаємодії з користувачем.

1.3 Поняття інформаційної системи, її цілі та властивості

Інформаційна система – це сукупність інформаційних, програмних, математичних та інших засобів, завдяки яким, інформація може створюватись, зберігатись, оброблятись та передаватись для певних задач [1]. Узагальнена схема інформаційної системи продемонстрована на рис. 1.1.

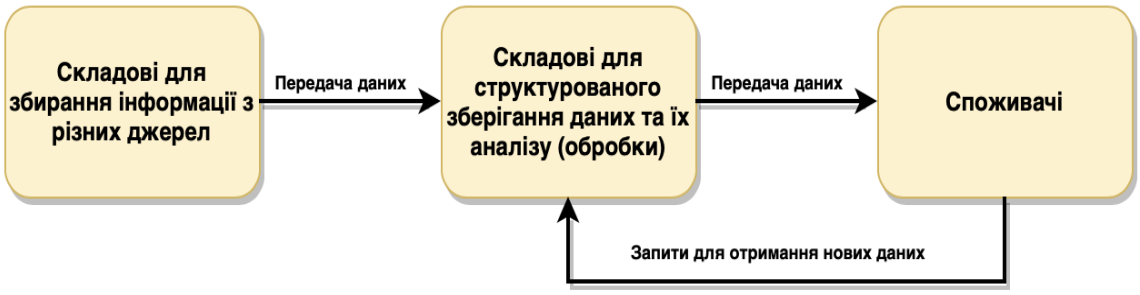

Рисунок 1.1 – Узагальнена схема інформаційної системи

Цілі інформаційних систем:

– обробка даних – ефективно аналізувати та обробляти великий масив даних для прийняття заданого рішення;

– забезпечення доступу до інформації – забезпечення доступу до даних користувачів, клієнтів, статистики в залежності від задачі;

– підвищення рівня інновацій – використання нових технологій, алгоритмів та ідей;

– захист конфіденційних даних – покращення та вдосконалення захисних механізмів для зберігання конфіденційних даних від несанкціонованого доступу;

– підвищення високої якості обслуговування – забезпечення доступності та зменшення часу відгуку.

Інформаційні системи мають властивості, які забезпечують оптимальне та ефективне її функціонування. Загальні властивості продемонстровані на рис 1.2.

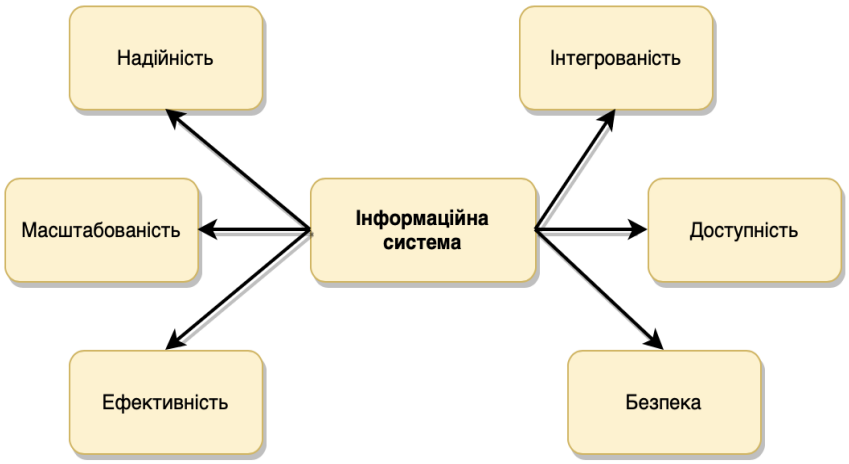

Рисунок 1.2 – Загальні властивості інформаційної системи

Властивості інформаційних систем:

– надійність – наскільки система коректно працює, видаючи правильний результат;

– масштабованість – наскільки система може розширюватись у порівнянні до зростання користувачів або обсягів даних;

– інтегрованість – можливість системи мати зв’язки з іншими системами для обміну даними, правил, стандартів;

– простота використання – система має бути інтуїтивно простою та зрозумілою для користувача, щоб полегшити користування нею;

– доступність – система має працювати увесь час без збоїв;

– ефективність – оптимальне використання ресурсів для певної задачі системи;

– безпека – система має надавати захист конференційній інформації користувача від несанкціонованого доступу;

– адаптація – система має коректно відображатись на всіх пристроях та їх ширині екранів;

– швидкодія – визначається можливістю системи швидко та ефективно виконувати задачі (відповідь на запити або обробка інформації);

– модульність – система повинна бути розбита на окремі та незалежні модулі, які ефективно будуть взаємодіяти між собою.

1.4 Висновки до розділу 1

В даному розділі, був проведений аналіз та огляд предметної області для інформаційної системи, який виявив важливі аспекти та вимоги, які потрібно застосувати при реалізації інформаційної системи для мережі барбершопів.

Була врахована потреба в ефективності, надійності, гнучкості, функціональності, масштабованості та безпеки системи, що допоможе у проектуванні та створенні системи.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Огляд та аналіз існуючих рішень інформаційних систем сфери краси

Веб-застосунки для салонів краси є важливими в сучасному світі beauty-індустрії, адже вони є обличчям компаній. Завдяки швидкому й потужному розвитку інформаційних технологій, все більше і більше людей обирають заклад салонів по онлайн-сервісах. Тому огляд та аналіз існуючих рішень веб-застосунків допоможе виділити потрібний функціонал, покаже особливості та сильні сторони таких платформ.

Основною перевагою цих веб-застосунків є можливістю швидко й легко зробити онлайн-запис до потрібного барбера. Користувачу потрібно вказати свою персональну інформацію, вибрати дату та час.

Також, веб-застосунки повинні демонструвати всю інформацію про барбершоп: актуальні послуги та їх вартість, доступний асортимент товарів, новини про заклад і в сфері краси, опис барберів, галерею, контактну інформацію тощо. Та й сам веб-додаток повинен бути візуально красивим.

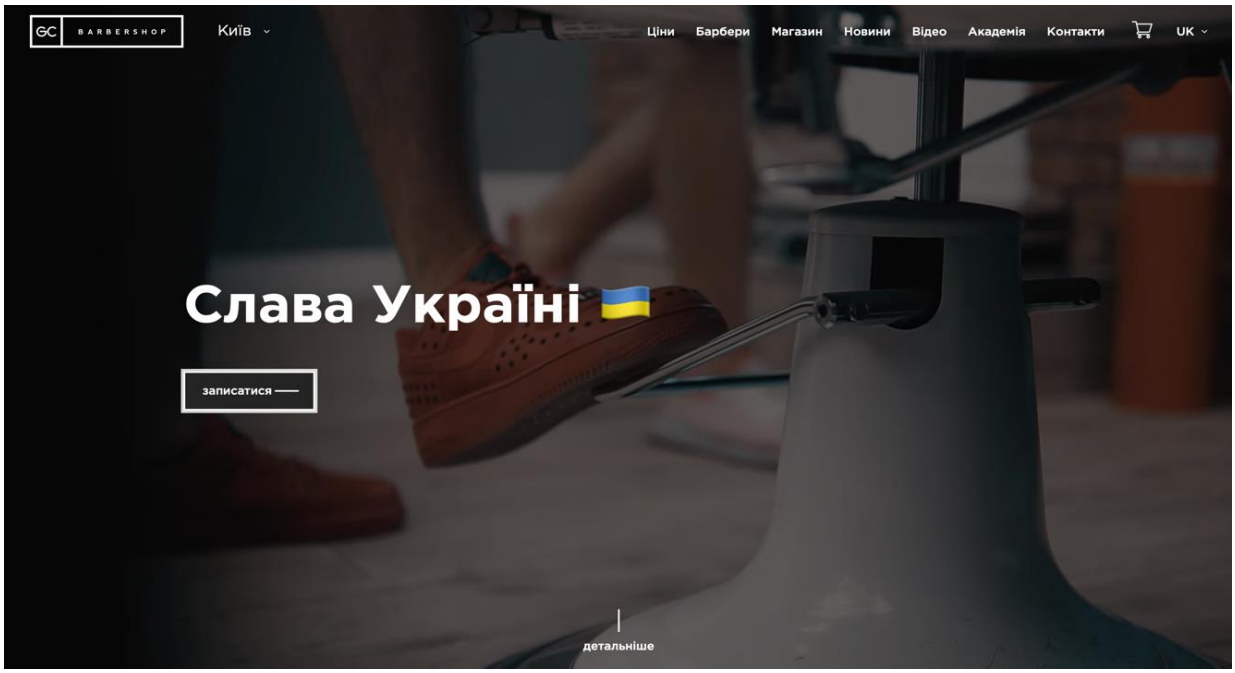

Рисунок 2.1 – Веб-додаток «GC Barbershop»

«GC Barbershop» є найпопулярнішим веб-додатком серед компаній сфери краси. Він привертає увагу своїм мінімалістичним дизайном та анімаціями. За допомогою нього, клієнти можуть легко зробити онлайн-запис. Окрім того, ви можете придбати спеціалізовані товари за доглядом обличчя, волосся та тіла, які відсортовані за призначенням. Також присутня інформація про барберів, новини, послуги. Продемонстрований веб-додаток «GC Barbershop» можна побачити на рисунку 2.1.

Для форми онлайн-запису використовується стороння технологічна база «Altegio». Це сервіс онлайн-запису для закладів та платформа, яка автоматизує сферу послуг. Тобто, «Altegio» – це цифрова екосистема та технологія ведення автоматизованого бізнесу.

Переваги:

– мінімалістичний дизайн;

– зручний та простий інтерфейс;

– детальна інформація про послуги;

– галерея внутрішнього процесу роботи;

– новини роботи барбершопу та тенденцій у сфері краси;

– можливість придбання спеціалізованих товарів;

– плавні анімації;

– наявність вкладки «Академія», де показана інформація про базовий курс;

– зміна мов веб-сайту;

– адаптивність (мобільно-дружність) веб-сайту.

Недоліки:

– немає авторизації (також і для барбера або менеджера);

– мало інформації на основній (головній) сторінці;

– залежність від сторонніх платформ автоматизації

– повільне завантаження при вході на основній (головній) сторінці через транслювання відео та повільне завантаження другорядних сторінок;

– немає зворотного зв’язку (форми для заповнення відгуку чи пропозиції).

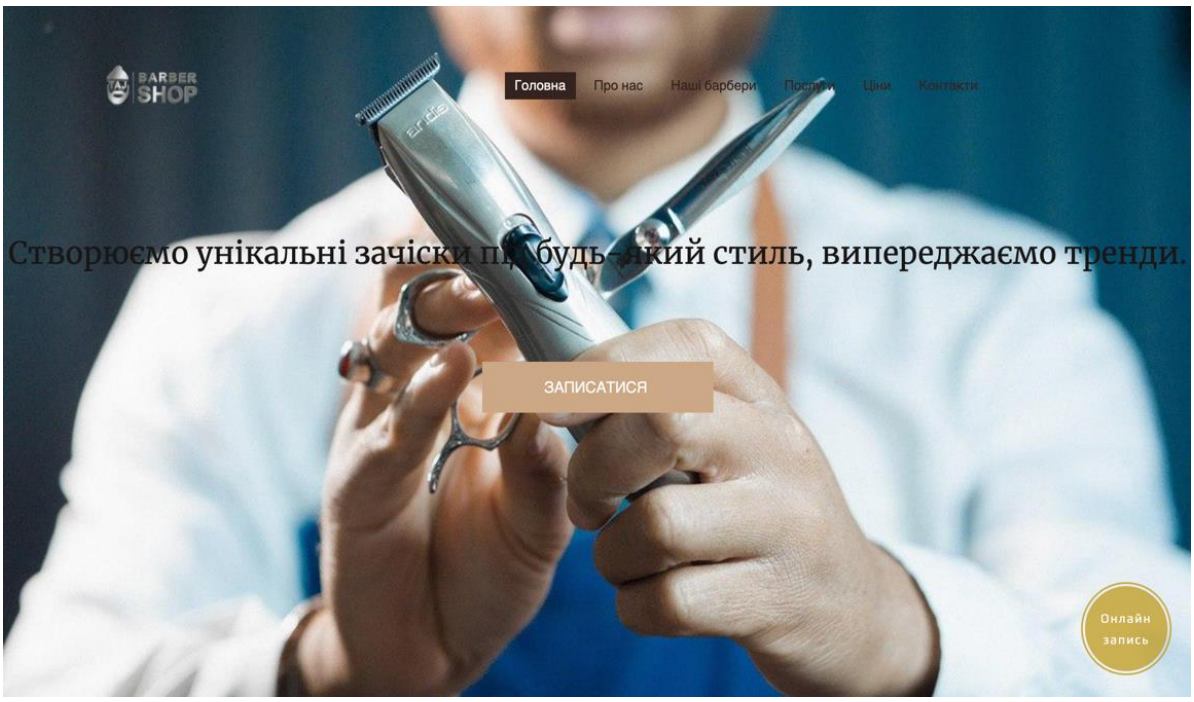

Рисунок 2.2 – Веб-додаток «TAJ Barbershop»

«TAJ Barbershop» – є веб-додатком мережі «TAJ» барбешопів («Taj з таджицького – корона). Має більш наповнену основну сторінку з детальною інформацією для клієнта. Також присутня фотогалерея з моментами роботи та готових зачісок в даних барбершопах.

Дана мережа також використовує сторонню технологічну базу «Altegio», що допомагає автоматизувати процес онлайн-запису на послуги.

Переваги:

– наповненість основної сторінки;

– детальна фотогалерея процесу роботи та зачісок;

– зміна мов веб-сайту.

Недоліки:

– залежність від сторонніх платформ автоматизації;

– простий дизайн вебдодатку;

– конфліктуюча колірна гамма;

– відсутність зворотного зв’язок для клієнта, тобто форма для заповнення відгуку чи пропозиції на веб-сайті;

– відсутність авторизації клієнтів або персоналу мережі;

– відсутність онлайн-магазину та можливості придбати спеціалізовані товари за доглядом волосся.

Як підсумок, огляд та аналіз наведених веб-додатків допоміг визначити основні функції, якими повинна володіти сучасна інформаційна система для мережі барбершопів на сьогодні. Це допоможе надалі розробити ефективну та оптимальну систему з урахуванням цих переваг та недоліків.

2.2 Функціональні вимоги системи

Функціональні вимоги – визначають як система спрапрацьовує та реагує на конкретні запити чи події [2]. Якщо порівнювати з функіоналом програми (загальним переліком можливостей інформаційної системи), то функціональні вимоги конкретизують як система має поводитись в заданих сценаріях (табл. 2.1).

Таблиця 2.1 – Опис функціональних вимог системи

№	Вимога	Опис функціональної вимоги
1.	Адміністративний доступ	В системі мають бути різні рівні доступу для менеджера, барберів та клієнтів для забезпечення безпеки в системі.
2.	Фінансовий облік	Здатність системи фіксувати вартість кожної покупки або послуги й зберігати це в історії покупок та послуг.
3.	Фотогалерея робіт	Система має включати в себе фотогалерею з внутрішнього процесу роботи та для представлення барберів.
4.	Моніторинг змін	Усі зміни в системі мають бути коректно відпрацьовані та збережені.
5.	Програма лояльності	Повинна підтримуватись програма лояльності для постійних клієнтів (надсилання знижок або різних промокодів).

№	Вимога	Опис функціональної вимоги
6.	Керування клієнтами	Система повинна містити в собі всю інформацію про клієнта, включаючи історію покупок та послуг.
7.	Повідомлення	Система має автоматично надсилати клієнтам всю інформацію про зроблений запис чи покупку на електронну пошту клієнта.
8.	Віддалений доступ	Система має підтримувати віддалений доступ для менеджера та барберів з будь-якого місця та який мати бути безпечним.

2.3 Нефункціональні вимоги системи

Нефункціональні вимоги – це вимоги, які інформаційна система барбершопу має забезпечувати й виконувати [2]. Тобто, ці вимоги визначають ефективність та якість виконання роботи системою (табл. 2.2).

Таблиця 2.2 – Опис нефункціональних вимог системи

№	Вимога	Опис нефункціональної вимоги
1.	Стабільність	Система здатна функціонувати надійно та без збоїв, перебувати в рівновазі протягом тривалого проміжку часу в ефективному режимі.
2.	Надійність	Система має бути готова до появи помилок на всіх етапах роботи. Якщо помилка зафіксована – або усунути самостійно, або видати змістовне повідомлення про причину помилки.
3.	Гнучкість	Система може змінювати свій функціонал без великих зусиль та залишатись ефективною.

№	Вимога	Опис нефункціональної вимоги
4.	Конфіденційність	Вся інформація про клієнта, барбера та менеджера повинна бути захищена від несанкціонованого доступу або хакерських атак.
5.	Інтегрованість	Система повинна легко, оптимально й ефективно взаємодіяти з іншими системами чи їх компонентами, адаптуватись до цих змін без помилок або збоїв системи.
6.	Масштабованість	Система може бути модифікована та покращена в майбутньому. Також повинна збільшувати навантаження, якщо збільшується обсяг даних чи клієнтів барбершопу.
7.	Продуктивність	Система повинна швидко та ефективно обробляти запити клієнтів та персоналу барбершопу. При цьому витрачати мінімум ресурсів для процесів.
8.	Адаптованість	Система має легко простосовуватись до нових умов або змін без збоїв та помилок.
9.	Зручність	Інформаційна система повинна мати сучасний дизайн та інтуїтивно зрозумілий інтерфейс для клієнта або персоналу барбершопу.
10.	Модульність	Система має бути модульною, де кожен модуль має відповідати за свою роботу.
11.	Переносимість	Можливість системи працювати на різному обладнанні або операційній системі.
12.	Точність	Вихідна дані для системи мають бути правильними, а після цього повинні правильно обробитись системою (надсилання логіну та паролю на перевірку, персональна інформація про клієнта тощо).

2.4 Висновки до розділу 2

В даному розділі спочатку було проведено огляд та аналіз популярних на сьогодні інформаційних систем інших компаній барбершопів та визначено їх переваги та недоліки.

Дана інформація стала основою в закладанні функціонала, функціональних та нефункціональних вимог до створення інформаційні системи для мережі барбершопів. Це допоможе проектуванню та створенню надійної, ефективної та зручної інформаційної системи, де клієнти зможуть швидко й легко отримати послуги в барбершопі.

3 ОГЛЯД СУЧАСНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

3.1 Огляд середовищ розробки

Інтегроване середовище розробки (IDE) – це набір інструментів, необхідних розробникам для створення, редагування, тестування та налагодження в одному інтерфейсі [3]. Вони зручні, збільшують продуктивність розробників, економлять їх час та спрощують розробку програм, при цьому вихідна програма є ефективною.

Основні компоненти IDE:

– текстовий редактор – потрібен власне для написання та редагування вихідного коду. Також підсвічує синтаксис та має автодоповнення коду, що робить розробку простішою;

– компілятор/інтерпретатор – відповідає за перетворення вихідного коду в виконуваний код;

– відлагожувач – допомагає відстежувати виконання програми завдяки точкам зупинки;

– плагіни – допомагають розширити функціонал середовища або додати підтримку різних мов програмування;

– автодоповнення – показує альтернативні варіанти закінчення коду або ж показує правильний варіант неправильного виконання;

– система керування версіями – відстежує всі зміни та веде історії версій (на випадок спільної розробки з іншим розробником);

– інструменти побудови та управління проектом – інструменти для управління конфігурації, залежностями та компіляції коду;

– інтеграція з зовнішніми інструментами – можливість взаємодіяти з іншими технологіями тестування, системи збирання.

В наш час, напрям розробки середовищ розробки швидко та якісно розвивається, додаються нові функції та зміни, які допомагають та полегшують роботу розробникам.

WebStorm – популярне середовище розробки, розроблене відомою компанією JetBrains, зі спеціалізацією різних технологій веброзробки. Має інтуїтивно зрозумілий інтерфейс, швидкодію, багатофункціональність, що разом створює високу продуктивність.

Переваги:

– функціональність – інтерфейс підтримує багато різних технологій та інструментів для ефективної розробки вебзастосунків;

– Git – інтеграція з системою керування версіями (VCS), що дозволяє простіше та швидше взаємодіяти з іншими учасниками розробки [4];

– тестування – вбудована підтримка тестів;

– інструменти – підтримка багатьох фреймворків та мов;

– автодоповнення – зручне та ефективне автодоповнення коду;

– відлагодження – функціональні інструменти для відлагодження коду;

– Live edit – функція для своєчасного відображення змін у системі.

Недоліки:

– платна підписка – це є найголовніший недолік, адже для студентів не найкращий варіант;

– навантаження – є потужним продуктом, тому пристрій має мати сучасні технічні характеристики;

– інтерфейс – є багатофункціональний інтерфейс, але початковому користувачу буде важко зорієнтуватись в ньому.

Visual Studio Code (VSCode) – відкрите середовище розробки, яке характеризується легким використанням та широким колом розробників, потужними інструментами, розширеним функціоналом. Розроблений компанією Microsoft.

Переваги:

– текстовий редактор – має потужний текстовий редактор з функціями настроювання під проект;

– розширення – має безліч сучасних та популярних на сьогодні розширень для продуктивного створення програм;

– робота з багатьма мовами – може підтримувати такі мови програмування: JavaScript, Java, Python, TypeScript тощо;

– Live share – допомагає спільно працювати розробникам над проектом в режимі реального часу;

– тестування – присутня інтегрована система тестування для забезпечення правильної компіляції коду;

– оновлення – постійне оновлення, яке покращує функціональність та ефективність середовища розробки;

– пошук та заміна – продвинуті інструменти пошуку

– Git – інтеграція з системою керування версіями (VCS), що дозволяє простіше та швидше взаємодіяти з іншими учасниками розробки;

– безоплатна підписка – це є найголовніша перевага, адже для студентів це найкращий варіант;

– низьке споживання ресурсів – IDE має маленький розмір й розробник сам вибирає які розширення йому потрібні.

Недоліки:

– відсутність для великих проектів вбудованих розширень та інструментів – через те, що середовище розробки має низький розмір даних, потрібно додатково довго шукати та встановлювати потрібні розширення (особливо якщо великий проект);

– проблеми з автоматичним оновленням – іноді оновлення можуть видати помилки та конфлікти через сторонні розширення, які вже не будуть підтримуватись цим середовищем розробки й приходиться шукати аналоги в журналі розширень;

– не всі функції вбудовані автоматично – деякі додаткові функції потрібно вручну активувати в налаштуваннях середовища.

Sublime text – також популярне середовище розробки. Відзначається простим інтерфейсом, гнучкістю та швидкістю.

Переваги:

– система вкладок – дозволяє швидко переходити між вкладками файлів;

– швидкодія – швидко запускається та працює (особливо на потужних та великих файлах або проектах);

– розширення – також наявний великий вибір різних розширень для ефективної розробки.

Недоліки:

– платна підписка – це є найголовніший недолік, адже для студентів не найкращий варіант;

– відсутність інтеграції з Git – немає вбудованого функціонала для взаємодії з системою керування версіями (VCS);

– відсутність вбудованого відлагожувача – немає функціональних інструментів для відлагодження коду, що може призвести до появи помилок і зменшення продуктивності розробника.

Як висновок, різноманіття середовищ розробки є важливим ресурсом розробників, адже у швидкозмінному світі програмування – продуктивність та ефективність є головними ознаками для створення програм.

3.2 Огляд технологій для створення клієнтської частини

Розробка клієнтської сторони інформаційної системи є тривалим та важким процесом, тому використовують різні технології та інструменти, щоб максимально спростити систему, зробити її ефективною.

Клієнтська частина є дуже важливою частиною для системи, адже через неї користувач взаємодіє із системою. Тому вона повинна бути легкою в розумінні інтерфейсу, мати чудовий (мінімалістичний) дизайн, масштабованою для подальших покращень, швидкою тощо. З цією метою були створені спеціальні технології та інструменти, які допомагають цьому.

React – найпопулярніша та найефективніша бібліотека для створення клієнтського інтерфейсу, створена компанією Facebook.

Основна логіка бібліотеки полягає у розбивання на компоненти, тим самим створюючи модульний код, який стає кращим та простішим для розуміння. Також

ці компоненти можливо перевикористовувати або вкладати один в одного, роблячи ієрархію.

React має віртуальний DOM – це концепція (дерево), яка використовується для покращення та оптимізації процесу оновлення відображення сторінки. Вона допомагає зменшити кількість разів звертання до справжнього DOM, тим самим збільшуючи швидкодію, продуктивність та ефективність [5].

Переваги бібліотеки React:

– компоненти – можна створювати власні компоненти та використовувати їх в програмі, тим самим оптимізуючи його;

– JSX – це є розширенням мови програмування JavaScript, яке полегшує роботу з UI-елементами;

– популярність – майже 60% вебдодатків створені даною бібліотекою. Має широку спільноту розробників;

– React Native – знаючи одну універсальну бібліотеку React, також можливо створювати мобільні додатки завдяки React Native, адже вся логіка побудови компонентів й переваги ті ж (синтаксис може бути трохи іншим);

– віртуальний DOM – як вище було зазначено, використовується для оптимізації оновлення сторінок;

– управління станами – допомагає зберігати, відслідковувати та передавати дані згідно задачі;

– реактивне оновлення – коли React бачить зміни в компоненті швидко реагує на ці зміни та оновлює сторінку;

– сумісність – бібліотека має велику сумісність та взаємодію з іншими технологіями (Redux, GraphQL тощо);

– хуки – створюють можливість взаємодії функціональних компонентів та контекстом, ефектом і станом. Ця взаємодія робить компоненти більш функціональними, продуктивними та ефективними.

Недоліки бібліотеки React:

– складність – React складна бібліотека зі своєю унікальною логікою та концепцією;

– залежність від JavaScript – не можна використати інші мови програмування для створення компонентів;

– синтаксичний цукор – JSX, класи в JS та інші елементи, які є синтаксичним цукром, це вимагає час розробника для звикання до них і розумінні як воно працює;

– залежність від інших інструментів – використання React часто відбувається з багатьма іншими бібліотеками, які мають кращий функціонал, ніж вбудовані функції в React і це створює залежність.

Angular – фреймворк для створення вебдодатків, розроблений Google. Він є саме структурним фреймворком, де основними принципами є використання компонентів (як у вище згаданій бібліотеці React) та двостороннє зв'язування даних, яке оптимізує взаємодію користувача та вебдодатком. Фреймворк дозволяє реалізувати розробникам створювати вебдодатки завдяки принципу єдності та структурованої архітектури [6].

Переваги фреймворку Angular:

– TypeScript – ця мова програмування є рекомендованою для розробки та має такі переваги: статичну типізацію, справжні класи, інтерфейси тощо;

– двостороннє зв'язування даних – автоматичне оновлення змін у стані та даних системи;

– ін'єкції – в Angular є залежність від ін'єкцій для ефективного управління залежностями;

– роутинг – є вбудований механізм маршрутизації;

– HHTP-модуль – для взаємодії HHTP-запитів із серверною частиною інформаційної системи (тобто, відправлення або отримання даних потрібних даних);

– форми – Angular має вбудовані інструменти для роботи з формами, їх валідацію для точності даних;

– RxJS – тісно використовується бібліотека RxJS для асинхронного та реактивного програмування;

– компоненти – додаток розбивається на компоненти, які не залежать один від одного, для структурованості системи.

Недоліки фреймворку Angular:

– час розгортання – потрібно більше часу для розгортання та першого завантаження у порівнянні з іншими бібліотеками чи фреймворками;

– ресурсомісткість – створює велике навантаження, вимагає Angular CLI для розгортання системи;

– стандарти – має високі стандарти і вимоги до структури системи;

– синтаксис шаблонів – є нестандартний синтаксис шаблонів, який є відмінним від HTML та JSX в React;

– бандли – бандли мають великий розмір.

Vue.js (3 версія) – це оновлена версія фреймворку, використовується також для створення вебдодатків. Головними особливостями є система композицій для покращення станів та логіки та ефективному використанні пам’яті [7].

Переваги фреймворку Vue.js:

– легкість освоєння – має легкий інтерфейс та логіку роботи;

– прогресивність – це прогресивний фреймворк, в якому можна використовувати те в фреймворку, що потрібно саме для індивідуальної розробки системи;

– двостороннє зв’язування даних – автоматичне оновлення змін у стані та даних системи;

– компоненти – архітектура фреймворку заточена на незалежності компонентів системи;

– реактивне оновлення – коли React бачить зміни в компоненті швидко реагує на ці зміни та оновлює сторінку;

– популярність – наявність широкої спільноти розробників;

– директиви – є вбудовані директиви (v-if, v-for тощо), які полегшують та оптимізують роботу з DOM.

Недоліки фреймворку Vue.js:

– екосистема – є менш розвинена у порівнянні з іншими фреймворками чи бібліотеками;

– мала спільнота – є малою, що впливає на документацію, відповідей на запитання у групах, плагінів тощо.

– не для великих проектів – адже менше вбудованого функціоналу та менше плагінів;

– менша сумісність – присутня менша сумісність до інших бібліотек чи фреймворків.

3.3 Огляд середовищ для розробки серверної частини

Для створення сучасної інформаційної системи, також відіграє велику роль і серверна частина. Тому також потрібно розглянути всі середовища для її розробки та оцінити всі можливості, недоліки та переваги для обрання найбільш оптимального та оптимального варіанту для майбутньої інформаційної системи мережі барбешопів.

Node.js – є вільним та відкритим середовищем для розробки серверної частини, який заснований на V8 від Google (V8 в свою чергу був створений для веб-браузера Google Chrome). Головна ідея Node.js це використання JavaScript та забезпеченні можоивості виконання коду на стороні серверу, а не тільки на клієнтській (оптимізується навантаження) [8].

Також його застосовують для створення API-додатків, real-time додатків мікросервісна архітектура тощо.

Переваги середовищ для розробки Node.js:

– асинхронність та неблокуючий ввід та вивід – допомагає приймати та обробляти багато запитів одночасно без блокування інших операцій, використовуючи асинхронну модель (особливо корисна перевага для real-time додатків);

– ефективність – використовує мало пам’яті та ресурсів для виконання операцій, та оптимізує результат;

– модульність – додаток розбивається на компоненти, які не залежать один від одного, для структурованості системи;

– популярність – має широку спільноту розробників;

– масштабованість – середовище відкрите для масштабування та модифікацій;

– JavaScript – спільна мова програмування, яка часто використовується на фронтенді (при побудові клієнтської частини);

– швидкість – через те, що середовище засноване на V8, має високу продуктивність та швидкість виконання коду;

– NPM – підтримує пакетний менеджер, що дозволяє швидко встановлювати залежності та оптимізувати ними систему;

– крос-платформеність – середовище може запускатись на різних операційних системах (OS).

Недоліки середовищ для розробки Node.js:

– однопоточність – це може бути недоліком при обчислювальній роботі;

– оновлення – іноді оновлення призводять до проблем із сумісністю системи;

– бібліотеки і асинхронність – не всі бібліотеки або модулі середовища можуть підтримувати асинхронність.

Deno – середовище виконання JavaScipt на сервері, яке було розроблено Райаном Дальєм (один із засновників Node.js). На відміну від інших середовищ має кращий рівень безпеки (доступ до ресурсів по правам). Також присутня модульність та вбудованої системи пакетів [9].

Переваги середовищ для розробки Deno:

– безпека – є найголовнішою ознакою, адже вимагає явного надання прав розробником;

– асинхронність та неблокуючий ввід та вивід – допомагає приймати та обробляти багато запитів одночасно без блокування інших операцій, використовуючи асинхронну модель (особливо корисно для real-time додатків);

– тестування – наявні вбудовані інструменти тестування;

– модульність – додаток розбивається на компоненти, які не залежать один від одного, для структурованості системи;

– обробка помилок – покращена обробка помилок допомагає покращувати якість коду;

– відлагожувач – допомагає відстежувати виконання програми завдяки точкам зупинки;

– популярність – має широку спільноту розробників;

– крос-платформеність – середовище може запускатись на різних операційних системах (OS).

Недоліки середовищ для розробки Deno:

– відсутність NPM – використовується власна система пакетів, що має менший функціонал у порівнянні з NPM менеджером пакетів;

– новизна – є відносно новим середовищем, тому має менше рішень та спільноту розробників. Також може мати не детальну документацію;

– відсутність гарячого перезавантаження – є відсутній цей механізм, який допомагає швидко оновлювати код або його перезапускати;

– відсутність Native бібліотек – менше можливостей для використання native бібліотек, що впливає на ефективність коду;

– не всі модулі Node.js підтримується – через суттєві відмінності середовищ не всі модулі підтримуються, що також впливає на ефективність коду.

GraalVM – це інноваційне виконавче середовище, яке розробило Oracle Labs. Підтримує різні мови програмування, а саме: JavaScript, Java, Ruby й дозволяє створювати серверну логіку для вебдодатків, використовуючи різні технології та інструменти [10].

Він орієнтований не тільки для створення серверної частини, а більше для універсальних сценаріїв. Наприклад, може застосовуватись для розробки десктопних додатків, дослідження, оптимізації тощо.

Переваги середовищ для розробки GraalVM:

– продуктивність – має високу оптимізацію та продуктивність під час виконання коду;

– Native Image Compilation – здатність створювати нативні виконувані файли, це оптимізує код, пришвидшує виконання коду та зменшує витрати пам’яті;

– Polyglot Integration – можливе легке створення поліглотних програм, адже можна застосовувати різні мови програмування.

– багатофункціональність – можна створювати і серверні застосунки, оптимізацію чи виконання коду різних мов програмування;

– інструменти – володіє багатьма сучасними інструментами для створення ефективного та продуктивного коду.

Недоліки середовищ для розробки GraalVM:

– ресурсомісткість – особливо при обробці зображень;

– технології – у порівнянні з іншими середовищами, не всі бібліотеки та фреймворки можна застосовувати;

– оновлення – іноді оновлення призводять до проблем із сумісністю системи;

– новизна – є відносно новим середовищем, тому має менше рішень та спільноту розробників. Також може мати не детальну документацію;

– відсутність вбудованого відлагожувача – немає функціональних інструментів для відлагодження коду, що може призвести до появи помилок і зменшення продуктивності розробника.

3.4 Огляд технологій для розробки бази даних, їх види та особливості

Бази даних є важливою частиною інформаційної системи, адже виконує функцію зберігання, доступу, управління та редагування великих масивів даних. У світі веб-застосунків, мобільних застосунків, розподілених систем, розуміння принципів роботи баз даних є основним елементом для розробників, аналітиків, інженерів.

База даних – це структурна колекція, яка зберігає та організовує інформацію і надає доступ до них. Вони використовуються для забезпечення надійності, узгодженості, цілісності та консистентності інформації. Також використовується і для покращення швидкодії та ефективності взаємодії з даними.

Тому щоб стати ефективним, досвідченим розробником в наш час, потрібно розуміти базові процеси, принципи роботи та практики ефективної взаємодії з базами даних.

Системи управління базами даних (СУБД) – це ПЗ, що відповідає за структуроване зберігання, організацію даних та взаємодію з даними. Тобто, основна ідея, що бази даних – це набір даних, а СУБД – це ПЗ, яке реалізує інструменти для роботи з цими даними [11].

Бази даних поділяються на різні види, кожен з яких заточений на свої конкретні задачі та цілі та відповідає власним вимогам. Бази даних можуть бути: реляційними, нереляційними, об'єктно-орієнтованими, графовими, ключ-значення тощо [12]. Їх опис можна продивитись у табл. 3.1-3.4.

Таблиця 3.1 – Опис особливостей реляційних баз даних

№	Особливіть	Опис особливостей графових баз даних
1.	Таблиці	Дані групуються та зберігаються в таблицях з назвами, де рядки представляють записи (кортежи), а стовпці – поля.
2.	Ключі	Кожна таблиця має свій ключ, який є неповторним та застосовується для зав’язків між таблицями.
3.	Зв’язки	Відношення між таблицями. Є різні типи: багато до одного, один до одного та один до багатьох.
4.	Нормалізація	Організовані дані (або ж вимоги до організації даних), для уникнення повторень та забезпечення консистентності даних.
5.	SQL	Використовує структуровану мову запитів для вибірки;
6.	ACID	Дані в кожному полі є атомарними, консистентність, ізоляція, довговічність.

№	Особливіть	Опис особливостей графових баз даних
7.	Додаткові можливості	Підтримує індексацію для швидкої вибірки інформації з бази даних.
8.	Транзакції	Можливість групувати запити в одну транзакцію, для гарантування атомарності та консистентності даних.
9.	Безпека	Надаються механізми для захисту та контролю доступу до даних.

MySQL – це відкрита реляційна СУБД, створена Oracle. Головна мова для взаємодії – SQL. Забезпечує швидкість, надійність, швидкодію [13]. Широко використовується у веб-розробці, бізнес-додатках, наукових дослідженнях, хмарних сервісах тощо.

Переваги реляційної бази даних MySQL:

– відкритість – є безкоштовною базою даних та з відкритим для публічного доступу вихідним кодом;

– драйвери та інтеграції – володіє широким вибором драйверів та бібліотек (для різних мов програмування), що робить розробку більш оптимальною та продуктивною;

– масштабованість – є підтримка горизонтального масштабування, щоб ділити дані на різні сервери;

– резерве копіювання (реплікація) – наявна можливість реплікації, яка забезпечує створювати резервні копії, тим самим забезпечуючи високу надійність та гнучкість системи;

– висока продуктивність та швидкість – внутрішні механізми оптимізації швидко обробляють великі масиви даних;

– популярність – наявність широкої спільноти розробників.

Недоліки реляційної бази даних MySQL:

– складні запити – наявна обмежена підтримка складних запитів (підзапити, вкладені запити);

– JSON – мало засобів для роботи з JSON-форматом (запити можуть бути менш продуктивними з цим форматом);

– витрата пам’яті – може вимагати великої кількості оперативної пам’яті;

– функції аналізу – має мало ефективних інструментів для аналізу та складних запитань;

– геопросторі запити – менш розвинені геопросторі запити.

Oracle Database – це комерційна реляційна СУБД, створена Oracle. Головна мова для взаємодії – SQL. Має набагато більший та ширший функціонал за MySQL, частіше використовується на великих підприємствах та компаніях, де має бути висока надійність, доступність, швидкість, масштабованість, продуктивність, оптимальність, ефективність, продуктивність та робота з геопросторовими даними тощо [14].

Переваги реляційної бази даних Oracle Database:

– висока продуктивність та швидкість – внутрішні механізми оптимізації швидко обробляють великі масиви даних;

– функціонал – в наявності розширений функціонал для обробки та роботи з великими масивами даних;

– безпека – база даних надає потужні інструменти для безпеки, наприклад: доступ по ролях, аудит, шифрування даних;

– документація та підтримка – при ліцензії є доступ до високотехнічної, обширної документації та підтримки від компанії;

– технологічність – система може підтримувати різні мови та технології, наприклад: Java, роботу з JSON, .NET тощо.

– аналітика – розвинуті інструменти для аналітичного аналізу, наприклад: аналітичні функції та OLAP-операції;

– хмарні технології – ефективно інтегрується з хмарними технологіями та інструмента.

Недоліки реляційної бази даних Oracle Database:

– ліцензійна підписка – це є найголовніший недолік, адже для студентів не найкращий варіант;

– складність – оскільки наявний обширний функціонал системи, важко налаштувати та оптимізувати систему;

– вимоги до ресурсів – через наявний обширний функціонал системи, вона вимагає значних обчислювальних ресурсів та пам'яті;

– замкнутість – як ліцензійний продукт, має високу замкнутість та вендорську залежність.

Таблиця 3.2 – Опис особливостей графових баз даних

№	Особливіть	Опис особливостей графових баз даних
1.	Представлення відносин	Легко моделювати та представляти відносини між об’єктами.
2.	Гнучкість	Можна додувати нові ребра та вузли до структури без всієї переконфігурації (тобто запезпечення швидкої та ефективної модифікації структури).
3.	Ієрархічні структури	Ефективно працюють з ієрархічними структурами, наприклад з генеалогічними деревами.
4.	Простий шарінг та реплікація даних	Забезпечується ефективний простий шарінг та реплікація даних, особливо в системах, які мають розподілену архітектуру.
5.	Підтримка великих графів	Створені для роботи з великими масивами даних та дають швидкий доступ до інформації у великих графових структурах.
6.	Запити для складних звітів	Дозволяють знаходити найкоротший шлях, визначити цикли, аналіз задач тощо.

Neo4j – це продуктивна графова база даних, де дані зберігаються у варіації графів, тобто використовується графова модель даних, де вузли – об’єкти, а ребра – відношення [15]. Це допомагає ефективніше аналізувати складні відношення між об’єктами.

Наявна своя мова запитів – Cypher, яка продуктивно взаємодіє з графовою системою. Ця графова база даних використовується при аналізі, соціальних мережах – тобто там, де важливі саме відношення між елементами.

Переваги графової бази даних Neo4j:

– гнучкість та ефективність графової моделі – завдяки графовій моделі є можливість ефективно та природно показати та проаналізувати відношення між об’єктами;

– популярність – наявність широкої спільноти розробників;

– Cypher – мова розроблена спеціально для роботи з графовими системами і дозволяє легко виражати питання й отримувати потрібні відповіді;

– швидкодія – графова система та оптимізація бази даних створює швидкодію при виконанні запитів;

– масштабованість – наявна гнучка система масштабованості, яка може розподіляти обробку даних на декілька серверів;

– управління транзакціями – є підтримка атомарних та консистентних транзакцій, що поліпшує та робить ефективним процес роботи з даними;

– візуалізація графів – система має інструменти для відображення графових об’єктів.

Недоліки графової бази даних Neo4j:

– складність установки – має складність під час встановлення та налаштування системи;

– ресурсомісткість – потребує значних ресурсів для обробки великих масивів даних або коли багато відношень (зв’язків);

– складність Cypher – хоч мова і допомагає у взаємодії з графами у базі даних, але є важкою у вивченні;

– розповсюдженість – оскільки йде робота з графами, це вузькоспеціалізоване направлення, тому використовується в обмежених сферах.

Amazon Neptune – керована графова база даних, де дані також зберігаються графами, тобто використовується графова модель даних, де вузли – об’єкти, а ребра – відношення.

Характерною особливістю для цієї бази даних є підтримка графів з орієнтованими та неорієнтованими ребрами, що забезпечує високу надійність та масштабованість для оптимального зберігання даних у формі графів. Також наявна свої мови запитів – SPARQL та Gremlin, яка продуктивно взаємодіють з графовою системою.

Переваги графової бази даних Amazon Neptune:

– безпека – база даних надає потужні інструменти для безпеки, наприклад: аутентифікація, аудит, шифрування даних;

– популярність – наявність широкої спільноти розробників;

– швидкодія – графова система та оптимізація бази даних створює швидкодію при виконанні запитів;

– масштабованість – наявна гнучка система масштабованості, яка може розподіляти обробку даних на декілька серверів;

– інструменти моніторингу – використовуються інструменти візуалізації та моніторингу, для ефективного відстеження графової системи бази даних та її оптимізації;

– інтеграція – оскільки це частина екосистеми AWS, простіше інтегрується з іншими інструментами, наприклад Amazon 3;

– SPARQL та Gremlin – підтримуються мови запитів SPARQL та Gremlin, які є продуктивними та ефективними у роботі в графовій системі.

Недоліки графової бази даних Amazon Neptune:

– вартість – дороге використання та дорогі додаткові послуги;

– складність SPARQL та Gremlin – хоч мови й допомагають у взаємодії з графами у базі даних, але є важкою у вивченні;

– швидкість зчитування та запису – ця швидкість є обмеженою, особливо при великому масиві даних;

– вибіркова реплікація – Neptune має обмежену вибіркову реплікацію даних по регіонах;

– налаштування – має обмежені можливості налаштування.

Таблиця 3.3 – Опис особливостей нереляційних баз даних

№	Особливіть	Опис особливостей графових баз даних
1.	Гнучкість	Є головною/основною перевагою для веб-застосунків, адже відсутність структурованної схеми дозволяє з легкістю додавати нові поля без зміни всієї схеми.
2.	Розподілена архітектура	Можливість працювати в розподіленому середовищі, що оптимізує обробку даних.
3.	Документи	Зберігання даних у так званих «документах», наприклад в JSON або BSON, яке дозволяє спростити відображення даних.
4.	Ключ-значення	Можливість зберігати в структурі «ключ-значення», для ефективного та оптимального зберігання великих масивів даних.
5.	Індексація	Використовується для миттєвого знаходження та доступу даних (може відрізнятись від індексації в реляційних базах даних).
6.	Можливість роботи без SQL	Не потребують структуровану мову запитів, це може полегшити роботу з операціями.
7.	Географічні дані	Деякі бази мають вбудовану функцію географічних даних для просторових даних.
8.	CAP-теорія	Засновані на концепції CAP-теореми, яка забезпечує консистентність, доступність та стійкість до розділення мережі.

MongoDB – це сучасна нереляційна СУБД, яка характеризується швидкодією, масштабованістю, гнучкістю в взаємодії з даними. Використовується документо-орієнтована модель, де JSON модель у BSON форматі. Важливою особливістю є застосування колекцій [16]. Вони відіграють головну роль організації та взаємодії з даними, забезпечуючи швидкість – для продуктивності

системи, гнучкість – для зберігання всіх типів даних, масштабованість – для роботи з великими масивами даних.

Переваги нереляційної бази даних MongoDB:

– гнучкість – є найголовнішою перевагою, адже дозволяє реалізувати різноструктурні дані в одній колекції;

– висока продуктивність та швидкість – внутрішні механізми оптимізації швидко обробляють великі масиви даних;

– популярність – наявність широкої спільноти розробників;

– оновлення – постійне оновлення, яке покращує функціональність та ефективність бази даних;

– індексація – можна створювати індекси, які допомагають швидко знайти потрібні дані та оптимізують запити. Також може робити автоматичну індексацію та механізми реплікації;

– mapReduce – вбудована підтримка функцій mapReduce для обробки та агрегації даних;

– горизонтальне масштабування – завдяки цьому масштабуванню дані розподіляються на різні сервери. Це допоможе оптимізувати роботу з великим масивом даних.

Недоліки нереляційної бази даних MongoDB:

– використання пам’яті – при обробці великих масивів даних може використовувати значну кількість оперативної пам’яті;

– join – є обмежування властивості з`єднань даних на відміну від реляційних баз.

CouchDB – це нереляційна СУБД, де дані зберігаються JSON-документами. Характерною особливістю CouchDB є реплікація даних між різними серверами для створення розподілених систем.

Переваги нереляційної бази даних CouchDB:

– легкість користування – CouchDB є простою у розгортанні та використанні.

– HTTP API – оскільки взаємодія відбувається через HTTP, це робить доступним використання різних мов програмування для цієї бази даних;

– гнучкість – дозволяє реалізувати різноструктурні дані в одній колекції;

– резерве копіювання (реплікація) – наявна можливість реплікації, яка забезпечує створювати резервні копії, тим самим забезпечуючи високу надійність та гнучкість системи;

– JSON – використовуються JSON-подібні документи, які є кращими та простішими для сприйняття.

Недоліки нереляційної бази даних CouchDB:

– транзакції – присутня слаба підтримка транзакцій, це може наблизити до проблем консистентності;

– індексація – повільна система індексації;

– ресурсомісткість – потребує значних ресурсів та пам’яті для обробки великих масивів даних;

– аутентифікація– відсутні засоби безпеки такі як аутентифікація або авторизація, що може призвести до зниження безпеки;

– mapReduce – база даних використовує власну мову запитів до даних, яка значно відрізняється від SQL і є важкою для розуміння.

Таблиця 3.4 – Опис особливостей об'єктно-орієнтованиих баз даних

№	Особливіть	Опис особливостей об'єктно-орієнтованиих баз
1.	Об'єктно-орієнтований підхід	Дозволяють моделювати дані об`єктами, що спрощує розуміння системи.
2.	Спадкування/поліморфізм	Використання спадкування та поліморфізму для створення ієрархії об’єктів, щоб спростити роботу зі спільними атрибутами в базі даних.
3.	Абстракція	Присутня абстракція даних для приховування деталей створення, щоб зосередитись на логіці.

№	Особливіть	Опис особливостей об'єктно-орієнтованиих баз
4.	Повторне використання	Завдяки об’єктно-орієнтованому підходу стає ліпше масштабованість та гнучкість системи.
5.	Багатопоточність	Деякі види об'єктно-орієнтованиих баз даних підтримують багатопоточність для поліпшення роботи з одночасним доступом.
6.	Доступ до даних	Зручний інтерфейс робить доступ до даних більш зручнішим тому спрощується взаємодія з інформацією в базі даних.
7.	JavaScript	Можливість використання JavaScript, адже ця мова програмування є досить популярною.

ObjectDB – це об'єктно-орієнтована база даних, розроблена для Java. Вона дозволяє зберігати об’єкти в самій базі даних, тобто це спрощення роботи з об’єктами в веб-розробці.

Підтримує індексацію, транзакції та запити для взаємодії з даними. Використовується для веб-розробки, системи управління контентом, системи реального часу тощо.

Переваги об'єктно-орієнтованої бази даних ObjectDB:

– простота – має простий інтерфейс для взаємодії з об’єктами, полегшуючи та оптимізуючи розробку;

– продуктивність – оскільки база даних працює з об’єктами, вона є оптимізованою для цієї роботи (продуктивна якщо робота зв’язана з об’єктами);

– JPA – підтримка Java Persistence API для взаємодії зі стандартами Java;

– трансформація даних – оскільки використовується об'єктно-орієнтований підхід, дані зберігаються в об’єктах, що спрощує трансформацію даних.

Недоліки об'єктно-орієнтованої бази даних ObjectDB:

– інструменти – менш розвинута екосистема технологій та інструментів у порівнянні з іншими видами баз даних;

– спільнота – є малою, що впливає на документацію, відповідей на запитання у групах, плагінів тощо.

– об’єктний підхід – вузькоспеціалізовний підхід, тому підходить не для всіх сценаріїв розробки;

– Java – присутня сильно орієнтацію на мову програмування Java;

– ресурсомісткість – для забезпечення стабільної роботи потрібен велика кількість пам’яті та ресурсів.

LokiJS – це вбудована об'єктно-орієнтована база даних, розроблена для мови програмування JavaScript. Славиться своєю швидкодією, ефективністю, продуктивністю та легкістю.

Вона оптимізована для роботи у браузерах або ж на сервері. Також ідеально підходить для роботи для вбудованих систем та веб-застосунків, адже використовує JSON-подібні об’єкти та надає API, щоб взаємодіяти з цими об’єктами в базі даних.

Переваги об'єктно-орієнтованої бази даних LokiJS:

– продуктивність – оскільки база даних працює з JSON-об’єктами, вона є оптимізованою для цієї роботи (продуктивна якщо робота зв’язана з об’єктами);

– Node.js – LokiJS має ефективну взаємодію з Node.js, тому може бути використана під управлінням Node.js;

– API – взаємодія з JSON-об’єктними структурами можливо через API, тому наявний прості та ефективні інструменти для цього;

– автоматичне збереження – LokiJS автоматично зберігає свої дані, запобігаючи втрат даних;

– індексація – можна створювати індекси, які допомагають швидко знайти потрібні дані та оптимізують запити. Також може робити автоматичну індексацію та механізми реплікації;

– оновлення – постійне оновлення, яке покращує функціональність та ефективність середовища розробки;

– резервне копіювання – наявні механізми резервного копіювання (або ж відновлення даних), щоб забезпечити відновлення системи у разі непередбачуваних помилок чи збоїв;

– підтримка запитів – наявні вбудовані запити для ефективної взаємодії з інформації з JSON-об’єктних структур;

– універсальність (використання в різних частинах системи) – може використовуватись як і на серверній стороні веб-додатку, так і на клієнтській стороні, що робить цю базу даних універсальною.

Недоліки об'єктно-орієнтованої бази даних LokiJS:

– новизна – є відносно новим середовищем, тому має менше рішень та спільноту розробників. Також може мати не детальну документацію;

– ресурсомісткість– потребує значних ресурсів для обробки великих масивів даних або коли багато відношень (зв’язків);

– масштабованість – відсутність механізмів масштабованості, тому ця база даних найкраще підходить для взаємодії з веб-застосунками або додатками, де є невеликий обсяг даних;

– робота в локальному середовищі – відсутня наявність вбудованих можливостей для роботи в розподіленому середовищі, адже LokiJS майже завжди використовується в локальному середовищі;

– багатокористувацький режим – немає вбудованого багатокористувацького режиму.

3.5 Вибір інструментів для розробки інформаційної системи

Підхід до вибору технологій є важливим етапом, бо він закладає основу створення майбутньої ефективної інформаційної системи, її архітектури та функціоналу.

Тому після проведеного огляду та аналізу з вище наведених інструментів та технологій (бекенд-середовищ, фронтенд-бібліотек, баз даних), я вирішив обрати створення інформаційної системи для барбершопів на основі сучасного стека

технологій «MERN» (MongoDB, Express.js, React, Node.js). Структурна робота стека продемонстрована на рис. 3.1.

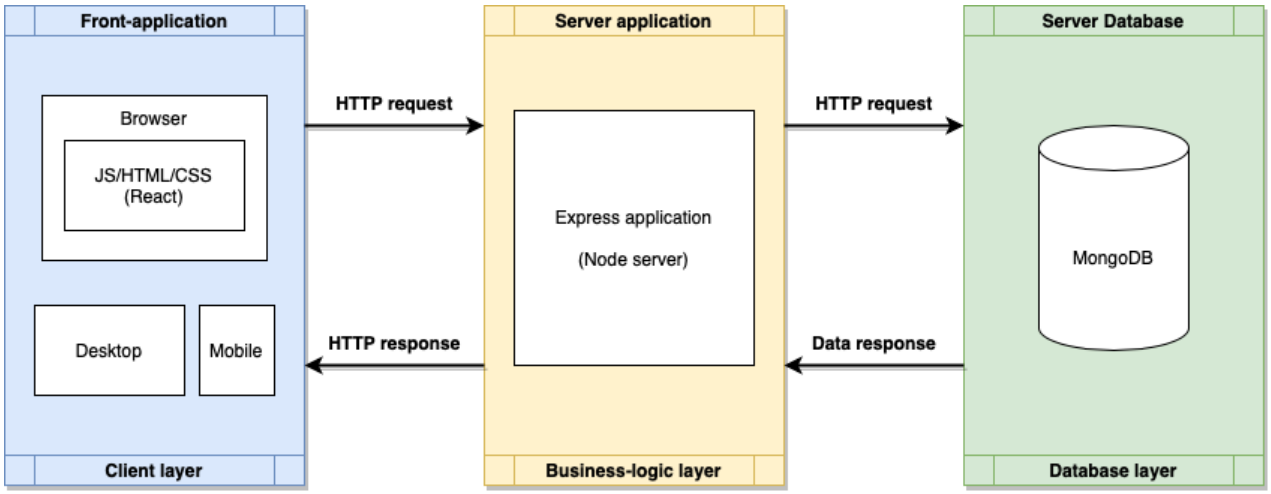

Рисунок 3.1 – Структура роботи стека технологій «MERN»

В даному стеці:

– MongoDB – зберігає дані у форматі документів, що надає гнучкість та масштабованість;

– Express.js – використовується для створення й управління API (забезпечення ефективного обміну даних). Також реалізує логіку на стороні серверу;

– React – потрібен для створення інтуїтивно зрозумілого та зручного інтерфейсу, який швидко та ефективно оновлюється. Технологія пропонує компонентний підхід для реалізації незалежних модулів, дозволяючи розробникам робити з високою продуктивністю та мінімальними зусиллями ефективний інтерфейс;

– Node.js – надає середовище на серверній стороні, яке полегшує швидке виконання коду й допомагає взаємодіяти з іншими технологіями разом оптимально та продуктивно;

Обрання цього стеку відображає не тільки мої технічні вподобання, а й забезпечує потреби моєї інформаційної системи в ефективності, швидкодії, масштабованості та гнучкості.

3.6 Висновки до розділу 3

Як висновок, використання стеку технологій «MERN» для інформаційної системи мережі барбершопів є обґрунтованим. Адже взаємодія даних технологій робить систему швидкою, гнучкою, надійною, здатною до майбутніх модифікацій та поліпшень, ефективною.

Тобто, інформаційна система для мережі барбершопів покращить управління, підхід до ведення бізнесу, обслуговування клієнтів та оптимізацію графіків персоналу у сфері краси.

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Основні функції інформаційної системи

Завдяки наведеному аналізу та огляду у розділі 2, інформаційна система для підтримки діяльності мережі барбершопів має містити в собі функціонал, який наведений в табл. 4.1. Також, схематично функціонал інформаційної системи наведений у додатку Б.

Таблиця 4.1 – Опис основних функцій системи

№	Функціонал	Опис функціоналу
1.	Реєстрація клієнта	Потенційний клієнт може зареєструватись, щоб з’явитись в базі клієнтів барбешопу.
2.	Авторизація	Захист системи від несанкціонованого доступу. Для клієнта дає змогу відстежувати історію своїх покупок та записів, переглядаючи їх детальну інформацію. Також оновлюється кошик та розділ улюблені товари.
3.	Реєстрація барбера	Після авторизації та отримання доступу, тільки менеджер може додати нового барбера, зареєструвавши його в своєму кабінеті.
4.	Запис на послуги	Клієнт обирає оптимальну для нього дату, послугу, час та барбера. Після цього відбувається онлайн-запис.
5.	Продаж товарів	Клієнт обирає товари, після цього вказує всі свої персональні дані, адресу доставляння та спосіб оплати. Далі відбувається купівля товарів.
6.	Оновлення та поширення новин	Після авторизації та отримання доступу, тільки менеджер через свій кабінет в системі може поширити новину для відображення її в системі.

№	Функціонал	Опис функціоналу
7.	Зворотній зв’язок	Після купівлі товару або запису на послуги, клієнту надходить повідомлення про підтвердження купівлі або запису на його електронну пошту, яку він вкаже в формі.
8.	Графік барбера	При авторизації, барбер може переглядати свої записи до нього з вказаним часом та датою у своєму кабінеті.
9.	Додання та оновлення інформації	Кожен барбер та клієнт, після реєстрації може додати або оновити свою персональну інформацію в кабінеті (для барберів тільки ім’я та прізвище, які будуть відображатись у списку при записі клієнта на послугу).
10.	Побажання та пропозиції	Кожен клієнт може надіслати свої побажання та пропозиції через спеціальну форму в системі, а менеджер в свою чергу може їх бачити та реагувати на них.
11.	Додавання товарів в розділ «улюблене»	При авторизації, клієнт має змогу додати товари, які йому сподобались в спеціальний розділ «улюблене».
12.	Взаємодія з онлайн-записом	Клієнт при авторизації має змогу скасувати свій візит на послугу у своєму кабінеті.
13.	Планування графіків та відстеження популярності послуг	Менеджер має змогу у своєму кабінеті переглядати всі онлайн-записи, тим самим оптимізуючи графіки барберів й аналізуючи рівень популярності послуг.
14.	Звіт та аналітика з продажу товарів	Менеджер має змогу у своєму кабінеті переглядати та аналізувати всі замовлення

№	Функціонал	Опис функціоналу
		клієнтів, тим самим відстежуючи та оновлюючи асортимент товарів
15.	Інформування клієнтів (надсилання рекламних акцій)	Менеджер має змогу у своєму кабінеті розсилати рекламні акції або новини про розширення послуг барбершопу, щоб залучити до себе більше клієнтів.

Взаємодія функціональності ІС та акторів (користувачів) продемонстрована у вигляді діаграми прецедентів у додатку В.

4.2 Розробка серверної частини системи та технології для розробки

В даному розділі пояснювальної записки буде описана розробка серверної частини інформаційної системи мережі барбершопів на стеці технологій «MERN», включаючи опис використаних інструментів та технологій.

4.2.1 Ініціалізація та налаштування середовища

Після огляду всіх переваг та недоліків середовищ для розробки програм, найефективнішим виявився – Visual Studio Code. Тому дана інформаційна система для мережі барбершопів була розроблена за допомогою Visual Studio Code, який є популярним та ефективним редактором коду на сьогодні.

Мовою програмування обрана JavaScript. JavaScript – це високорівнева та популярна на сьогодні мова програмування, яка використовується для спеціалізованого створення веб-застосунків. Вона підтримує всі типи даних, об’єкти, логічні значення, рядки та навіть принципи ОПП (є «синтаксичним цукром»). Це дозволяє створювати ефективні, масштабовані, гнучкі, надійні, легкі у розробці веб-додатки.

Щоб почати ініціалізацію серверної частини на пристрій розробки було встановлено Node.js та Node Packet Manager (NPM). Вони надають ефективні

інструменти для розробки серверної частини інформаційної системи та збільшують продуктивність її розробки.

Node.js – дозволяє використовувати мову JavaScript, що уніфікує загальну розробку системи, адже використання єдиної мови програмування на всіх етапах оптимізує час розробки.

NPM в свою чергу потрібен для керування залежностями інформаційної системи та встановлення пакетів. Пакет в NPM – це структурований файл з кодом, який потрібен для використання конкретного функціоналу чи бібліотеки в інформаційній системі [17].

Після цього була проведена ініціалізація проекту, завдяки команді npm init. Вказується вся початкова інформація про проект та його налаштування для інформаційної системи.

4.2.2 Архітектура та вхідний файл

Архітектура серверної частини системи є важливим аспектом її реалізації, адже саме архітектура визначає структуру системи, організацію папок та файлів та оптимальність їх взаємодії один з одним.

Оптимально спроектована архітектура робить систему інтуїтивно-зрозумілою у розумінні, сприяє масштабованості, розширеності, ефективності та підтримці проекту. Архітектура в серверній частині системи зображена на рис. 4.1.

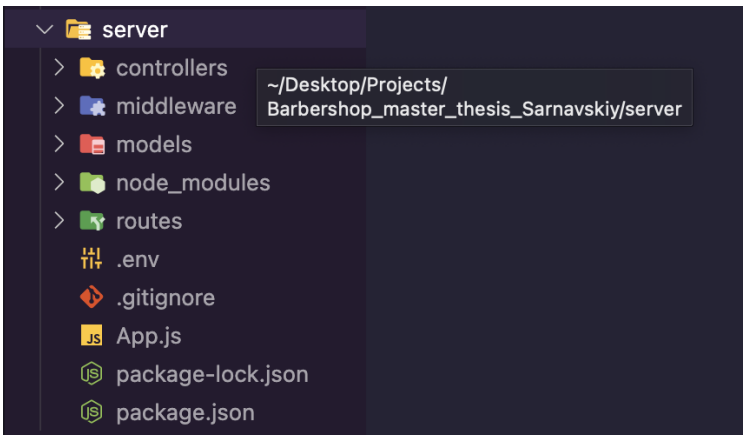

Рисунок 4.1 – Архітектура в серверної частини системи

Вона включає в себе наступні папки та файли:

– папка contollers – зберігаються файли, які відповідають за логіку обробки запитів з клієнтської частини;

– папка middlewares – зберігаються файли, які відповідають за функції, які потрібно виконати перед обробкою запиту;

– папка models – зберігаються файли, у яких присутні моделі даних. Вони описують структуру, тип даних та додаткові налаштування;

– папка node_modules – зберігаються залежності серверної частини інформаційної системи;

– папка routes – зберігаються файли, які відповідають за маршрутизацію в інформаційній системі;

– файл .env – файл конфігурації, який дозволяє зберігати секретні змінні серверного середовища (наприклад: паролі, адреси, URL-для з'єднання з БД);

– файл .gitignore – файл, який вказує системі керування версіями (Git), які файли та папки повинні ігноруватись при відправки на віддалений репозиторій (зазвичай складається з папок build, node_modules та інших спеціалізованих файлів з середовища виконання. build, node_modules – дані папки мають великий розмір даних, що може ускладнити передачу);

– файл package-lock.json – файл, який гарантує коректне встановлення NPM-пакетів. В ньому зберігаються версії пакетів та їх залежності;

– файл package.json – файл, який є конфігурацією середовища. Він вказує назву, версію, скрипти тощо.

Головним або «вхідним» файлом є server.js. Він відіграє головну роль у налаштуванні та управлінні серверної частини інформаційної системи. До нього підключаються різні залежності, головною є Express.

Express – гнучкий та ефективний фреймворк, який відповідає за створення веб-застосунків, завдяки мові JavaScript. Головними його функціями є спрощення створення ефективного API та розробки веб-додатків, роблячи процес реалізації більш продуктивним та зручним [18].

Функції фреймворку Express складають:

– middleware – використання концепції middleware, для обробки функції, які потрібно виконати перед обробкою запиту;

– маршрутизація – у легкий спосіб можна визначити маршрути, щоб вказати серверу який вид він повинен обробляти;

– шаблонізатори – можуть використовуватись різні шаблонізатори;

– статичні файли – є вбудований процес обробки та відправки статичних файлів (наприклад: CSS, HTML, тощо);

– інтегрованість – фреймворк можна легко розширювати.

Також, в головному або «вхідному» файлі server.js застосовується бібліотека Mongoose. Застосування в server.js фреймворку Express та бібліотеки Mongoose продемонстровано на рис. 4.2.

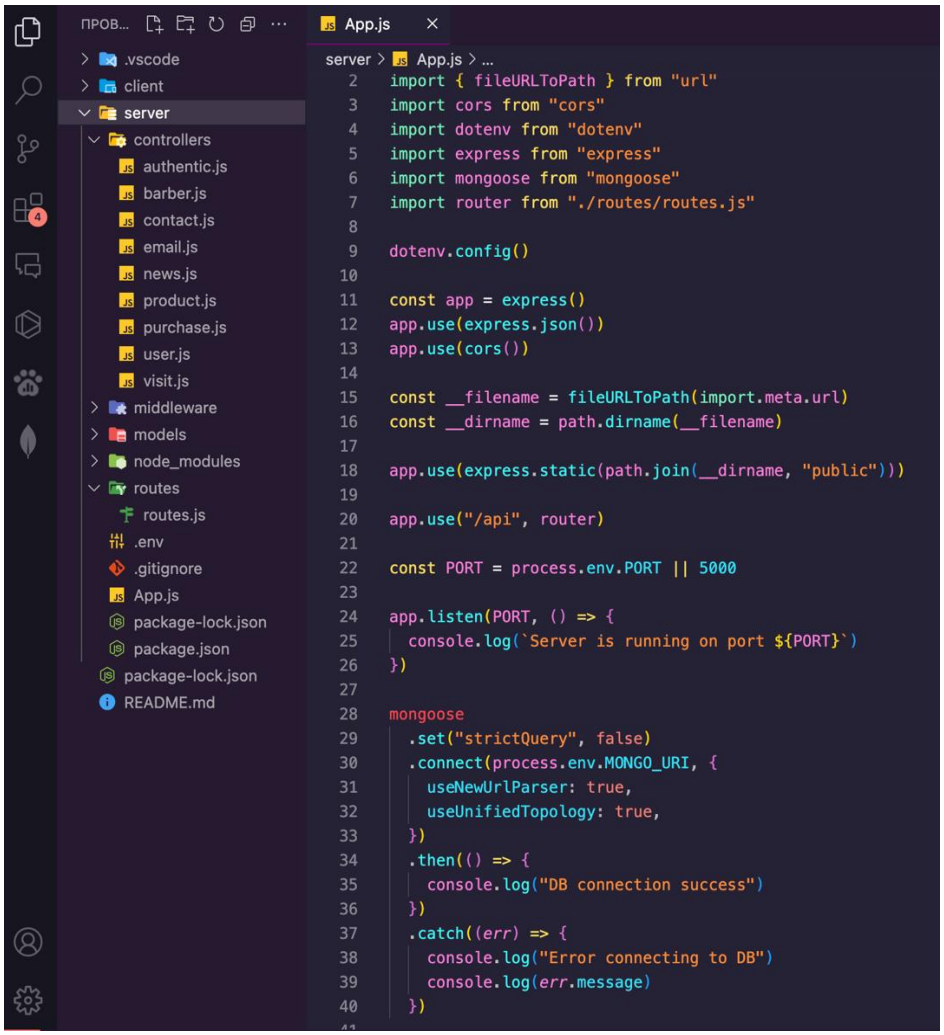

Рисунок 4.2 – Головний або «вхідний» файл server.js

4.2.3 Підключення бази даних та опис процесу обробки запиту

Mongoose – це ефективна бібліотека для роботи з MongoDB в Node.js на мові програмування JavaScript. Як було наведено в описі технологій минулого розділу, MongoDB є нереляційною базою даних і працює з JSON-подібними документами. Mongoose надає інструменти для швидкої, зручної та ефективної роботи з MongoDB (тобто, міст між MongoDB та Node.js) [19]. На рис. 4.3 зображено з'єднання з MongoDB, завдяки методу connect(), де URL для підключення знаходиться в змінній файлу конфігурацій.

Також Mongoose допомагає у визначенні схеми. Вони зберігаються у папці models. Схема в Mongoose – це структура даних, де описується структура, тип даних, поля. На основі схеми створюється модель, яка вже буде взаємодіяти з колекцією в базі даних MongoDB. Створення схеми, а потім моделі користувача та візиту зображено на рис. 4.3.

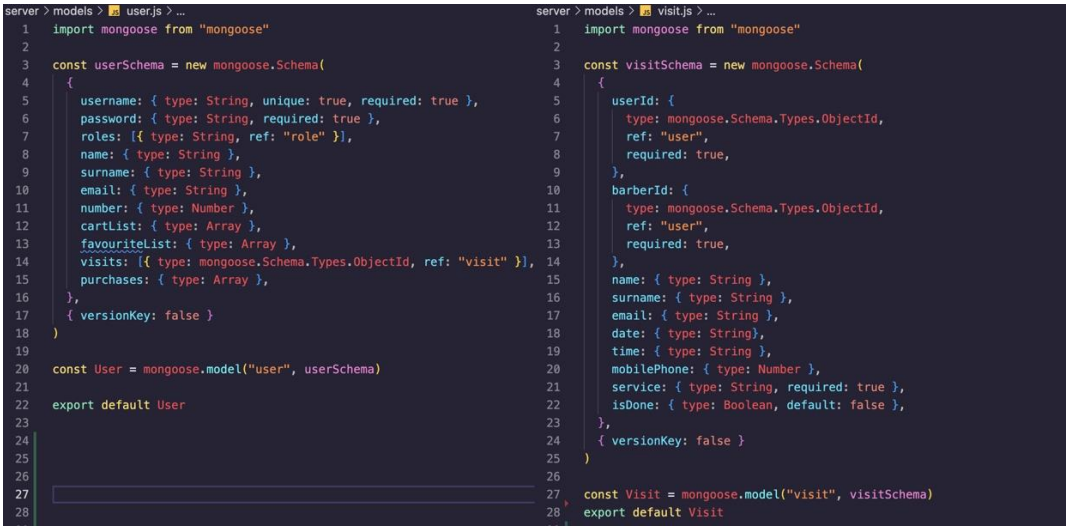

Рисунок 4.3 – Створення моделі користувача та візиту

У папці routes зберігаються маршрути. Це важлива частина взаємодії серверної частини з клієнтською. Маршрутизація визначає, які дії повинні виконуватись для різних URL-запитів. Це допомагає зберігати всі маршрути організованими та структурованими. Приклад файлів папки routes продемонстровано на рис. 4.4.

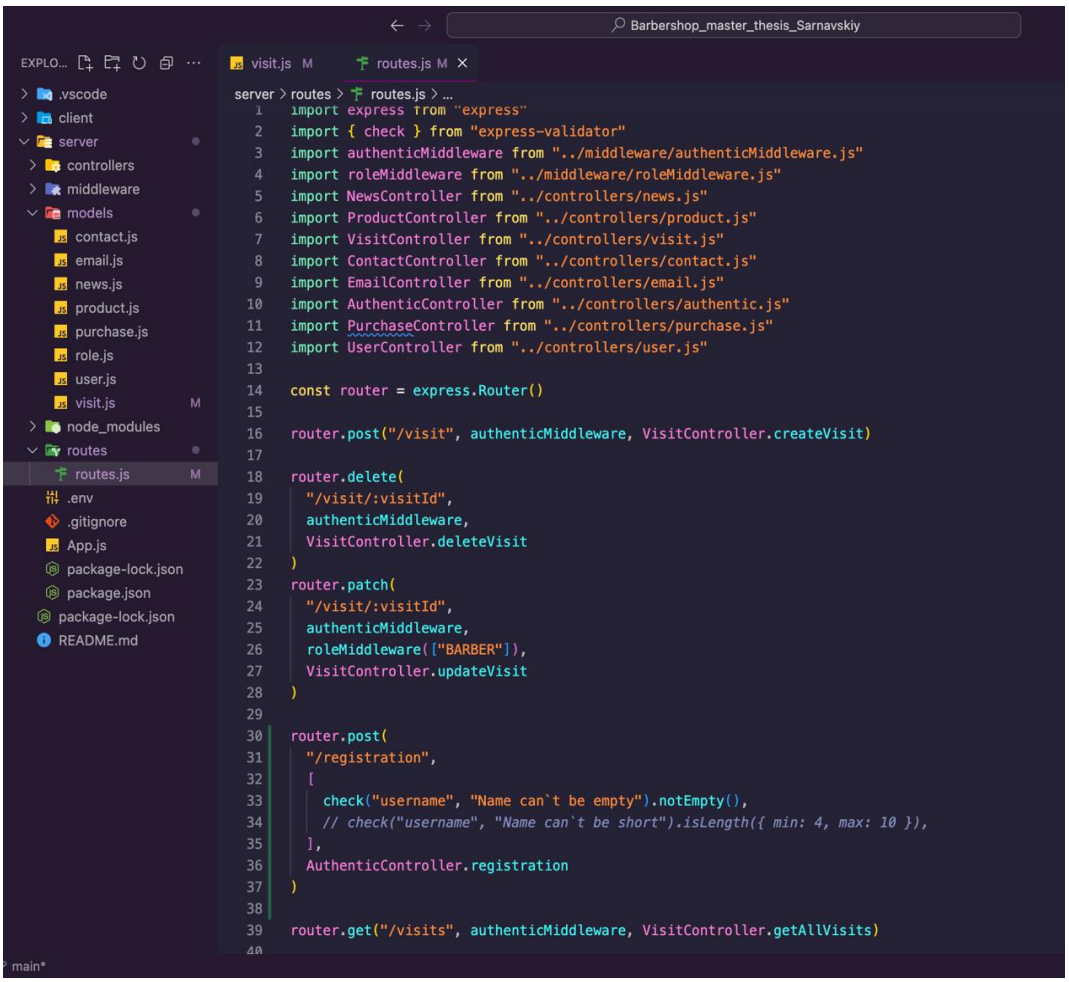

Рисунок 4.4 – Файл routes для маршрутизації запитів

Також в файлі routes під час маршрутизації виконуються спочатку мідлвари, а потім – контроллери. Тому спочатку пояснення мідлварів.

У папці middleware знаходяться файли з проміжними обробниками. Проміжні функції – це функції, які мають доступ до об’єктів запиту-відповіді (це об’єкти res, req), та наступної функції next у циклі. Тобто основна їх ціль використання – це виконання заданих дій перед основним виконанням контролером, що розширює функціонал обробки запиту.

Проміжні функції ідеально підходять для визначення доступу до певних запитів чи ресурсів у інформаційній системі. Тобто, йде перевірка чи має користувач токен авторизованості (зображено на рис. 4.5), які він має права доступу до ресурса. Схема процесу аутентифікації користувачів зображена у додатку Г.

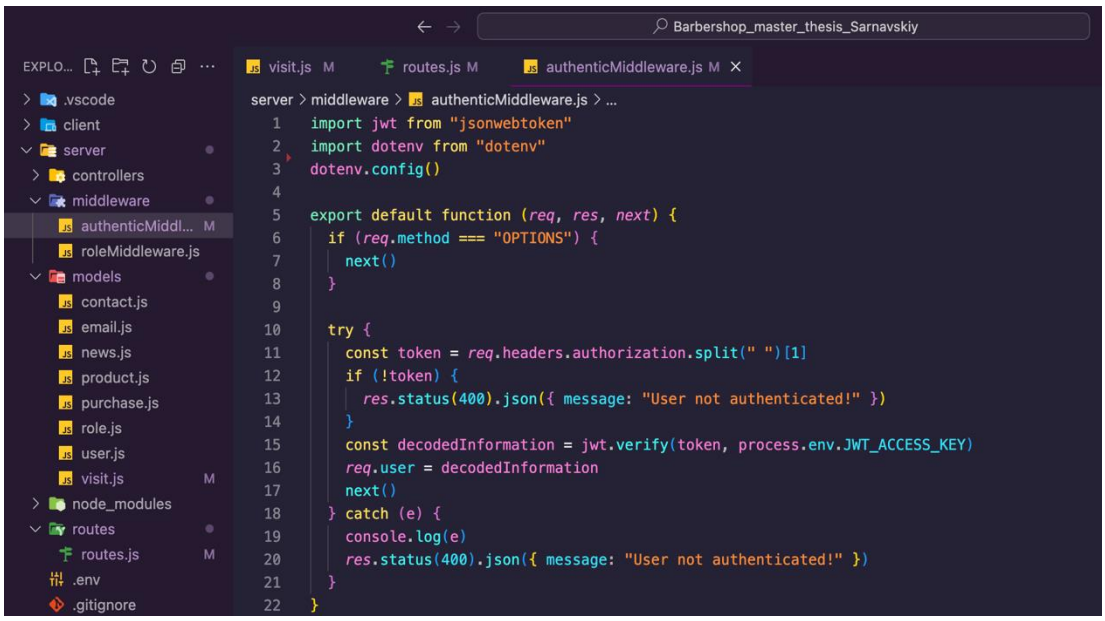

Рисунок 4.5 – Файл middleware для перевірки авторизованості користувача

Після мідлварів та перед виконанням обробки контролером може йти перевірка або валідація даних у вигляді функції check() бібліотеки express-validator, яка також зображена на рис. 4.4.

Express-validator – це ефективна бібліотека для валідації даних при запиті фреймворку Express. Перевіряється коректність даних згідно заданим вимогам та правилам і якщо знаходить помилку – автоматично її обробляє, тим самим підвищуючи безпеку інформаційної системи.

Після проміжних функцій та перевірки даних, запит передається контролерам, які знаходяться в папці controllers.

Контролери в даній архітектурі та в контексті Express.js забезпечує основну обробку логіки системи та взаємодіє з даними, які приходять з запиту. Приклад контролеру створення візиту продемонстрований на рис. 4.6. Контролери мають наступні функції:

– обробка запитів – при різних типів запитів HTTP або HTTPS, контролери визначають методи для кожного запиту – свій метод;

– логіка – зберігається основна логіка система, обробка та маніпулювання даними з запиту;

– взаємодія з моделями – працюють з моделями, які представляють дані та виконують операції над ними. Також викликаються методи моделей для маніпуляції над даними;

– обробка помилок – оброблюються помилки, які можуть статись під час обробки запиту і відправляти їх на клієнтську частину;

– розділення відповідальності – контролери розподіляють відповідальність між компонентами системи.

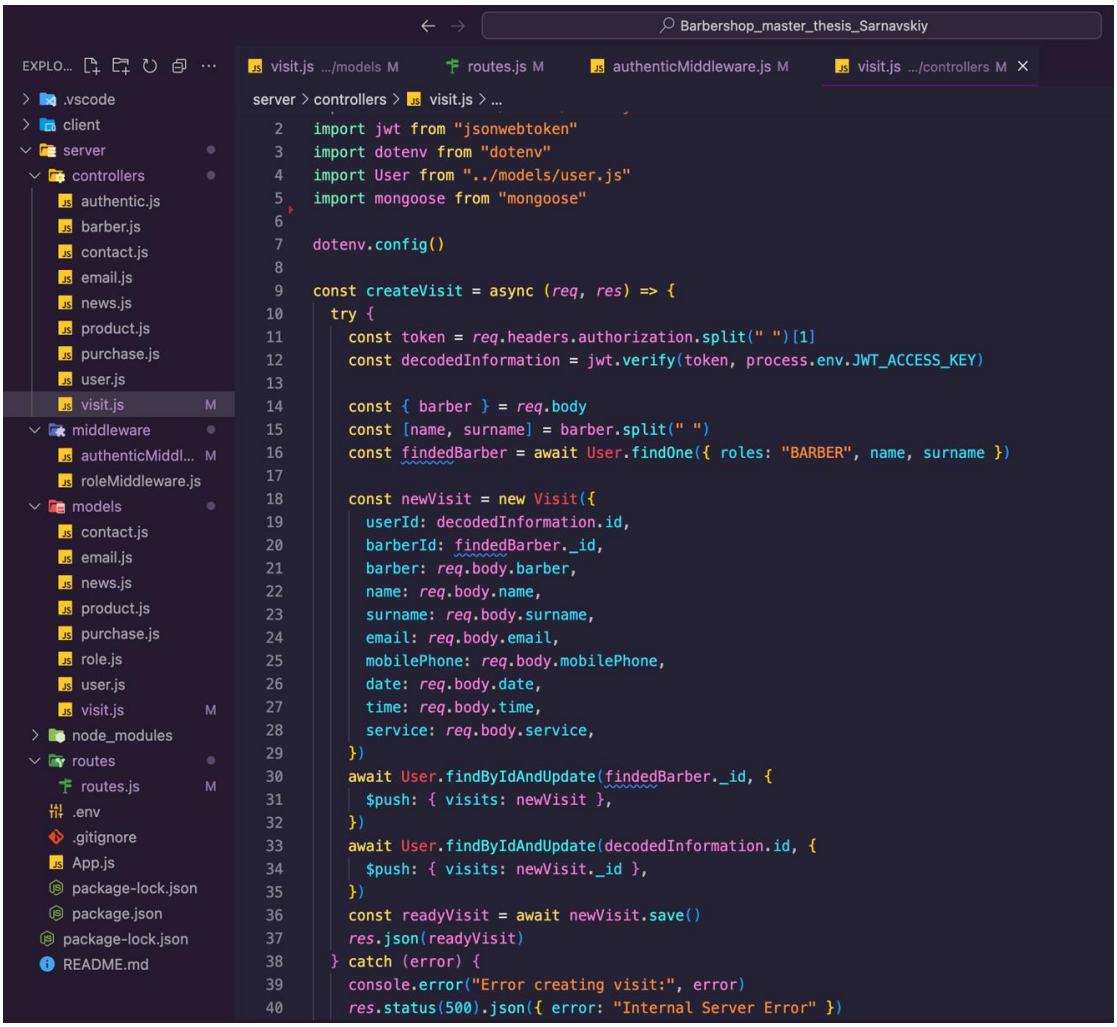

Рисунок 4.6 – Файл middleware, а саме чи авторизований користувач

Після обробки даних контролером, в прикладі на рис. 4.6. завдяки функції save() візит, зроблений користувачем, надішлеться в нереляційну базу даних (так як в методі HTTP був POST запит і продемонстровано на рис. 4.7), а відповіддю на клієнтську сторону прийде об’єкт створеного візиту.

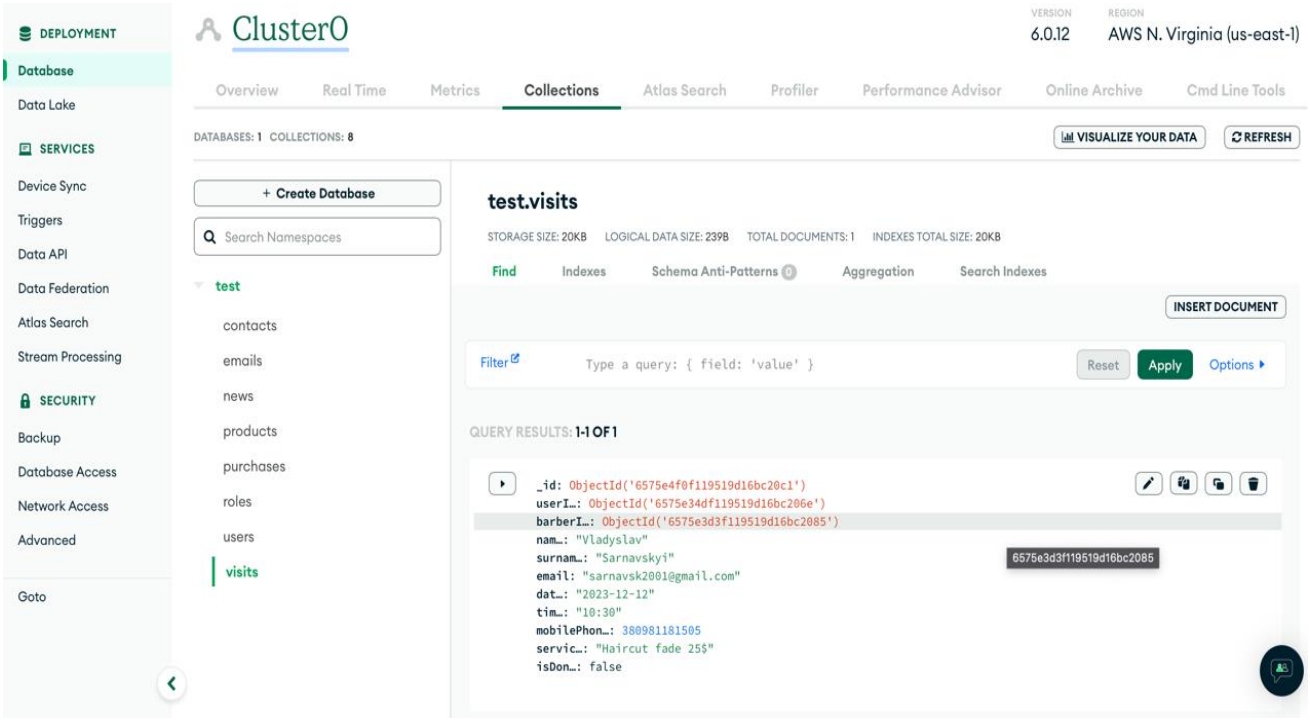

Рисунок 4.7 – Створення документа візиту в колекції visits в базі даних MongoDB

Загалом, інформація про зміст бази даних MongoDB (колекції, документи, розмір даних, індекси тощо) продемонстрована на рис. 4.8. Схема нереляційної бази даних MongoDB зображена у додатку Д.

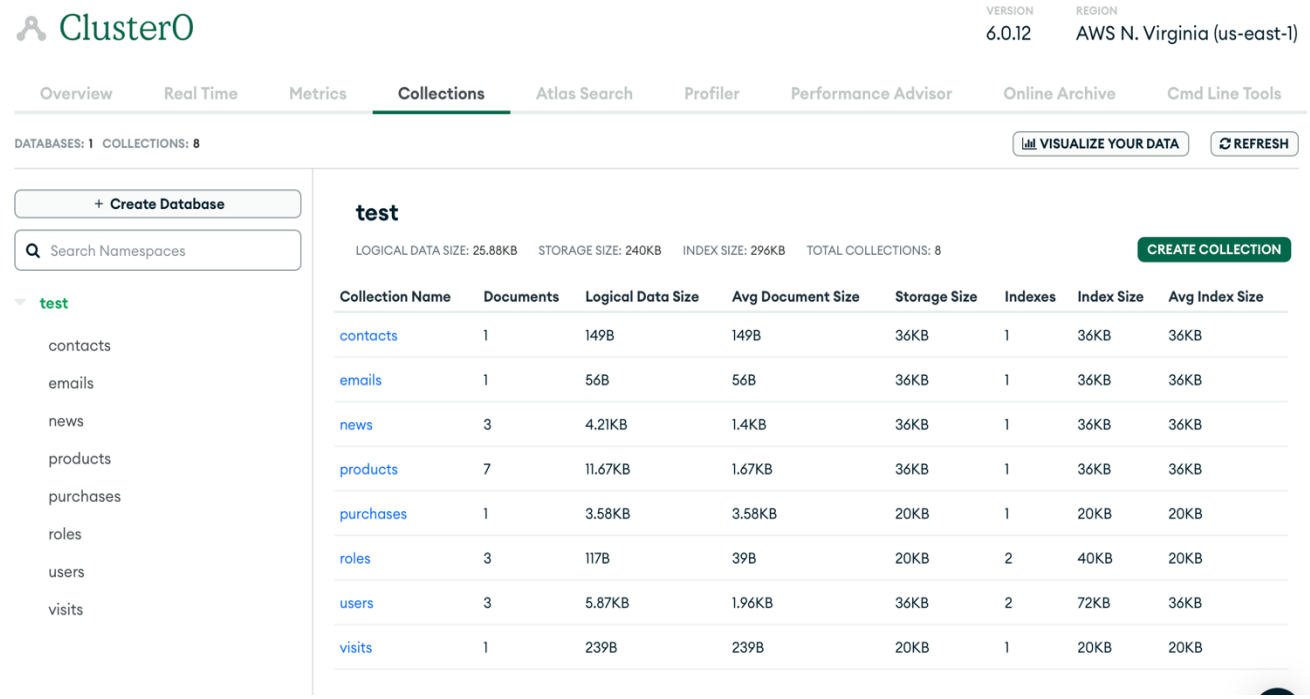

Рисунок 4.8 – Інформація про зміст бази даних

Для налаштування роботи серверної частини та перевірки коректності виконання задач, використовувався інструмент Postman. Postman – це сучасний інструмент для розробки та тестування API. Він спрощує розробку API, де завдяки його застосуванні можна створювати, отримувати HTTP запити та переглядати їх результати. Інтерфейс інструмента продемонстровано на рис. 4.9.

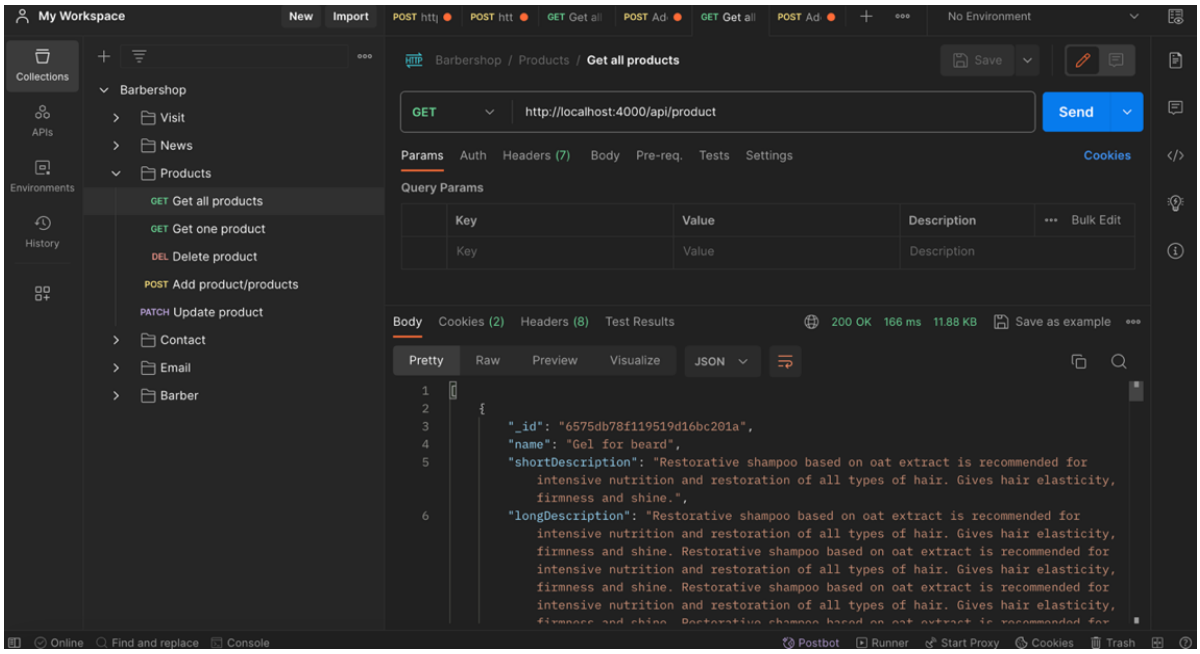

Рисунок 4.9 – Інтерфейс та результат роботи отримання всіх продуктів у Postman

4.3 Розробка клієнтської частини системи технології для розробки

В даному розділі пояснювальної записки буде описана розробка клієнтської частини інформаційної системи мережі барбершопів на стеці технологій «MERN», включаючи опис використаних інструментів та технологій.

4.3.1 Ініціалізація та налаштування середовища

Клієнтська частина буде також розроблятись у середовищі розробки Visual Studio Code. Мовою програмування обрана також JavaScript, яка використовується для спеціалізованого створення веб-застосунків.

Застосовується бібліотека React, яка працює принципом розподілу на незалежні компоненти. Кожен компонент реалізує свій стан та має свою логіку та відображення. Також використовується віртуальний DOM, що дозволяє швидко та ефективно відображувати зміни. Бібліотека є популярною й тому має безліч ефективних інструментів та технологій для продуктивної та оптимальної розробки.

В React також застосовується мова програмування JavaScript та також JSX, яке є розширенням синтаксису JS та дозволяє в React використовувати HTML-подібний код. Цей спосіб робить розробку простішою, адже мову розмітки легко розуміти.

Для стилізації використовується SCSS, яке є модифікованим та сучасним розширенням CSS. Це розширення надає додатковий функціонал та полегшує розробку клієнтської частини (міксини, медіа-запити, БЕМ-методологія, яка використовується в даній інформаційній системі).

Також використовується Babel. Babel – це транспілятор для перетворення коду JS з сучасного стандарту в попередні стандарти (щоб старіші версії браузерів могли коректно розпізнавати код JavaScript). Babel застосовується і для обробки JSX в звичайний JS, щоб програма могла розпізнати написаний код.

Щоб почати ініціалізацію серверної частини на пристрій розробки було встановлено Node.js та Node Packet Manager (NPM). Вони надають ефективні інструменти для розробки клієнтської частини інформаційної системи та збільшують продуктивність її розробки.

Після цього була проведена ініціалізація проекту, завдяки команді npx create-react-app Barbershop_master_thesis_Sarnavskiy. Вказується вся початкова інформація про проект та його налаштування для інформаційної системи.

4.3.2 Архітектура та вхідний файл

Архітектура клієнтської частини системи є важливим аспектом її реалізації, адже саме архітектура визначає структуру системи, організацію папок та файлів та оптимальність їх взаємодії один з одним.

Оптимально спроектована архітектура робить систему інтуїтивно-зрозумілою у розумінні, сприяє масштабованості, розширеності, ефективності та підтримці проекту. Архітектура в клієнтській частині системи продемонстрована на рис. 4.10.

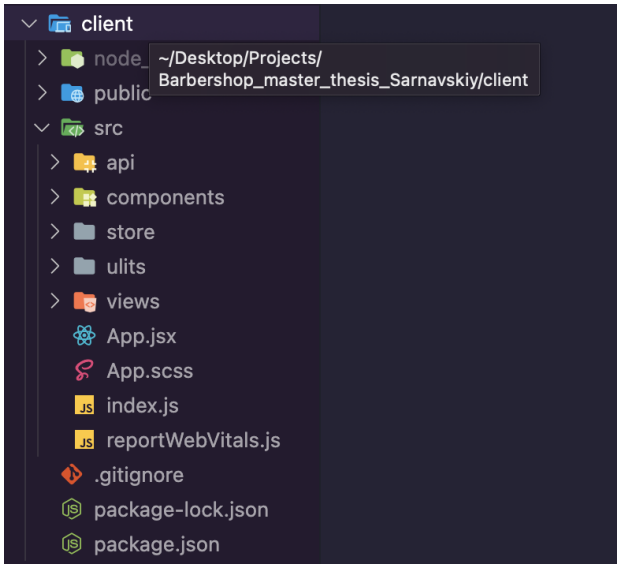

Рисунок 4.10 – Архітектура в клієнтської частини системи

Архітектура включає в себе наступні папки та файли:

– папка node_modules – зберігаються залежності серверної частини інформаційної системи;

– папка build (при розгортанні системи*) – зберігається готові для розгортання файли;

– папка public – папка призначена для публічного доступу (зберігається файл index.js та всі зображення чи відео);

– папка src – є папкою, в якій розробляється проект;

– папка api – папка для надсилання запитів;

– папка components – папка з файлами компонентів, які можуть бути перевикористанні;

– папка store – для взаємодії з глобальним станом (якщо застосовується бібліотека Redux);

– папка ulits – допоміжні файли для розробки (міксини, змінні, допоміжні функції тощо);

– папка views – відображення різних вкладок або сторінок в веб-застосунку;

– файл App.jsx – є основним файлом, в якому визначається структура та логіка відображення клієнтської частини;

– файл App.scss – файл стилізації для основного файлу;

– файл index.js – вхідний файл (виконується рендеринг в DOM);

– файл reportWebWitals – для збору та звітності про події;

– файл .gitignore – файл, який вказує системі керування версіями (Git), які файли та папки повинні ігноруватись при відправки на віддалений репозиторій (зазвичай складається з папок build, node_modules та інших спеціалізованих файлів з середовища виконання. build, node_modules – дані папки мають великий розмір даних, що може ускладнити передачу);

– файл package-lock.json – файл, який гарантує коректне встановлення NPM-пакетів. В ньому зберігаються версії пакетів та їх залежності;

– файл package.json – файл, який є конфігурацією середовища. Він вказує назву, версію, скрипти тощо;

Головними або «вхідними» файлами є index.js та App.js, які зображені на рис. 4.11. Вони відіграють головну роль у налаштуванні та управлінні серверної частини інформаційної системи. До них підключаються різні залежності.

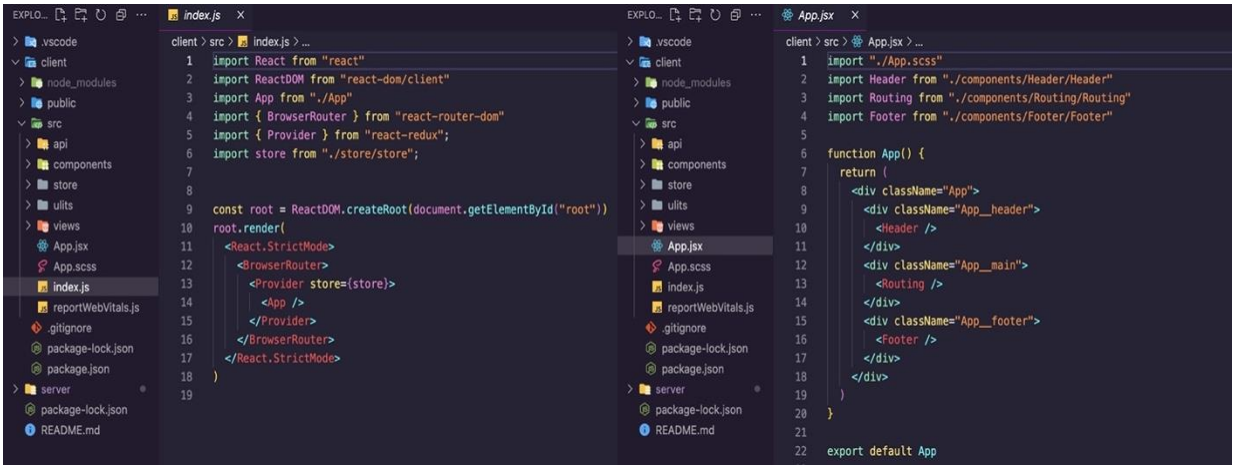

Рисунок 4.11 – Основні файли index.js та App.js

В даних файлах підключаються роутинг (бібліотека React Router) та використання глобального стану (Redux). В папці component присутній файл Routing.jsx (продемонстровано на рис. 4.12), в якому обробляються переходи на різні сторінки, щоб вони не перезавантажувались, збільшуючи швидкодію та ефективність системи.

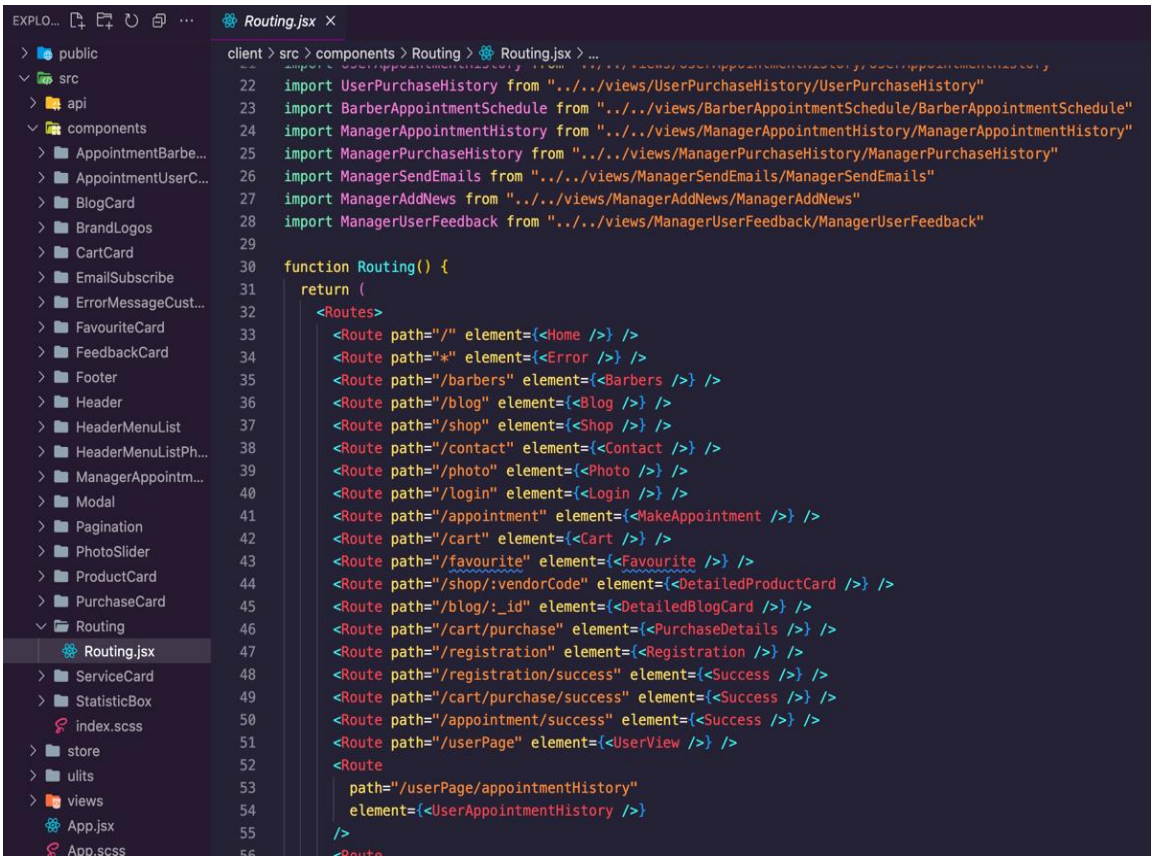

Рисунок 4.12 – Файл Routing.jsx

Папка ulits зберігає в собі допоміжні файли, які спрощують та оптимізують процес створення системи. Так як застосовується SCSS, він дозволяє використовувати змінні (потрібно виділяти символом «$»), міксини (блок коду зі стилізацією, який можна перевикористати в SCSS), оператори функцій, імпорти. Тому в папці ulits наведені файли reset (так як браузери різні, вони мають кожен свої різні вбудовані стилі, а reset – усуває їх), міксини (дозволяють робити медіа запити для розробки «респонсивного» веб-застосунку, наведено в рис. 4.13), та змінні.

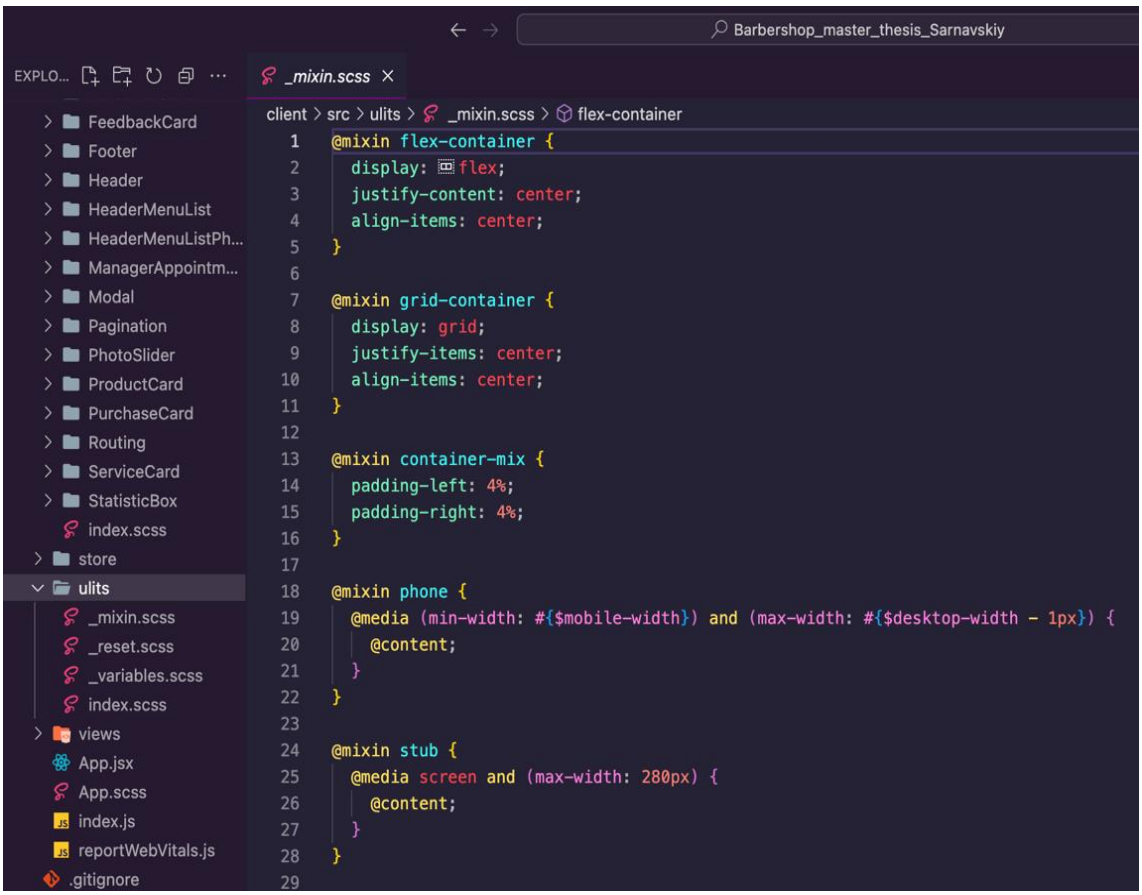

Рисунок 4.13 – Файл mixin.scss

В папках views та components, зберігаються файли основних компонентів клієнтської частини, які відображуються користувачу. Різниця в тому, що компоненти в views є окремими сторінками, а в components є меншими та використовуються в views (можуть і перевикористовуватись). Компоненти в цих папках є основою в клієнтській частині.

Папка api, зберігає в собі файли, які використовується для взаємодії з сервером або з іншими API, завдяки axios.

Axios – це бібліотека, яка спрощує процес та функціонал виконання HTTP запитів. Дана бібліотека надає зручний та оптимальний інтерфейс для легкого відправлення даних та обробки відповідей.

Використання даної бібліотеки продемонстровано на рис. 4.14, де наведені різні типи запитів щодо візитів на серверну частину системи.

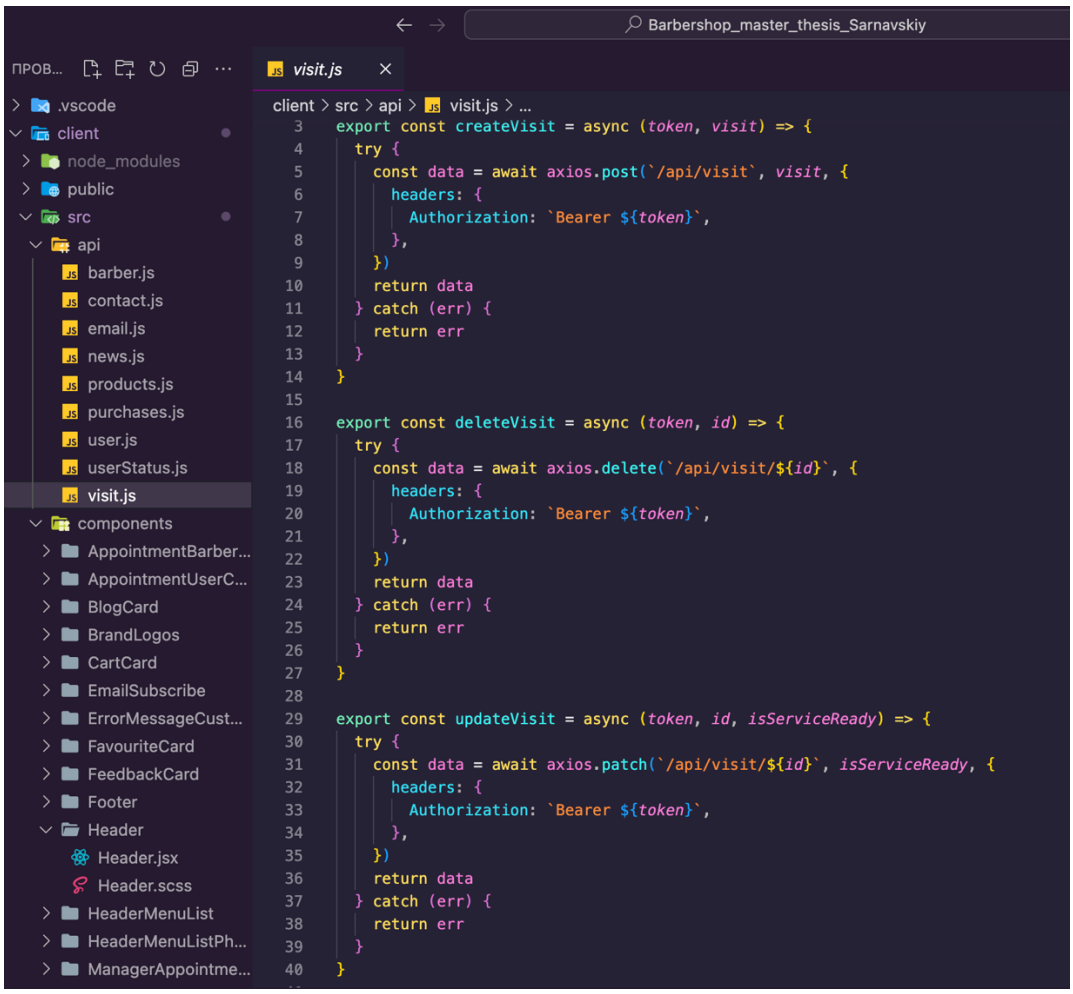

Рисунок 4.14 – Api-файл для запитів щодо візитів

Останнім елементом архітектури є папка store – є головною в обробці даних, адже є глобальним станом для компонентів в React. Бібліотека Redux забезпечує створення цього глобального стану в JavaScript додатках [20].

На рис. 4.15 продемонстровано загальний файл store з налаштуваннями, до якого підключаються інші незалежні компоненти store (reducer та action), які зображені на рис. 4.16.

Має наступні основні компоненти:

– store – центральне місце зберігання даних;

– actions – об’єкти з інформацією змін у системі;

– reducer – змінюють стан та обробляють дії;

– dispatch – метод надсилання дій в reducer;

– middleware – функції, які виконуються перед reducer.

```
const syncFavouriteListLS = (store) => {
  return function (next) {
    return function (action) {
      if (
        action.type === "ADD_FAVOURITE" ||
        action.type === "REMOVE_FAVOURITE"
      ) {
        const result = next(action) // starts reducer
        localStorage.setItem(
          "myFavorites",
          JSON.stringify(store.getState().favouriteList)
        )
        return result
      }
      return next(action)
    }
  }
}

export const reducer = combineReducers({
  news: newsReducer,
  products: productsReducer,
  barber: barberReducer,
  visit: visitReducer,
  cart: cartListReducer,
  favourite: favouritesListReducer,
})

const devTools = window.__REDUX_DEVTOOLS_EXTENSION__
  ? window.__REDUX_DEVTOOLS_EXTENSION__()
  : (f) => f

const store = createStore(
  reducer,
  compose(applyMiddleware(thunk, syncCartListLS, syncFavouriteListLS), devTools)
)
```

Рисунок 4.15 – Файл store.js

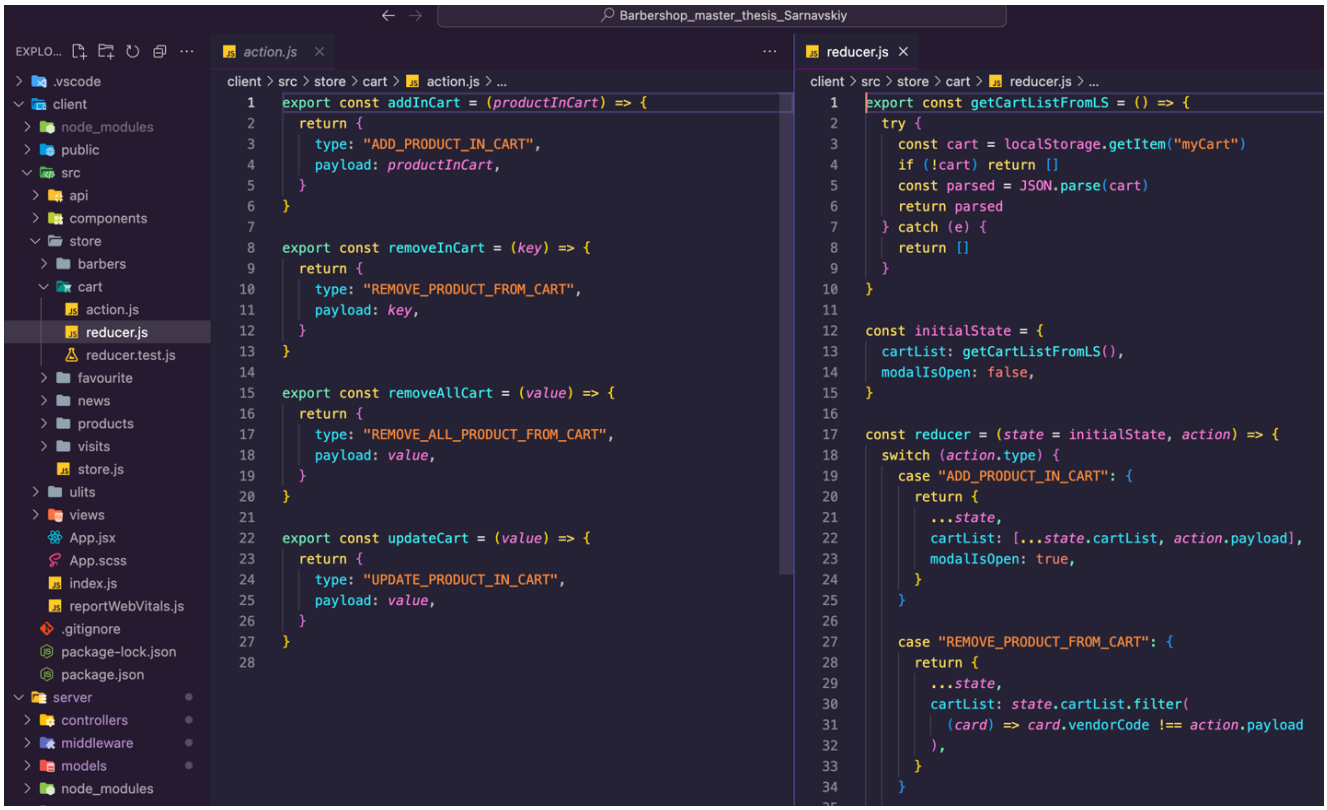

Рисунок 4.16 – Файли action.js та reducer.js

4.3.3 Функціонал та інструменти клієнтської частини

В клієнтській частині також використовуються інші бібліотеки та інструменти. Наприклад, Formik, що є бібліотекою та допомагає спростити взаємодію з формами. Завдяки їй можна легко створювати та контролювати форми.

Разом з Formik використовується та інтегрується бібліотека Yup. Yup – це ефективна бібліотека для React, що валідує та порівнює дані з заданими вимогами та правилами. Дозволяє робити опис простими словами та при помилках вивід результату помилки. Використання даних бібліотек в компоненті додавання новини менеджером, продемонстровано на рис. 4.17.

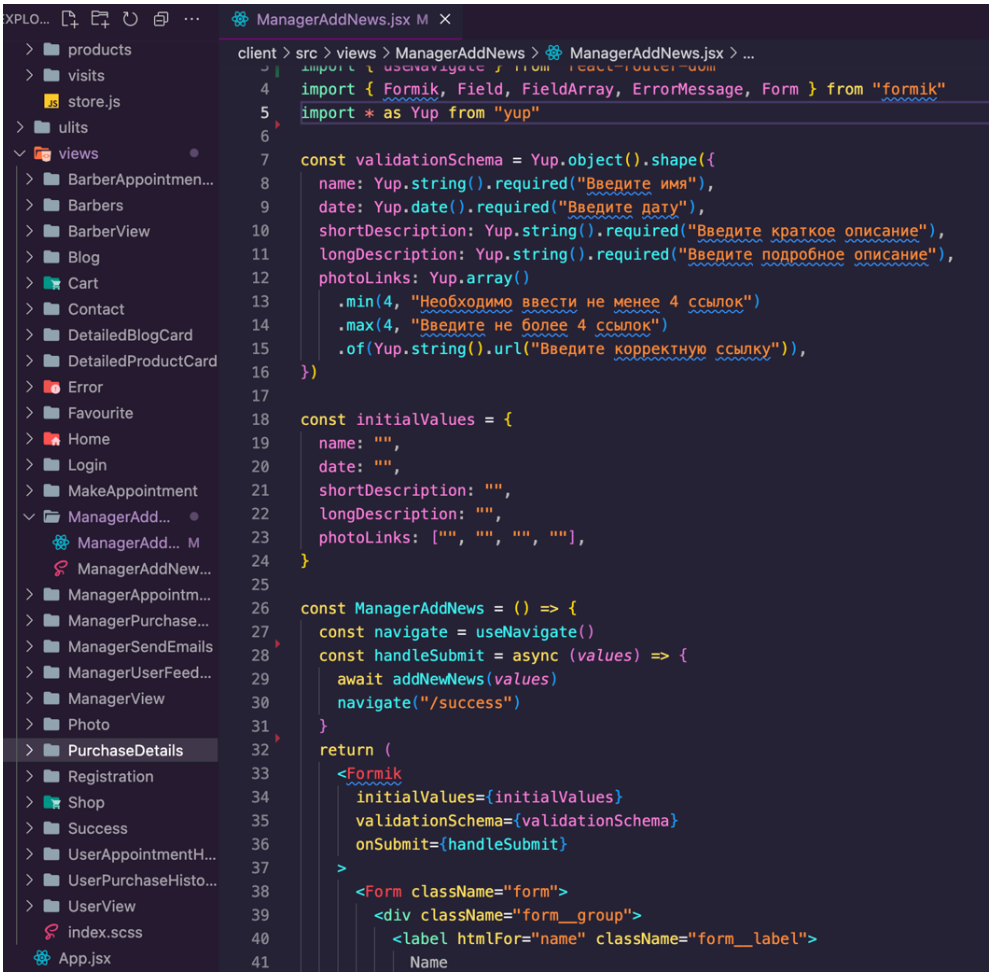

Рисунок 4.17 – Використання бібліотек Formik та Yup у файлі

Також, була використана бібліотека Material-UI. Дана бібліотека є популярною на сьогодні для створення інтерфейсу користувача. Головна

особливість - Material-UI надає готові рішення стилізованих компонентів, що робить процес розробки більш продуктивним, швидким та ефективним [21].

Завдяки Material-UI були використані такі компоненти, як іконки (кошика, улюблених товарів) та індикатор завантаження у компонентах Shop й Blog. Індикатор завантаження спрацьовує через прапорець в store, коли дані ще не надійшли з серверу. Приклад використання компонента Material-UI у вигляді індикатора завантаження наведено на рис. 4.18.

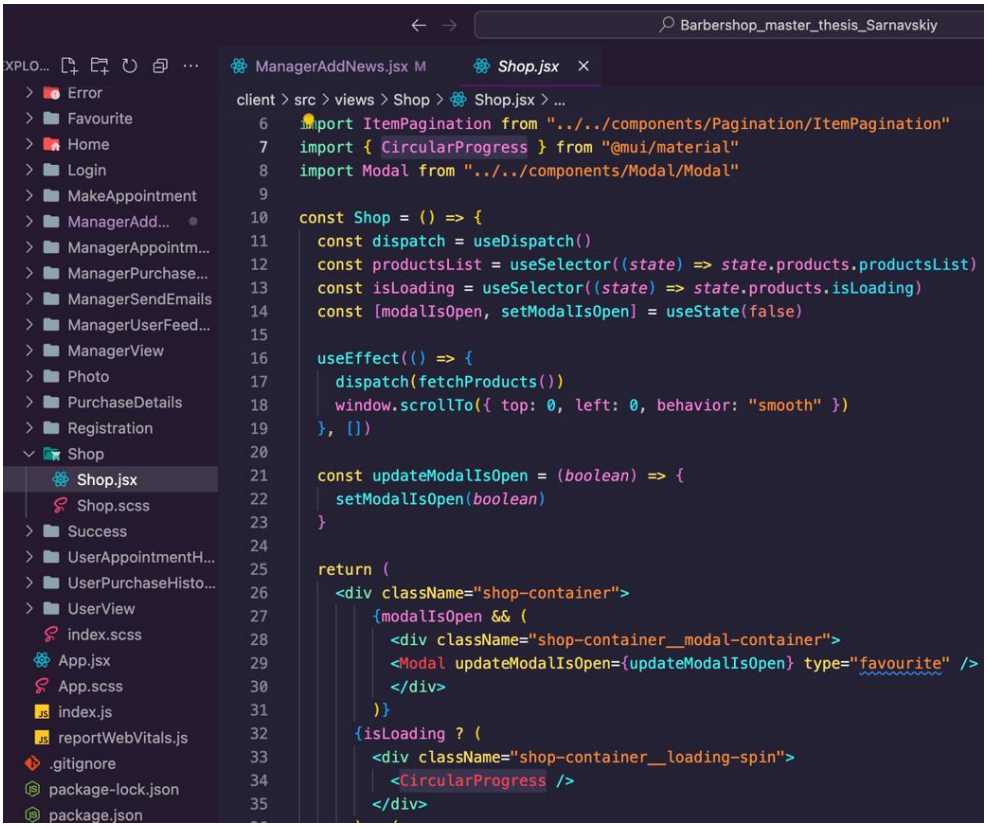

Рисунок 4.18 – Використання компоненту індикатора завантаження Material-UI

Для зберігання унікального токена, який давався після авторизаціїі, використовувався localStorage. LocalStorage – сховище для веб-ресурсів, яке знаходиться у браузерах і зберігає пари ключ-значення у локальному сховищі. Він має невелику місткість, але завдяки цьому інструменту не потрібно постійно звертатись до серверу за даними.

Також наявний алгоритм для визначення доступного часу на послуги барбера. Алгоритм продемонстрований на рис. 4.19, а схема роботи – у додатку Е.

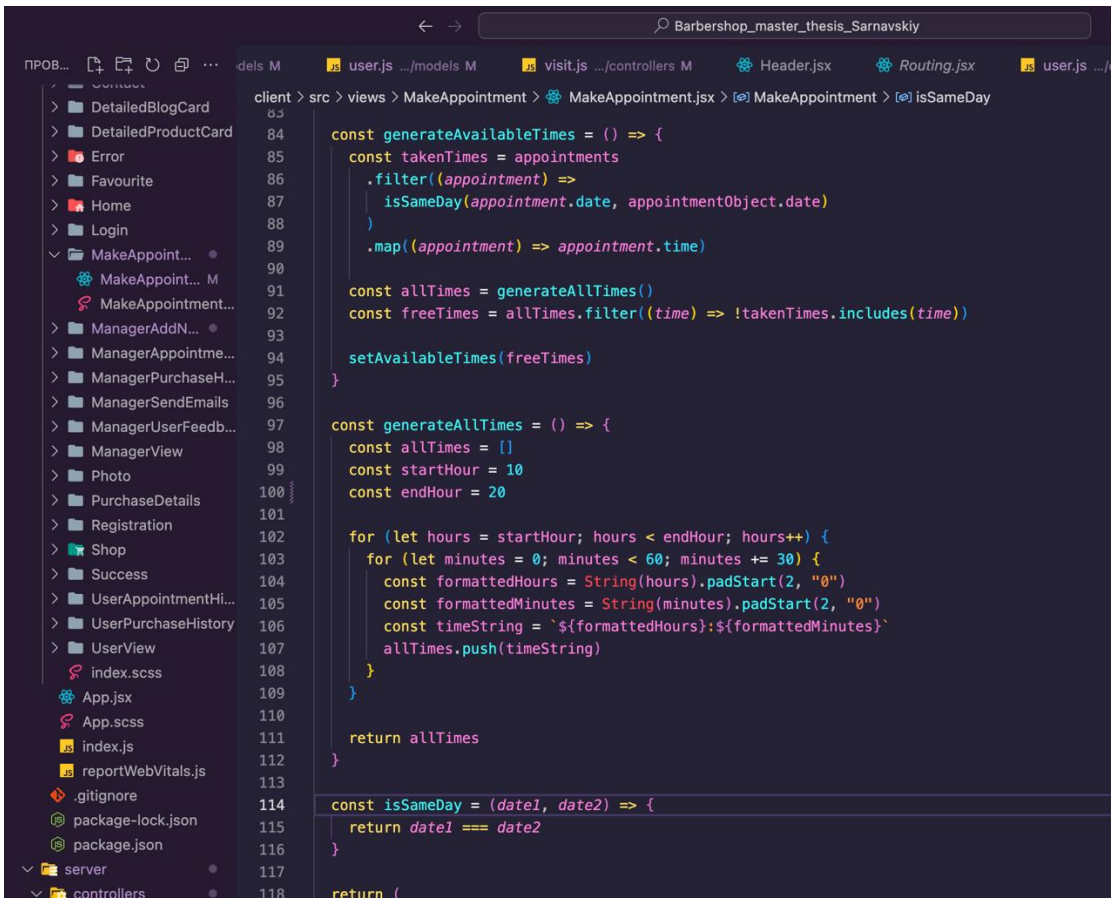

Рисунок 4.19 – Алгоритм визначення доступного часу на послугу

В наведеному алгоритмі наявні наступні дії:

– generateAllTimes() – функція для генерації часу з інтервалом по 30 хвилин між годинами роботи закладу (від 10:00 до 20:00). Набір часів форматується у рядок «години:хвилини» та надсилається до масиву allTimes;

– isSameDay() – проста функція, яка порівнює дві дати і повертає булеве значення (якщо однакові – true, якщо різні – false);

– generateAvailableTimes() – функція використовує іншу функцію isSameDay(), яка порівнює дві дати (одну яка приходить з серверу та іншу, яку вказав клієнт). Далі фільтрує це все, витягуючи ті часи й зберігає в масиві takenTimes. На кінець викликається функція setAvailableTimes(), яка і встановлює дані часи в компонент для запису.

Щоб при замовленні товарів, створенні візиту чи розсилки менеджера на електронну пошту приходила потрібна інформація про зроблену дію, використовувалась бібліотека Email.js. Застосування бібліотеки можна побачити на

рис. 4.20. Дана бібліотека відправляє повідомлення з клієнтської частини. Пошта відправника sarnavsk2001@gmail.com – вказана в налаштуваннях бібліотеки. Також підтримує різні шаблони для різних повідомлень, які можна налаштувати в персональному кабінеті бібліотеки (рис. 4.21).

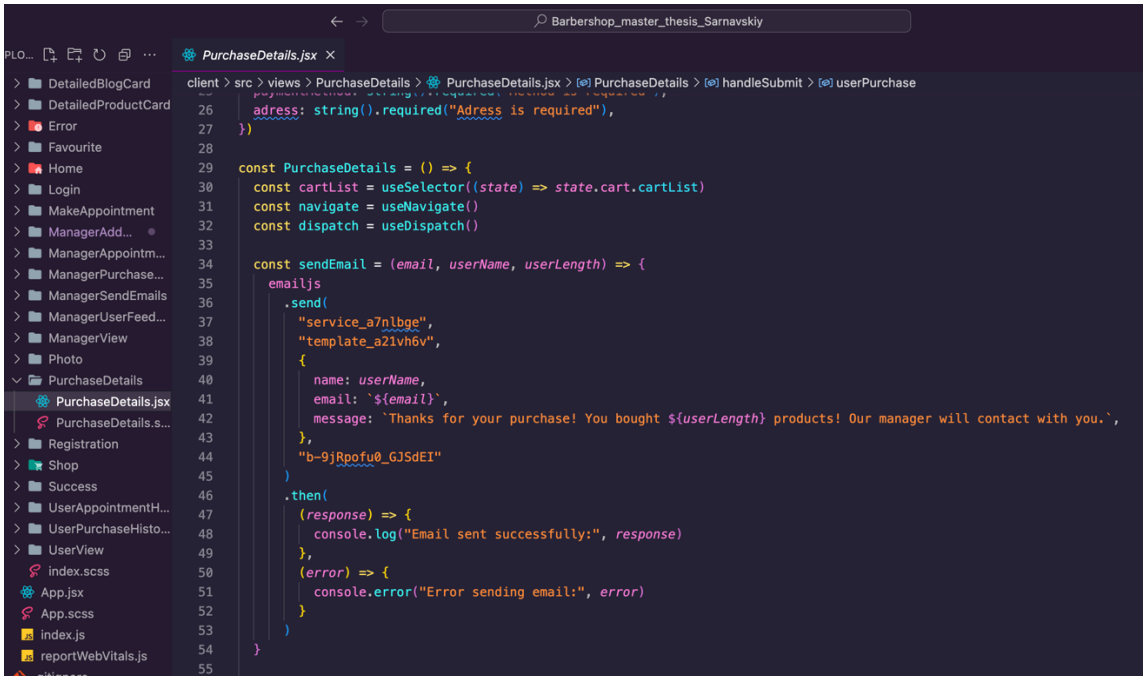

Рисунок 4.20 – Використання email.js для відправки електронних повідомлень

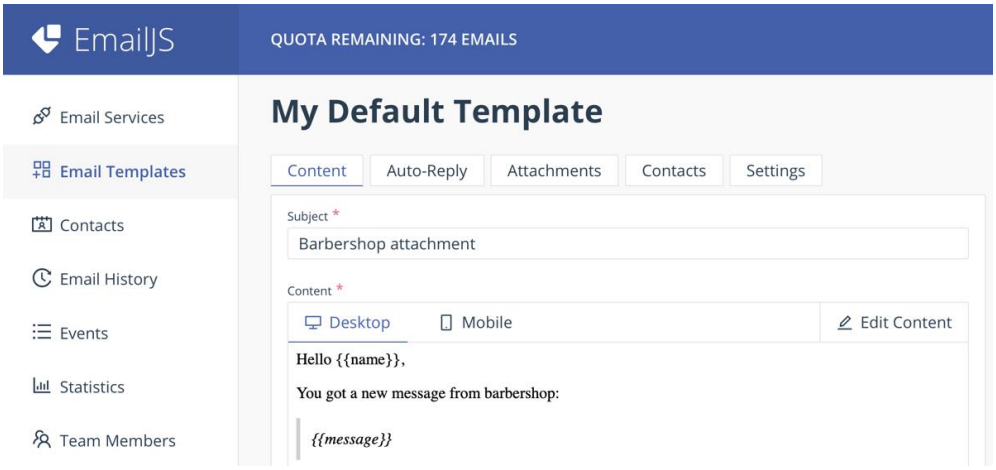

Рисунок 4.21 – Налаштування в кабінеті email.js шаблонів для відправки

Загальна схема зв'язку між мережею барбершопів та клієнтами через розроблену інформаційну систему зображена у додатку Ж, а схема рекламно-маркетингових засобів у розробленій інформаційній системі у додатку И.

4.4 Структура роботи розробленої інформаційної системи

Інформаційна система реалізована на стеці технологій «MERN» (MongoDB, Express.js, React, Node.js). Структурна схема інформаційної системи підтримки діяльності мережі барбершопів продемонстрована у додатку К, а посилання (QR-код) на репозиторій Git у додатку Ж.

Клієнтська частина створена на бібліотеці React, яка гарантує швидкодію, масштабованість, ефективність, плавність у створенні інтерфейсу. Використовується компонентний підхід побудови, де компоненти – це основні будівельні одиниці, які розділяють функціонал та можуть перевикористовуватись для оптимізації та продуктивності розробки.

Для управління станами використовуються локальні стани (в кожному компоненті) або бібліотека Redux, яка зберігає стан системи централізовано, а зміни вносяться через дії. Redux полегшує відстеження й управління станами у великих додатках, в той час, як локальні кращі у випадках, коли не потрібне централізоване управління.

Далі використовується API, яке допомагає взаємодіяти з сервером інформаційної системи, щоб надіслати або отримати дані, викликати серверні функції тощо.

Сам процес надсилання HTTP-запитів здійснюється завдяки бібліотеці Axios. Дана бібліотека допомагає спростити роботу з серверною частиною, надаючи зручний інтерфейс, який ефективно виконує запити та обробляє відповіді із запитів.

Серверна частина реалізована в середовищі виконання Node.js, яке дозволяє створювати веб-додатки на стороні сервери, завдяки мові JavaScript (яка також використовується для створення клієнтської частини).

В Node.js застосовується фреймворк Express, що допомагає спростити реалізацію та API. Фреймворк відповідає за маршрутизацію та обробку запитів від клієнтської частини. API-маршрути визначають шлях між URL та контролером, який повинен обробити даний запит.

Обробка запиту перед потраплянням до контролера, здійснюється завдяки мідлварам (посередниках). Вони використовувались для перевірки авторизації та ролі користувача.

Після посередників, запит попадає на визначений контролер, який вже містить в собі основну логіку обробки запиту. Контролери взаємодіють з моделями, обробляючи запит, та дають відповідь.

Бібліотека Mongoose – це міст між Node.js та MongoDB, який спрощує CRUD операції в базі даних. База даних має в собі колекції, які забезпечують швидкість реакцій на запити. Вони групують в собі документи за спільними характеристиками. Документи, в свою чергу, це JSON-об’єкти, які можуть мати різну структуру, поля та значення. Такий підхід допомагає MongoDB бути гнучкою, продуктивною, швидкою та ефективною.

4.5 Висновки до розділу 4

Використання стека технологій «MERN» для реалізації інформаційної системи для мережі барбешопу є обґрунтованим. Адже застосовують всі популярні, ефективні та оптимальні технології для створення системи.

MongoDB – забезпечує гнучкість та масштабованість структури даних, Express – дозволяє ефективно проводити обмін даних та забезпечує логіку серверної частини, React – відображає простий у розумінні та швидкий в оновленні інтерфейс для клієнту, а Node.js – поєднує всі частини стека разом та забезпечує продуктивну розробку серверної частини.

5 ОГЛЯД РОЗРОБЛЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

5.1 Системні вимоги та інсталяція

Інформаційна система є веб-застосунком, тому не потребує інсталяції (встановлення) на пристрій і має такі системні вимоги:

– технічні характеристики пристрою: повинен бути сучасний процесор, оперативна пам’ять та графічний адаптер.

– операційна система – може бути будь-яка, наприклад: Windows, MacOS або Linux;

– браузер – щоб було правильне та ефективне відображення веб-застосунку, потрібно мати браузер (бажано найновішої версії, адже старіші версії можуть не підтримувати деякий функціонал та неправильно відображувати елементи системи). Щодо видів браузерів – можуть застосовуватись всі, наприклад: Google Chrome, Safari, Microsoft Edge, Mozilla Firefox тощо;

– інтернет-з’єднання – впливає на швидкість відображення, взаємодії користувача з системою, оновлення та підгрузку потрібної інформації на певних сторінках, адже в веб-застосунку використовується великий масив інформації та взаємодія з сервером. Оптимальна швидкість інтернет-з’єднання не менше ніж 60 мбіт за секунду;

– відсутність блокувальних плагінів – деякі розширення в браузері можуть блокувати функціонал веб-додатку;

– кукі – веб-застосунок вимагає увімкнення кукі для коректної роботи функціоналу;

– роздільна здатність екрану: застосунок адаптований під всі роздільна здатність, крім менше 280 пікселів, адже в такій низькій якості неможливо взаємодіяти з користувачем функціонально та візуально. Тому на такий випадок в інформаційній системі реалізована заглушка, яка нижче продемонстрована на рис 5.1.

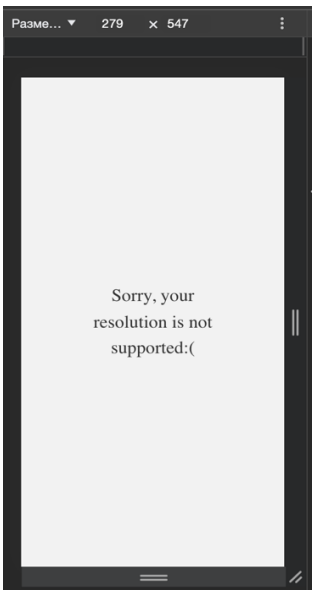

Рисунок 5.1 – Заглушка при низькій роздільній якості пристрою

5.2 Методика роботи користувача з роллю клієнта

При вході в веб застосунок, клієнт може бачити головну сторінку інформаційної системи, де його зустрічає головна інформація щодо барбешопу та послуг, мінімалістичний дизайн веб-застосунку. Продемонстрована на рис. 5.2-5.4.

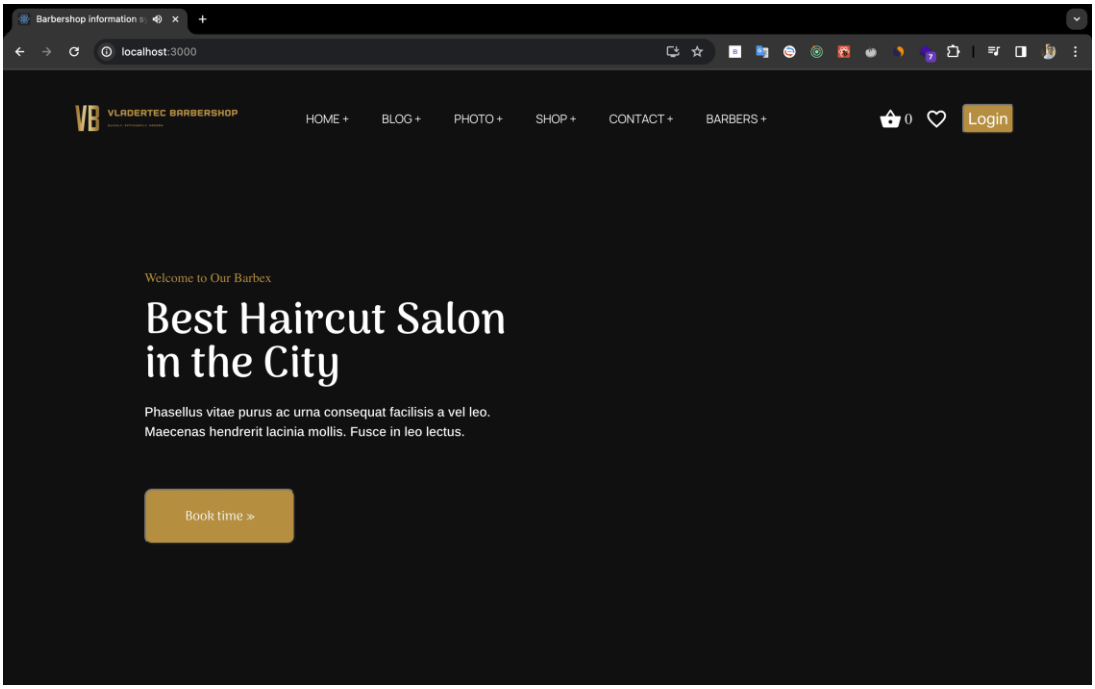

Рисунок 5.2 – Верхній колонтитул головної сторінка

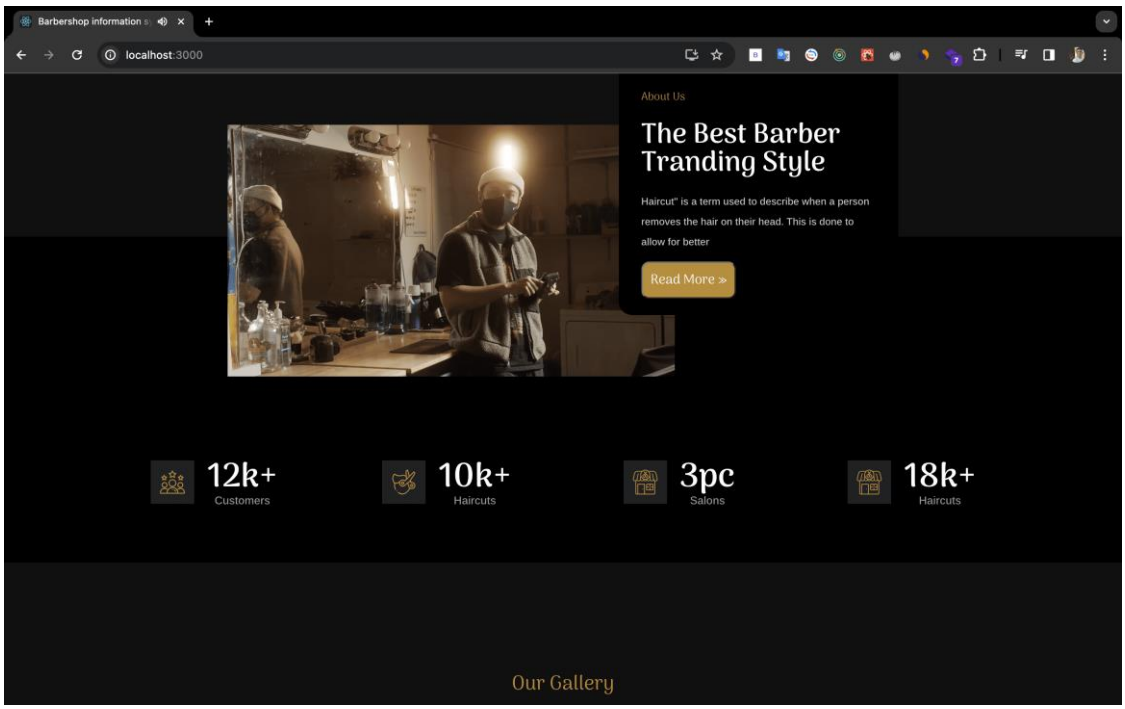

Рисунок 5.3 – Основний вміст головної сторінки

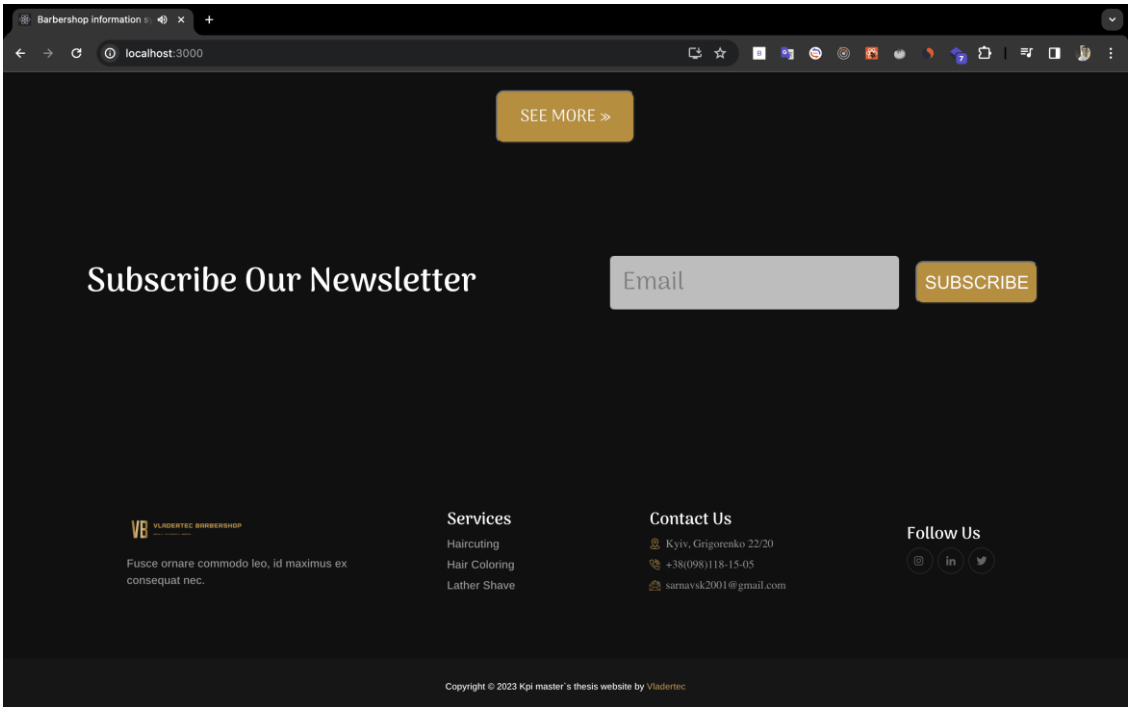

Рисунок 5.4 – Нижній колонтитул головної сторінки

Зверху веб-додатка клієнт може бачити меню вкладок, що дозволяє йому переходити до потрібної вкладки системи. Друга вкладка – це блог, де клієнт може побачити та ознайомитись з усіма останніми та актуальними новини з життя барбершопу чи в загальному зі сфери краси. Продемонстрована на рис. 5.5.

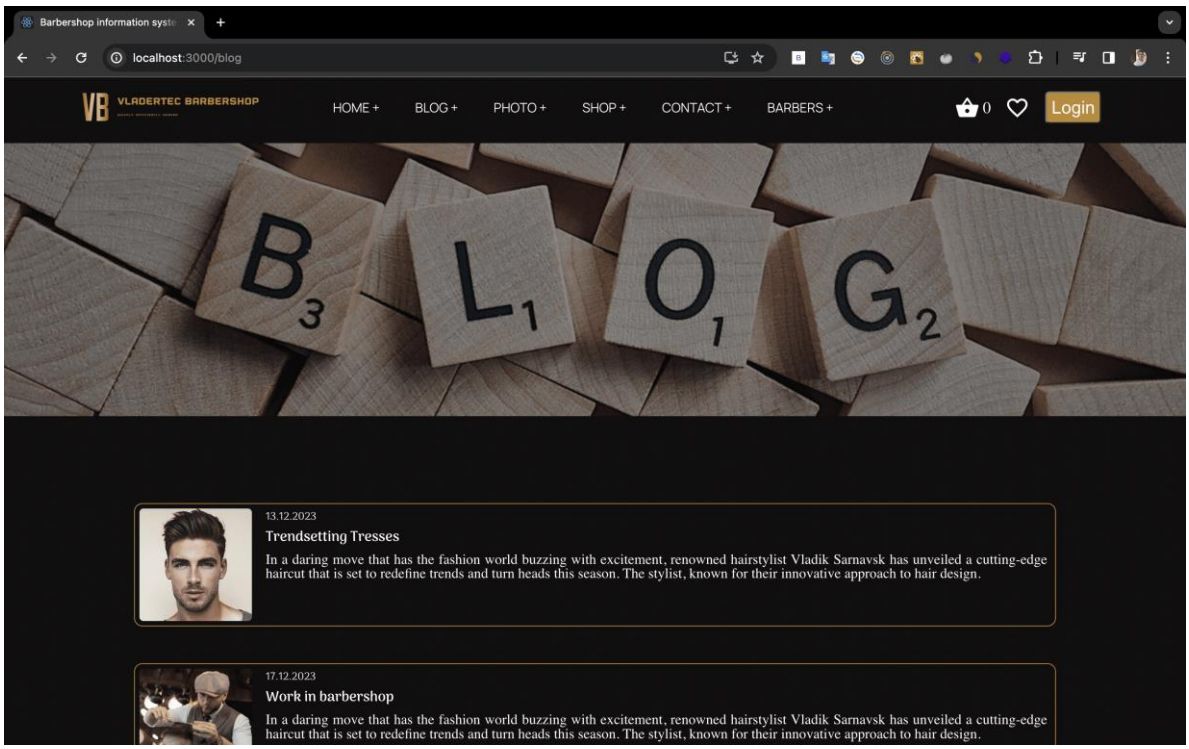

Рисунок 5.5 – Сторінка блогу з останніми новинами

Кожну картку можна переглянути, де буде більше детальна інформація та більше фотографій з новини. Продемонстрована на рис. 5.6.

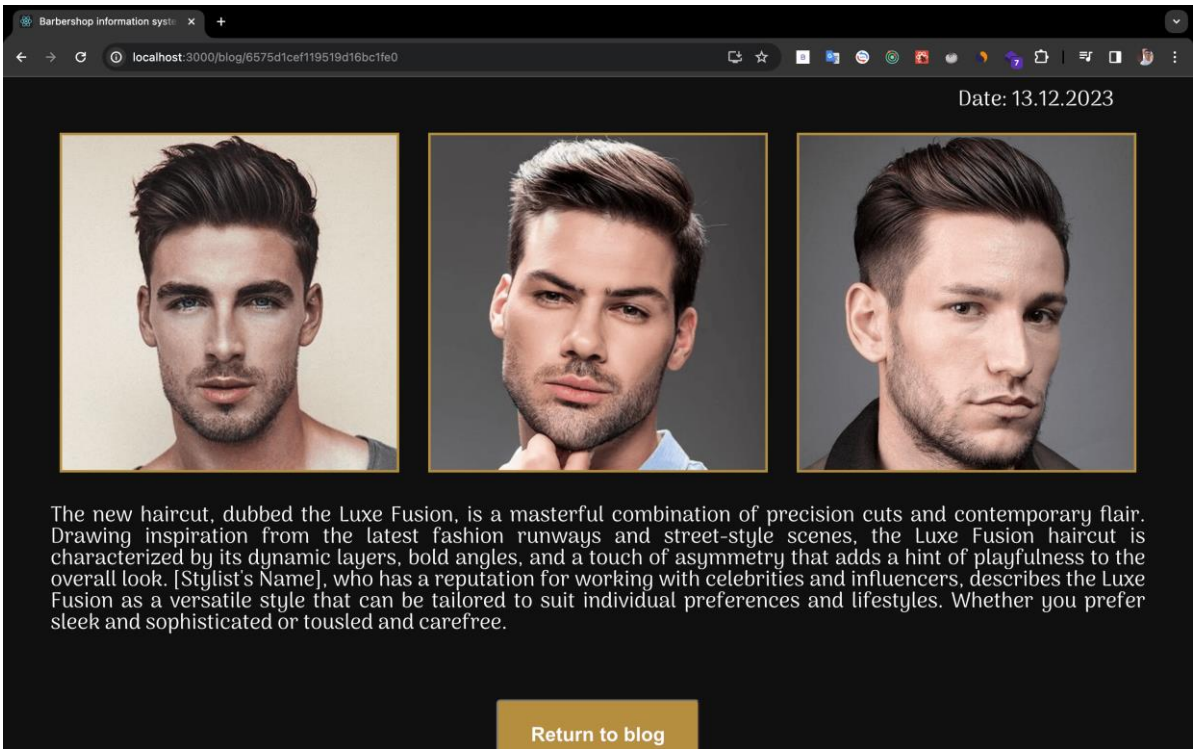

Рисунок 5.6 – Картка новини (детальний опис новини)

Наступна вкладка – це галерея, де клієнт може побачити всі фотографії зі внутрішньої роботи барбешопу та готові зачіски. Це все відображується завдяки слайдеру, через який клієнт може взаємодіяти з фотографіями та перемикати їх один за одним. Продемонстрована на рис. 5.7.

Галерея допомагає клієнту у зручності вибору, у порівнянні альтернатив (інших барбершопів) та робить взаємодію з клієнтом кращою й ефективною. Тобто, зображення в галереї є ефективним засобом маркетингу, після перегляду якого у клієнта з’являється відчуття довіри.

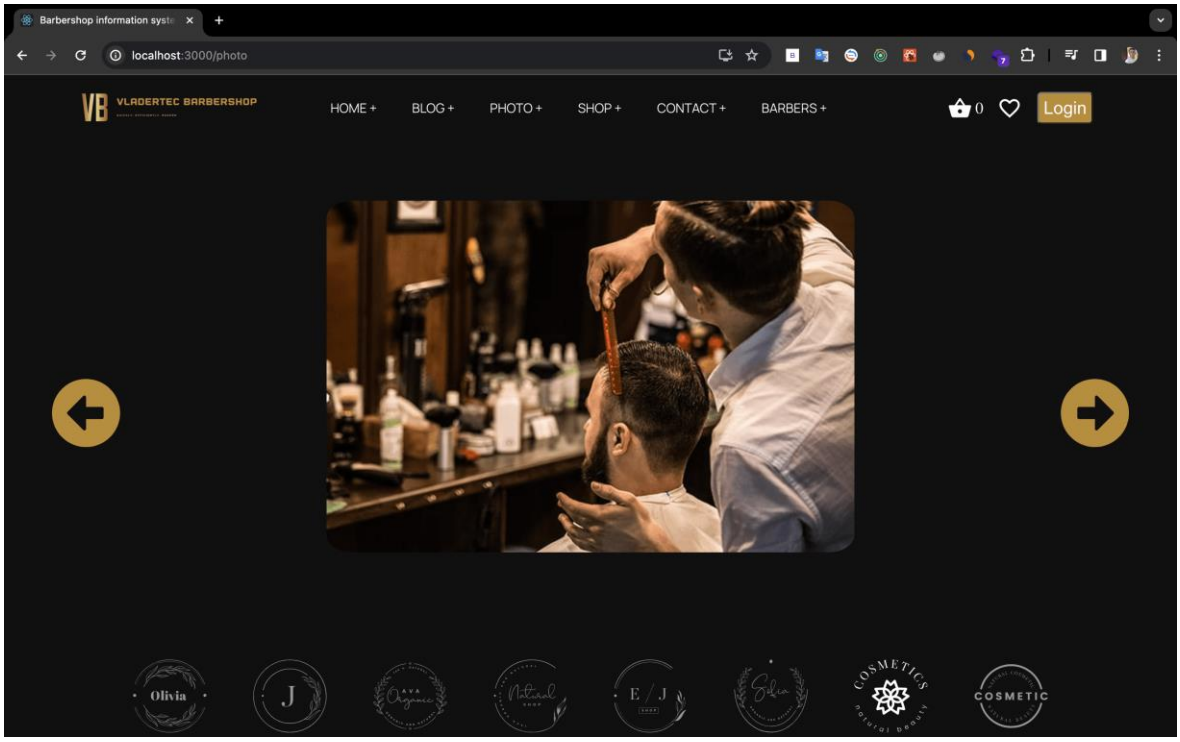

Рисунок 5.7 – Сторінка галереї

Наступна вкладка – це товари у магазині. Наявність магазину дозволяє клієнтам робити онлайн-замовлення спеціалізованих товарів догляду за красою. Це є важливим інструментом для створення у клієнтів почуття позитивного враження, збільшення продажів. Це потребує автоматизації процесу купівлі товарів через створення цих функцій і відображення товарів у інформаційній системі.

Тому сторінка забезпечує перегляд різноманітних спеціалізованих для краси товарів, їх назви, ціну та фотографію. Продемонстрована на рис. 5.8.

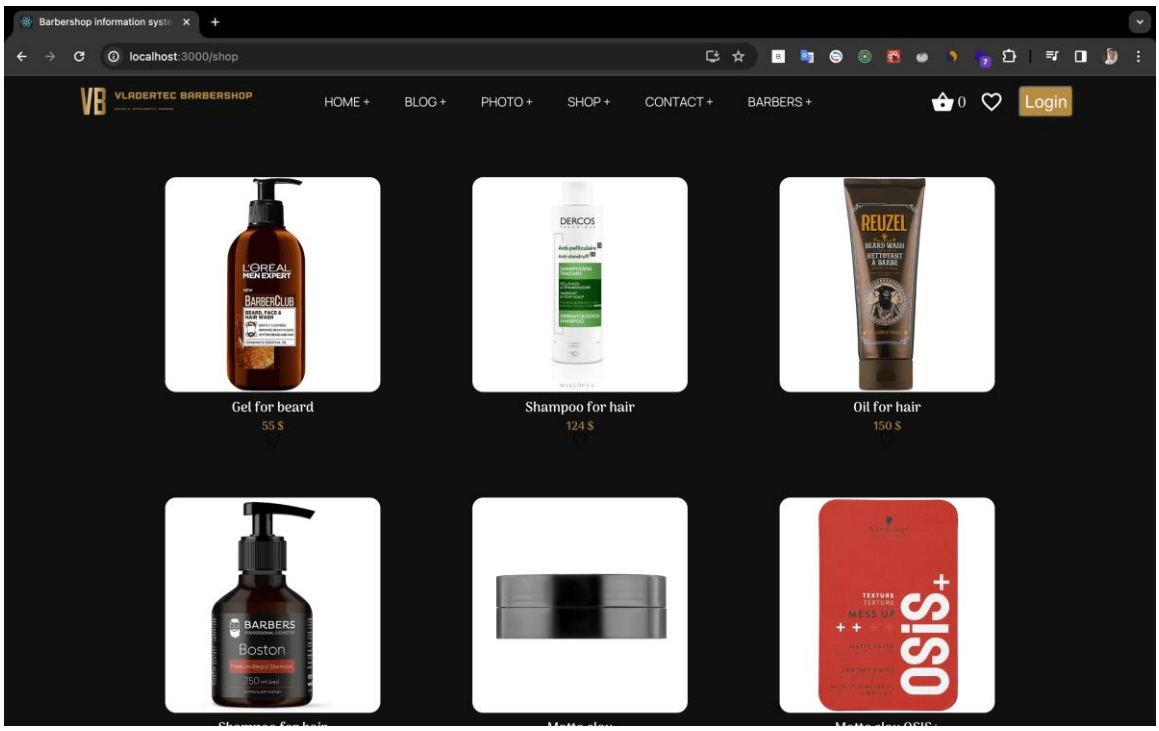

Рисунок 5.8 – Сторінка магазину

Кожну картку можна переглянути, де буде більше детальна інформація та більше фотографій з продукту. Також, клієнт може додати продукт корзину, щоб в майбутньому придбати її, або додати в улюблене. Продемонстрована на рис 5.9.

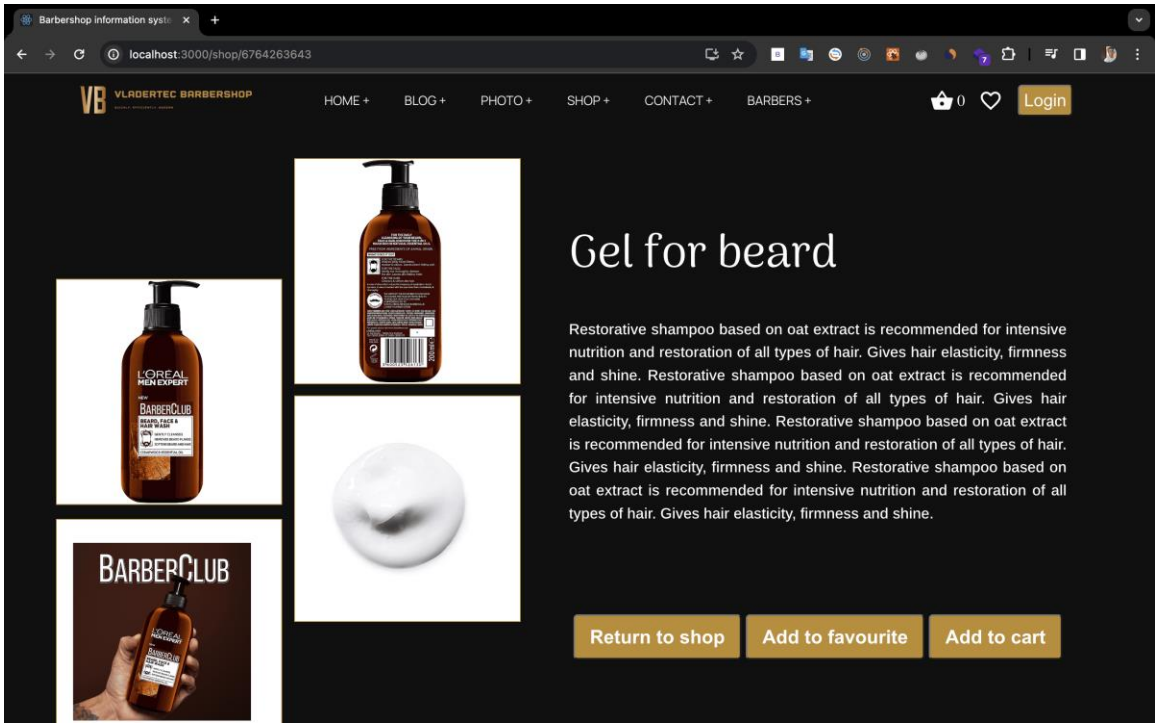

Рисунок 5.9 – Картка продукту (детальний опис продукту)

Наступні вкладки – це контакти та барбери. Завдяки ним, клієнт може ознайомитись із адресами, контактами, формою для пропозицій з керівництвом, барберами. Сторінки продемонстровані на рис. 5.10, рис. 5.11.

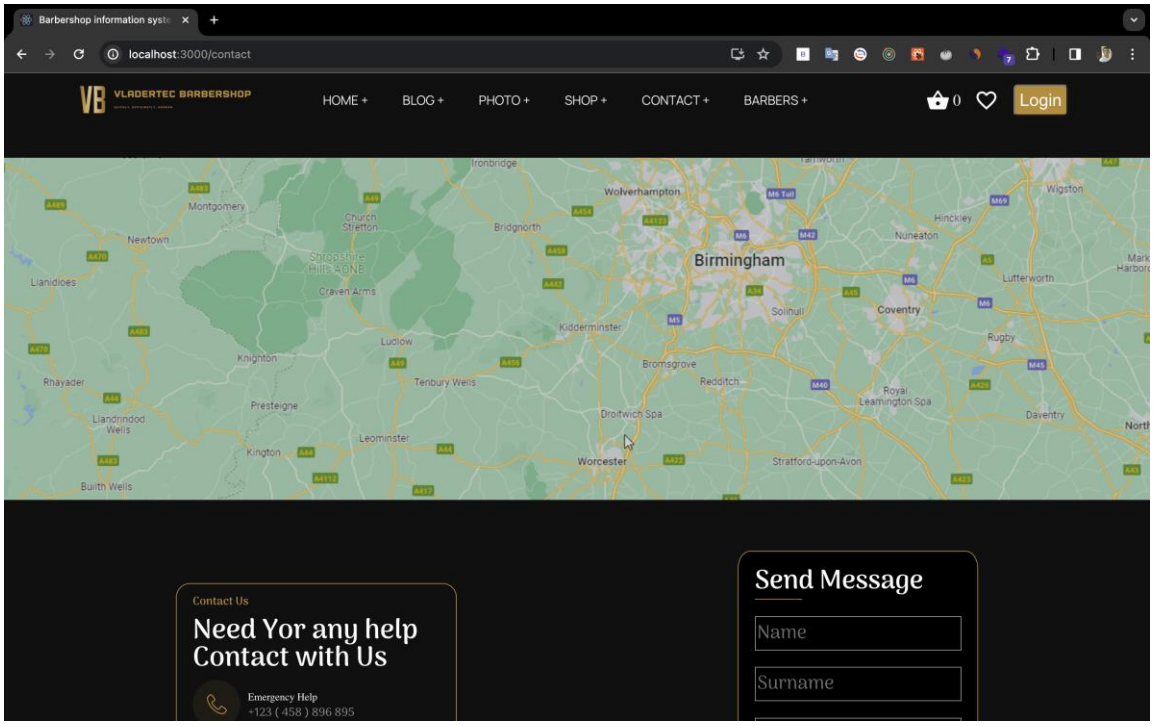

Рисунок 5.10 – Сторінка контактів та зворотного зв’язку з керівництвом

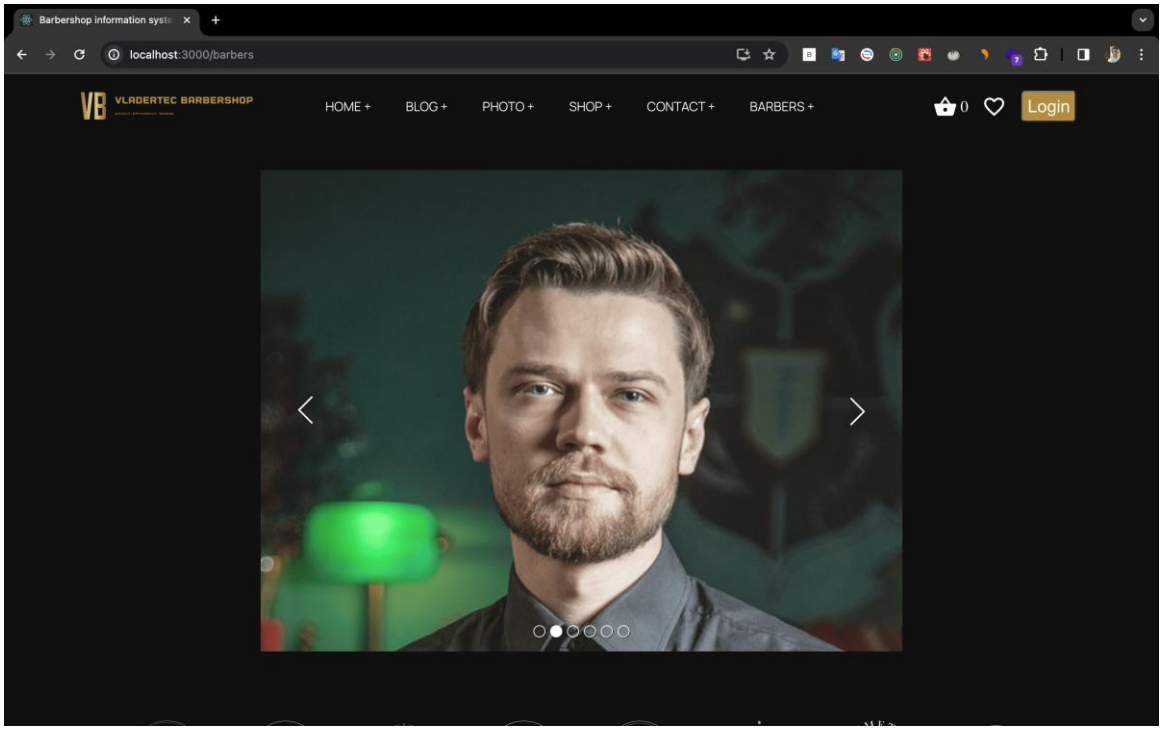

Рисунок 5.11 – Сторінка барберів

У інформаційній системі присутня автентифікація, яка захищає від несанкціонованого доступу та ідентифікує користувача, підтягуючи всю інформацію про нього та його корзину і улюблені товари.

При натисканні кнопки «Login» клієнт перекидається на сторінку і якщо він не зареєстрований може перейти на сторінку Реєстрації та зробити це (рис. 5.12).

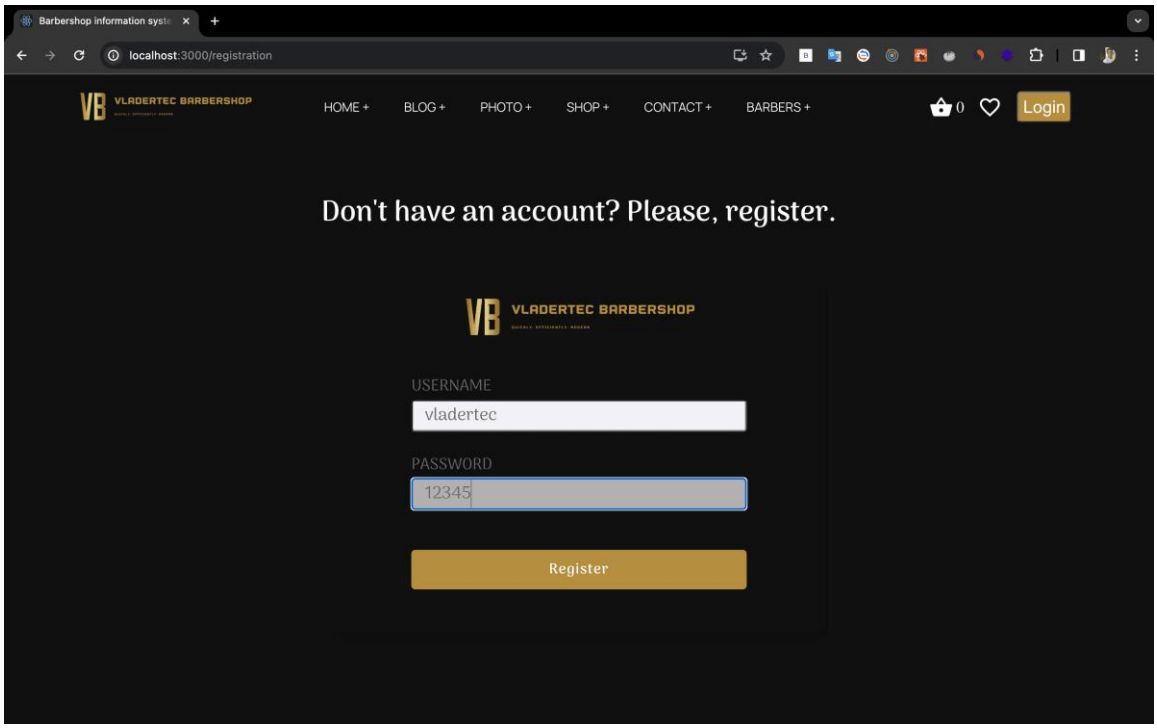

Рисунок 5.12 – Сторінка реєстрації

Після цього, клієнт з’являється в базі інформаційної системи. Далі, якщо система фіксує успішний запит, він перекидається на сторінку Логіну, де він повинен ввести коректний логін та пароль. Сторінка продемонстрована на рис. 5.13.

Якщо пароль введено невірно, то запит повертає помилку (помилка виводить від типу проблеми: невірний пароль, клієнт не зареєстрований), яка відобразиться нижче кнопки червоним коліром і клієнту потрібно знову ввести правильний логін та пароль.

При правильному паролі та логіні, кнопка «Login» змінюється на «Logout», підтягуються з бази даних вся інформація (кошик та улюблені товари, які були обрані до виходу з системи (якщо клієнт зареєструвався раніше)).

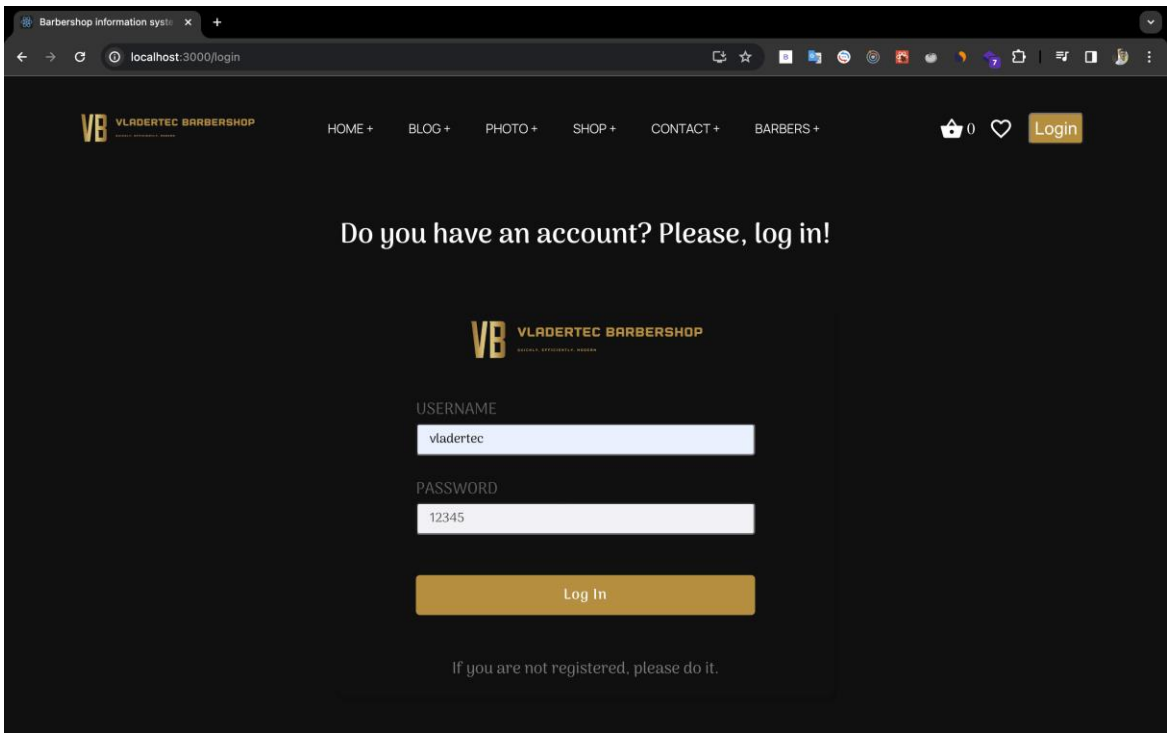

Рисунок 5.13 – Сторінка логіну

Після успішного входу, клієнт може зайти на вкладку магазин, вибрати товар, який він хоче придбати та додати його в корзину. Сторінка корзини продемонстрована на рис. 5.14.

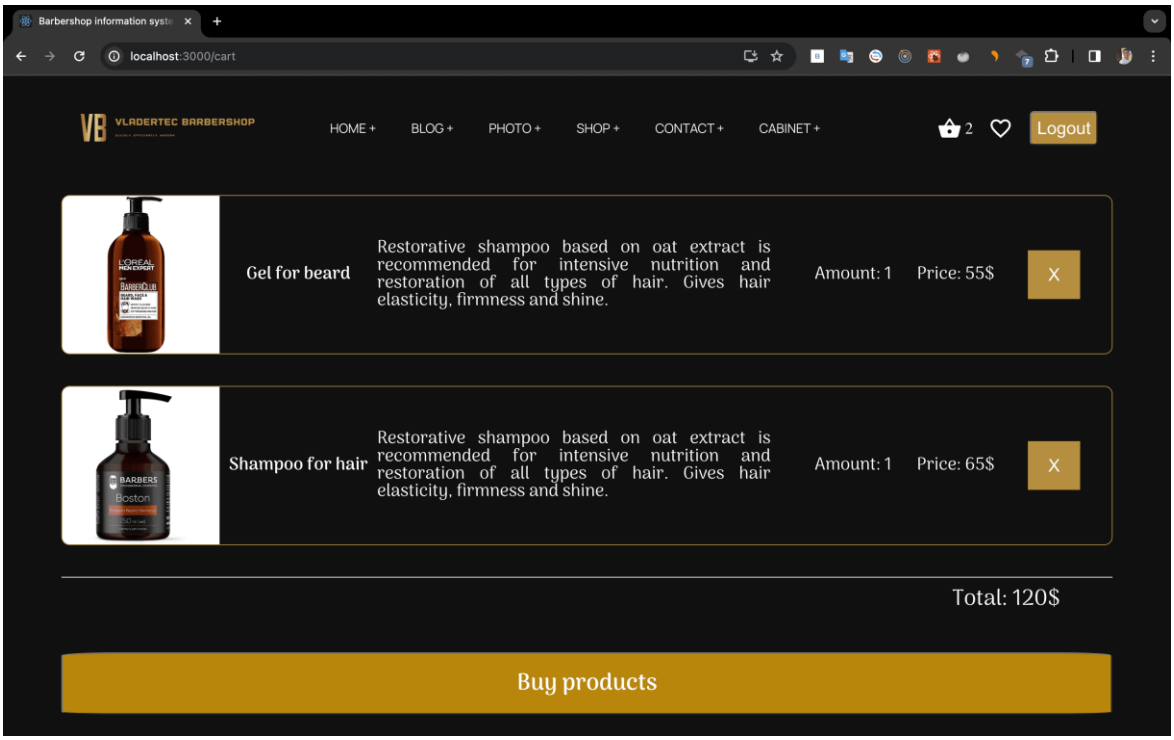

Рисунок 5.14 – Сторінка кошику

При натисканні кнопку «Buy products», клієнт пересилається до форми для замовлення, куди потрібно внести персональну інформацію. Сторінка форми продемонстрована на рис. 5.15. Вся інформація валідується на її коректність. Якщо все успішно, клієнта пересилає на головну сторінку, або на сторінку помилки. Також приходить лист на електронну пошту, яку клієнт вказав в формі (показано на рис. 5.16).

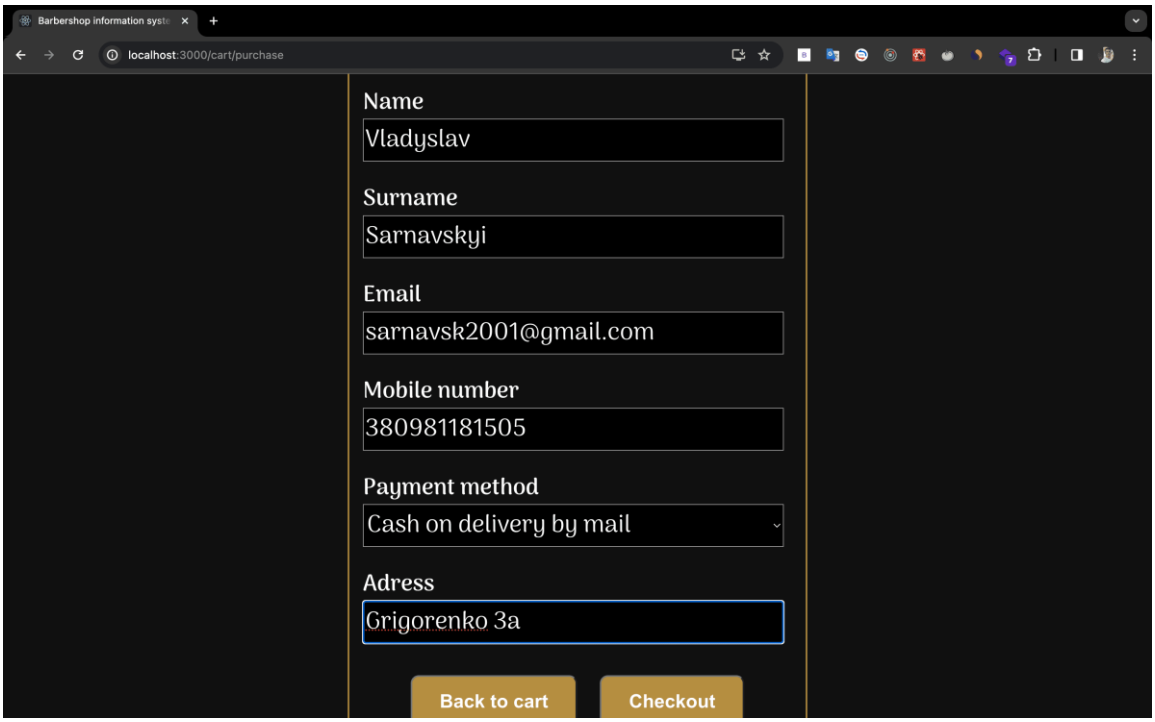

Рисунок 5.15 – Сторінка форми для здійснення покупки

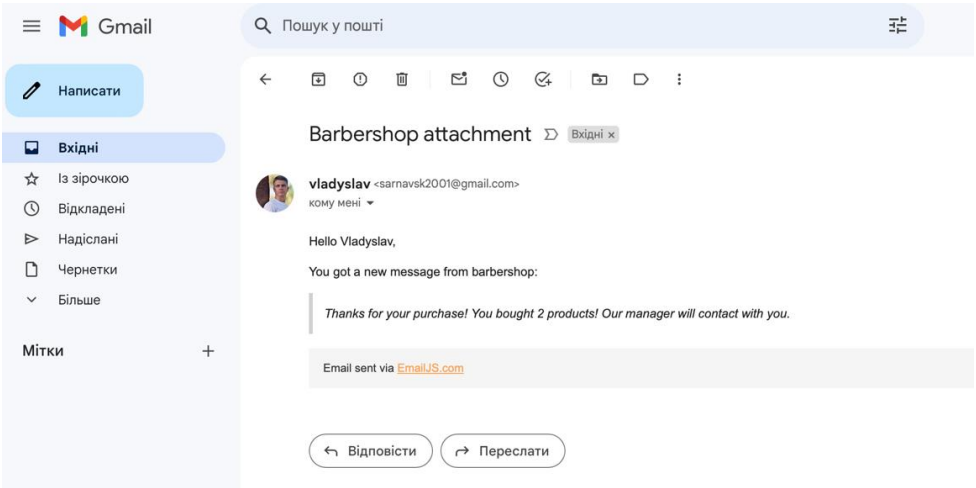

Рисунок 5.16 – Отримане повідомлення на електронній пошті клієнта

Схожий алгоритм дій має і запис на послугу. Клієнт повинен натиснути на головній (основній) вкладці кнопку «Book time». Після цього його переносить на сторінку форми створення запису, куди потрібно внести персональну інформацію вибрати барбера (ім’я та прізвище барберів беруться з бази даних), дату, час та послугу. Сторінка форми для запису продемонстрована на рис. 5.17. Вся інформація валідується на її коректність Якщо все успішно, клієнта пересилає на головну сторінку, або на сторінку помилки. Також приходить лист на електронну пошту, яку клієнт вказав в формі (показано на рис. 5.18).

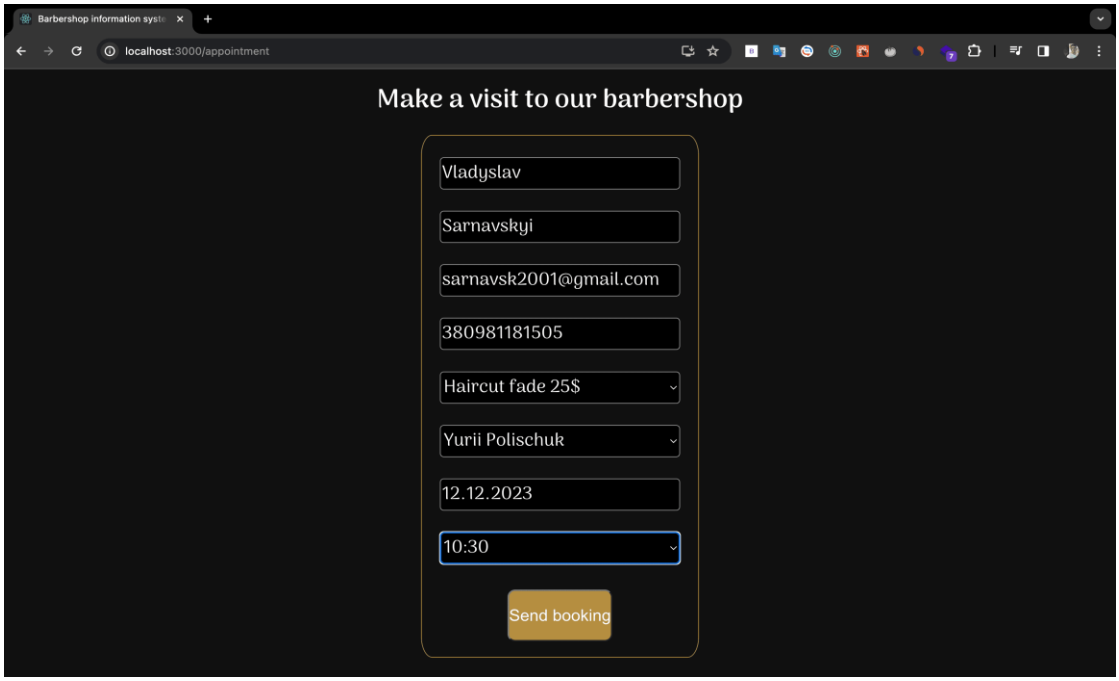

Рисунок 5.17 – Сторінка форми для запису на послугу

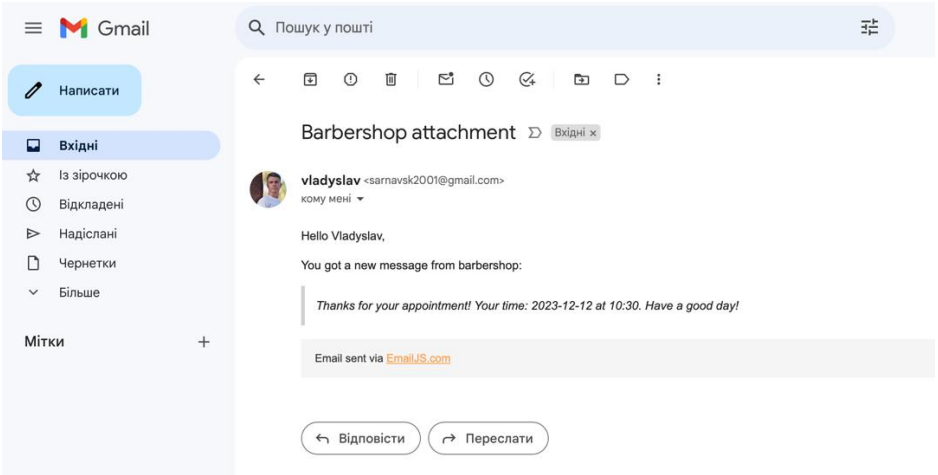

Рисунок 5.18 – Отримане повідомлення на електронній пошті клієнта

При успішній авторизації, у клієнта з’являється зверху меню вкладка «Cabinet». Натиснувши, клієнт перенаправляється в свій кабінет, де може додати чи оновити (якщо вже додана) свою інформацію. Кабінет продемонстрований на рис. 5.19. Також, клієнт може продивитись історію замовлень (рис. 5.20).

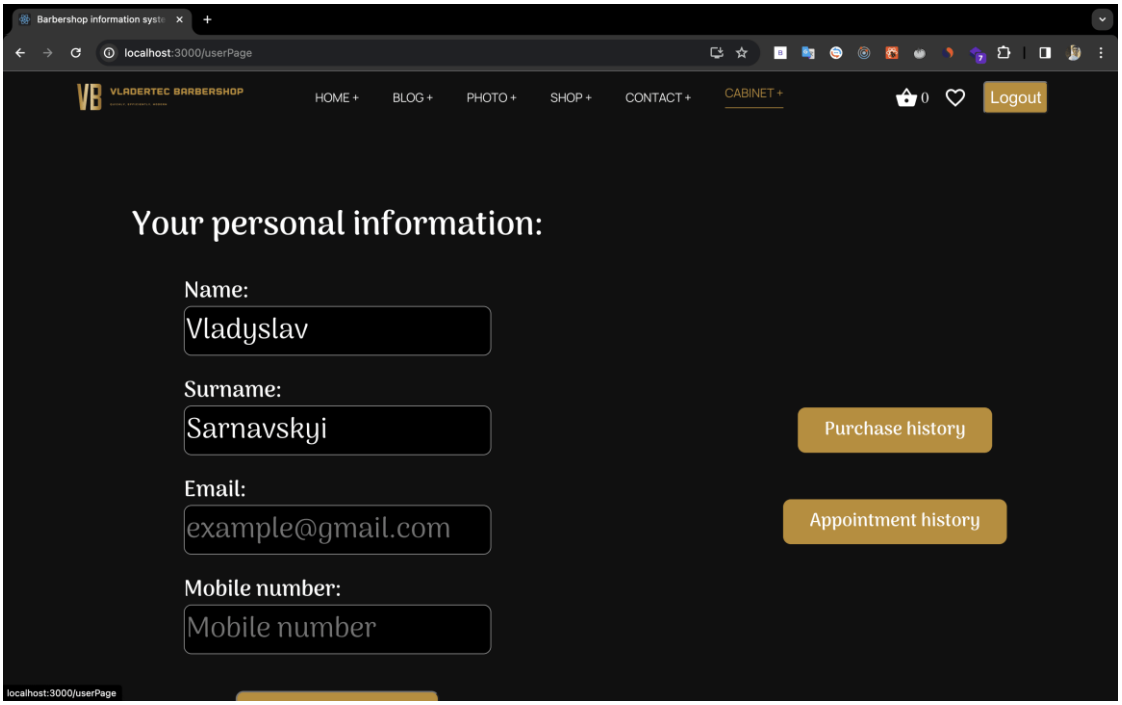

Рисунок 5.19 – Сторінка персонального кабінету клієнта

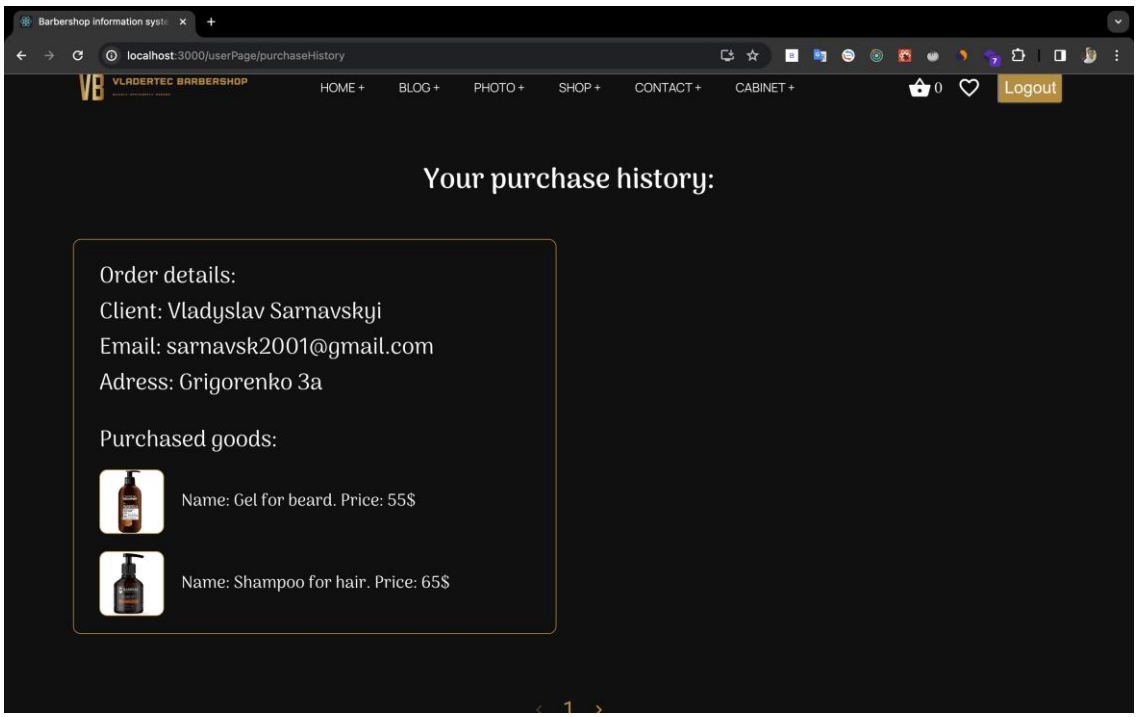

Рисунок 5.20 – Сторінка історії замовлень

Також, клієнт може продивитись свою історію візитів на послуги, сторінка якої продемонстрована на рис. 5.21. Якщо в нього немає змоги прийти на вибраний ним час, він може у будь-який момент скасувати свій візит до барбера.

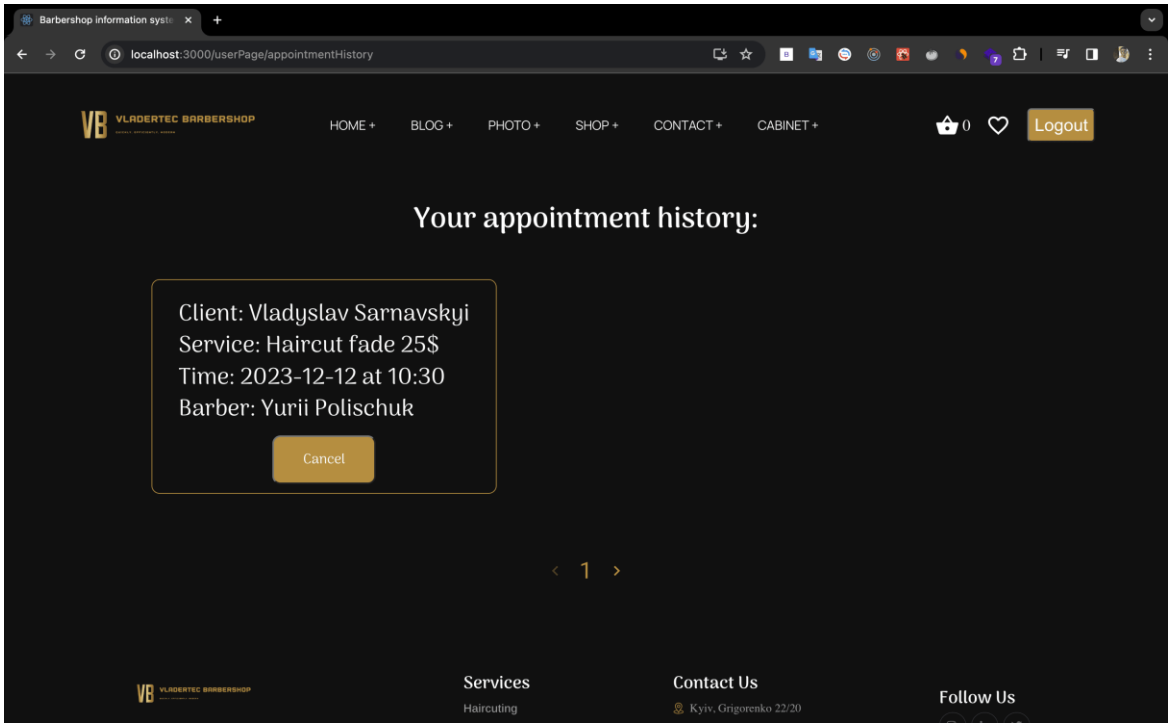

Рисунок 5.21 – Сторінка історії візитів

5.3 Методика роботи користувача з роллю менеджера

Менеджер барбершопу є ключовою фігурою барбершопу. Він забезпечує ефективність та продуктивність бізнесу, позиціонується як організатор, лице барбешопу, роботу з клієнтами, фінансами, аналізу, маркетинговими функціями закладу. Тому менеджер потребує інформаційну систему, щоб автоматизувати ці процеси. В інформаційній системі він повинен мати широкий функціонал та інтуїтивно зручний, щоб забезпечити оптимальну роботу.

В базі даних вже завчасно створений користувач з роллю адмін (менеджер). Йому не потрібно реєструватись в системі, а достатньо зразу ввести вірний логін та пароль на тій же вкладці да реєструється клієнт.

Після успішної авторизації, відобразиться кабінет для менеджера (рис. 5.22), де він зможе зареєструвати нового барбера (доступ до цього має тільки менеджер).

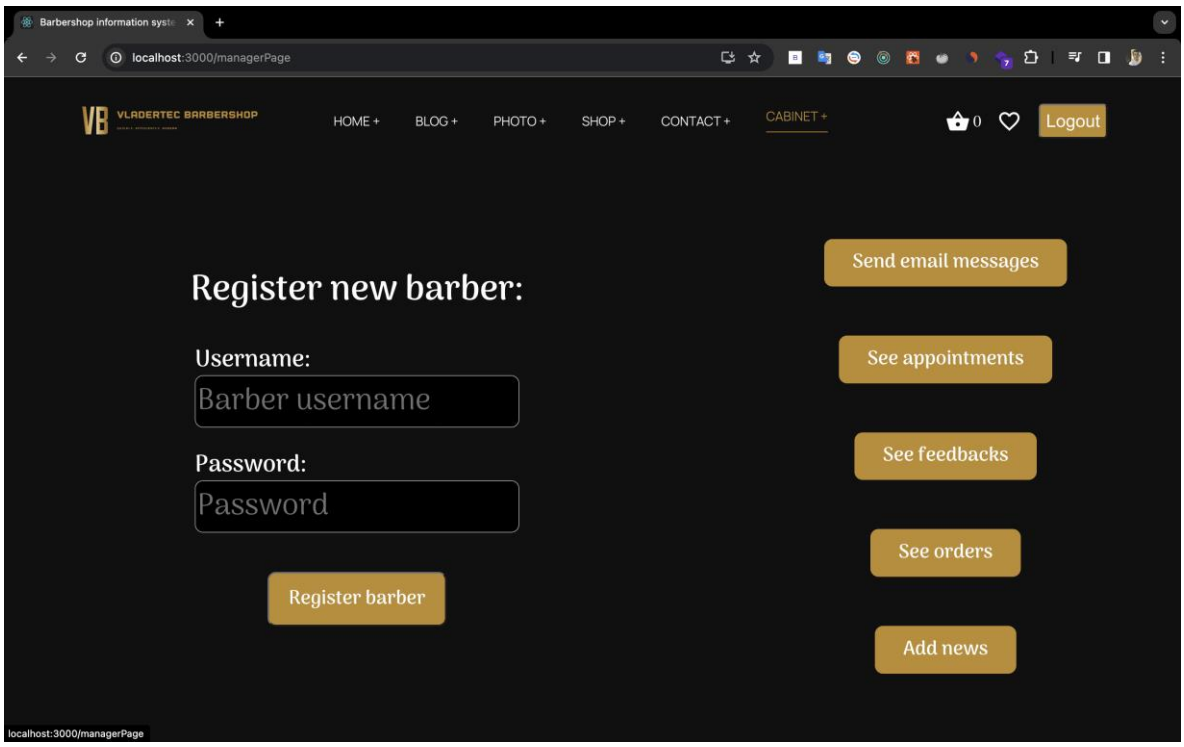

Рисунок 5.22 – Сторінка історії візитів

Менеджер в своєму кабінеті може додати новину, натиснувши на кнопку «Add news». Його зустріне сторінка додавання новини на рис. 5.23. Йому потрібно додати фото, опис, дату, посилання на фото (можна застосувати сервіс Cloudinary).

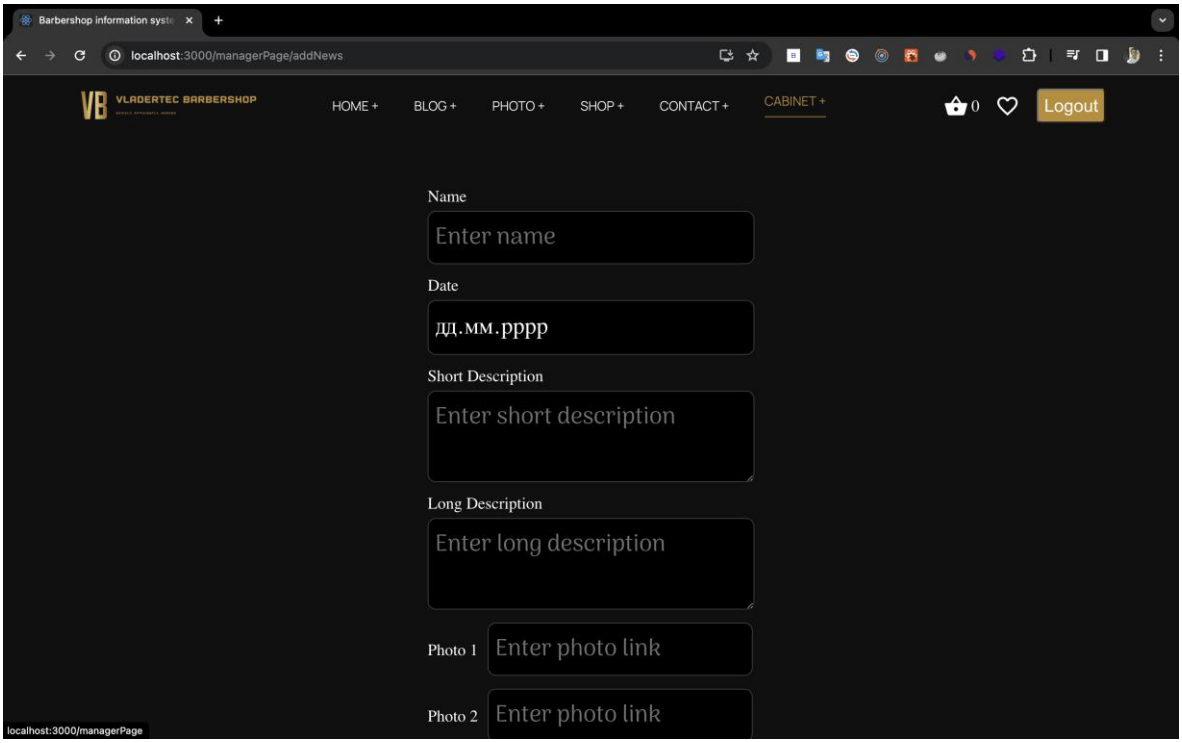

Рисунок 5.23 – Сторінка форми додавання новини

Так як на головній сторінці та блогу є спеціальні поля для вводу електронної пошти для підписки на новини, менеджер може розсилати свої повідомлення (промокоди, знижки) на ці електронні пошти. Сторінка розсилки на рис. 5.24. Результат розсилки на електронні пошти, можна побачити на рис. 5.25.

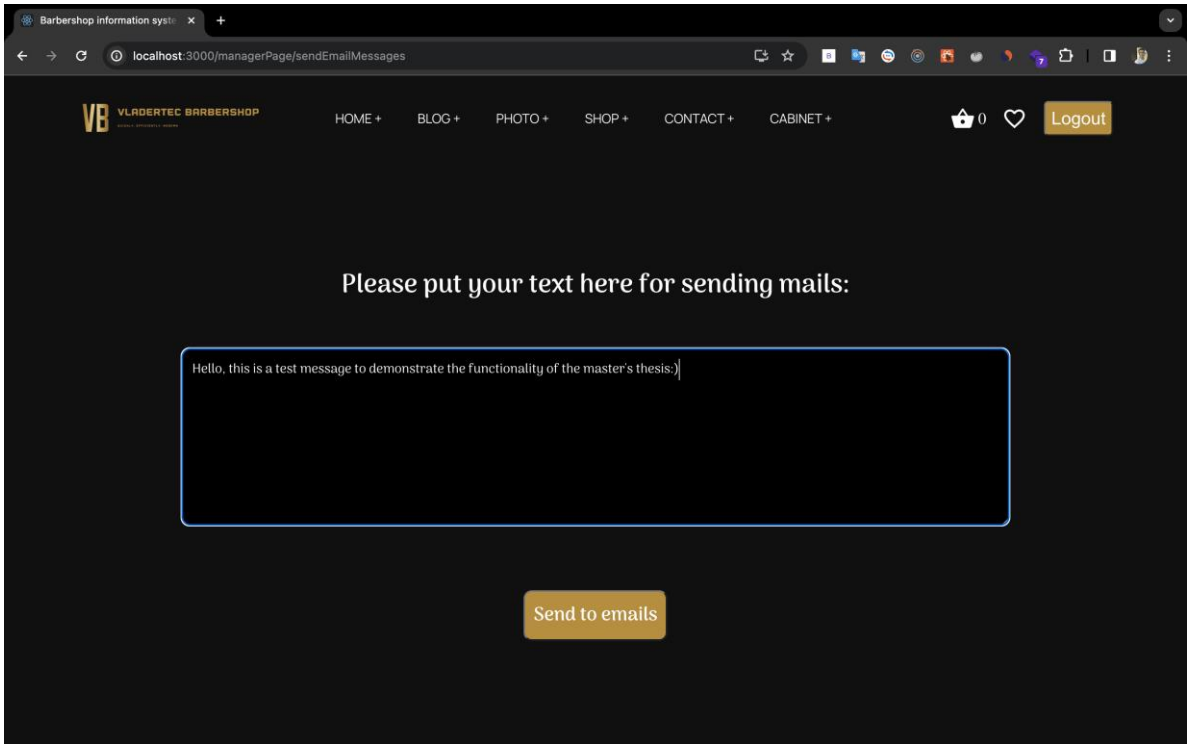

Рисунок 5.24 – Сторінка розсилки повідомлень на електронні пошти клієнтів

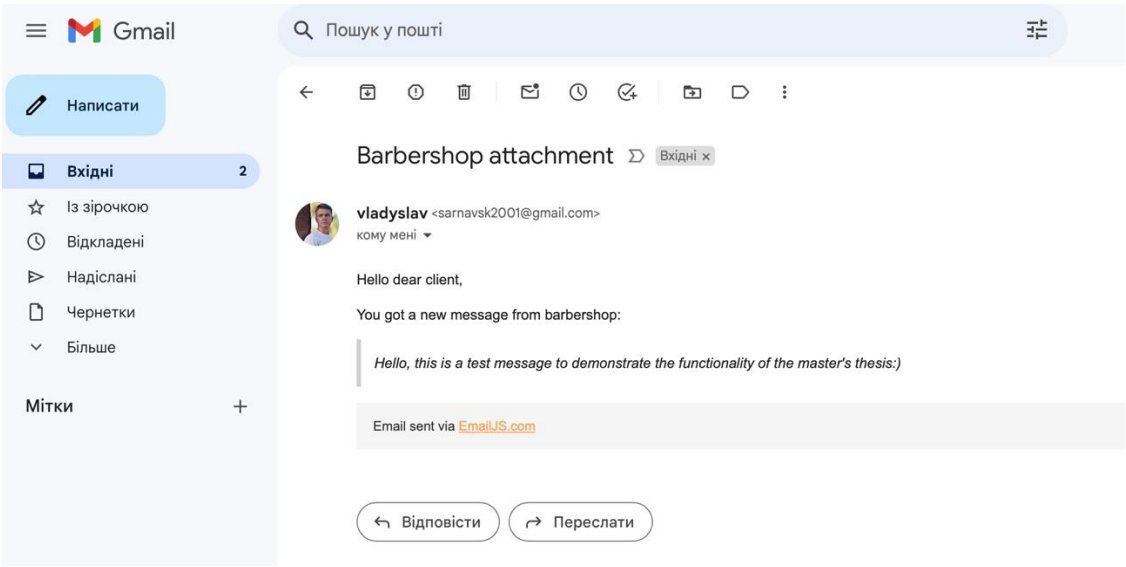

Рисунок 5.25 – Отримане повідомлення на електронній пошті клієнта

В кабінеті менеджер має змогу продивитись всі записи та історію всіх покупок, щоб оптимально створювати графік для персоналу, фінансові звіти, оновлення товару та послуг, завдяки їх аналізу. Дані сторінки продемонстровані на рис. 5.26 та на рис. 5.27.

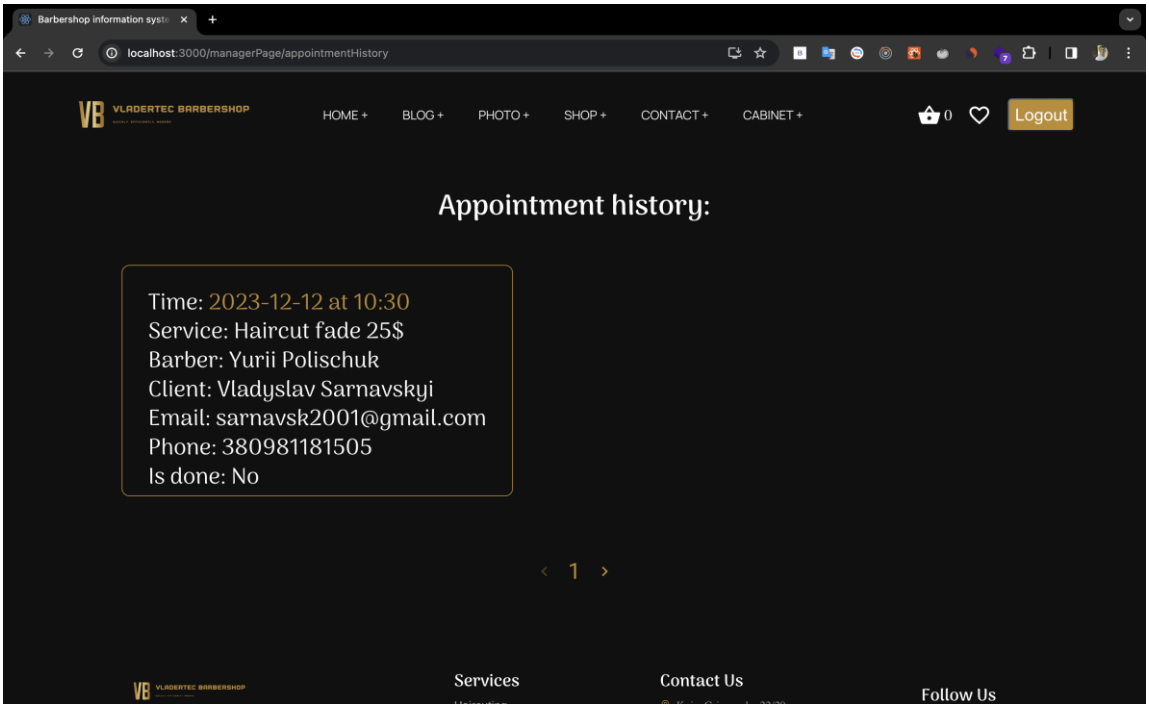

Рисунок 5.26 – Сторінка історії візитів

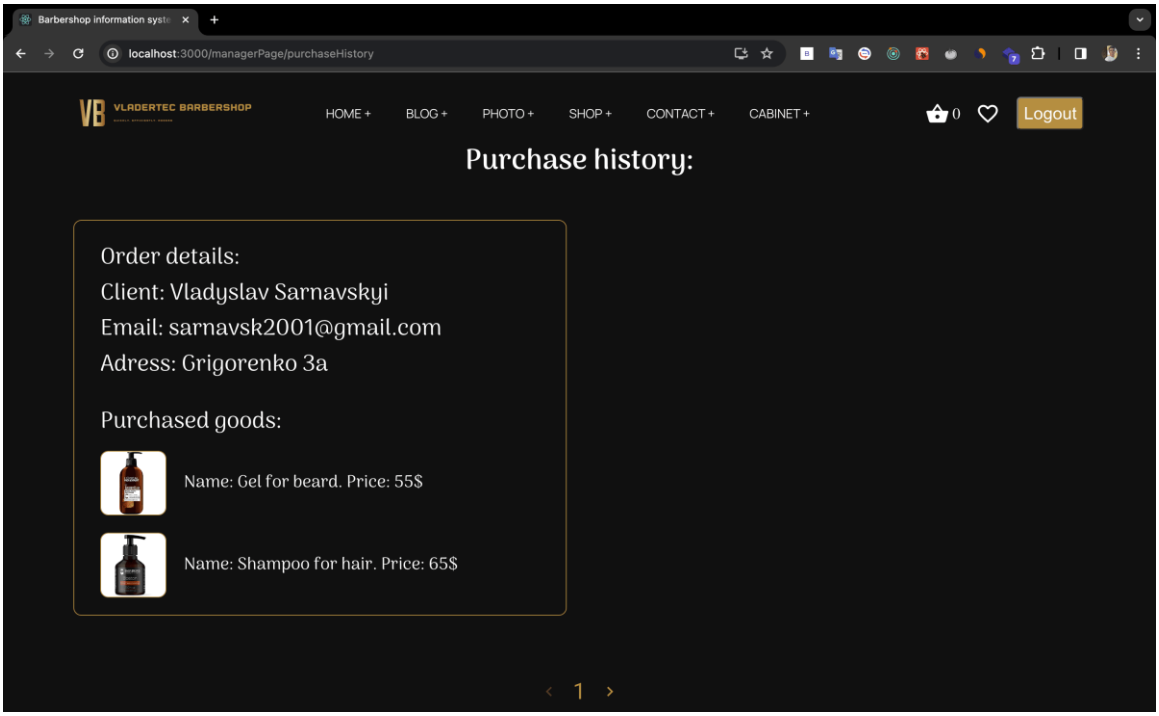

Рисунок 5.27 – Сторінка історії замовлень

Також менеджер може передивлятись всі пропозиції чи повідомлення для зворотного зв’язку з керівництвом від клієнтів. При натисканні кнопки «See feedbacks» в кабінеті менеджера, він переходить на сторінку історії фідбеків, де може їх переглядати. В карточці фідбеку присутня персональна інформація клієнта та його повідомлення для керівництва барбершопу. Дана сторінка продемонстрована нижче на рис. 5.28.

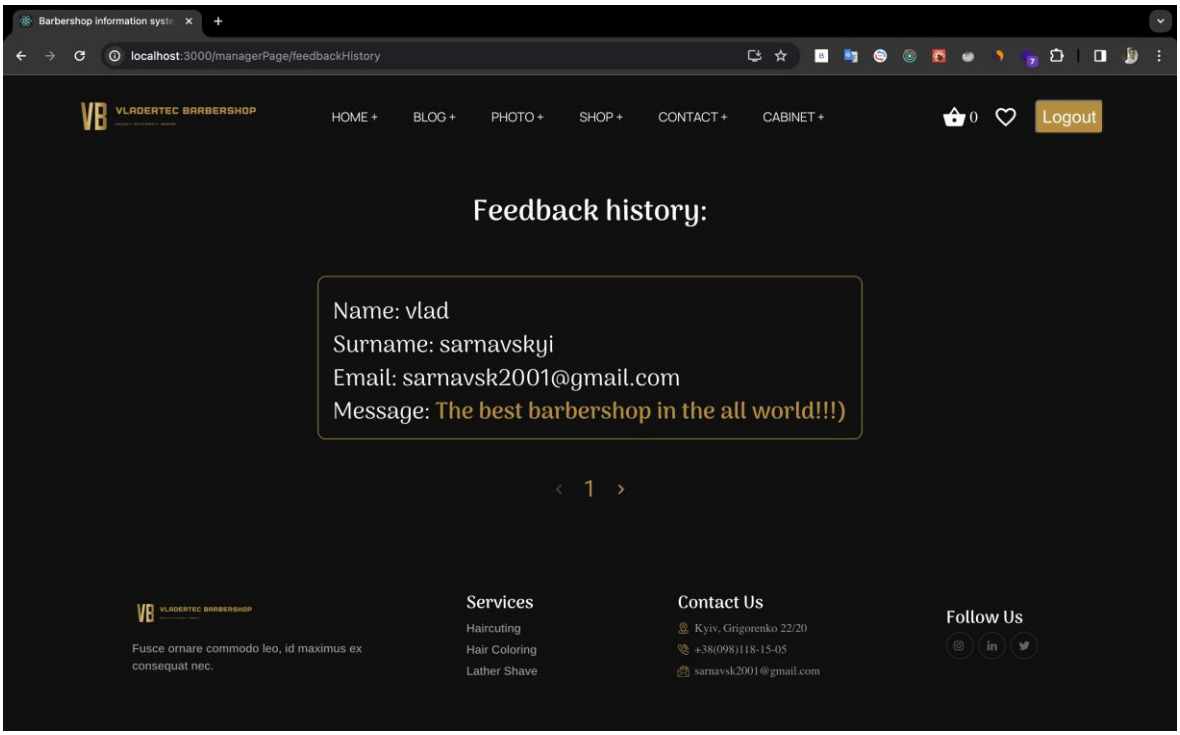

Рисунок 5.28 – Сторінка історії фідбеків

5.4 Методика роботи користувача з роллю барбера

Барбер – це не лише звичайна фігура у догляді за волоссям, це також ключовий гравець в бізнесі. Барбер також повинен володіти навичками для взаємодії з інформаційною системою, щоб планувати свій робочий час, організовувати прийом клієнтів та вести облік послуг.

Зареєструвати барбера в системі повинен менеджер. Після успішної авторизації, відобразиться кабінет для менеджера (рис. 5.29), де йому потрібно додати інформацію про себе, а саме: ім’я та прізвище, яке буде відображатись у

клієнтів на формі при записі на послугу. Також барбер через кабінет, натиснувши на кнопку, може продивитись історію своїх візитів (рис. 5.30) та після закінченого візиту натиснути кнопку «Is done», щоб завершити його та прибрати з бази.

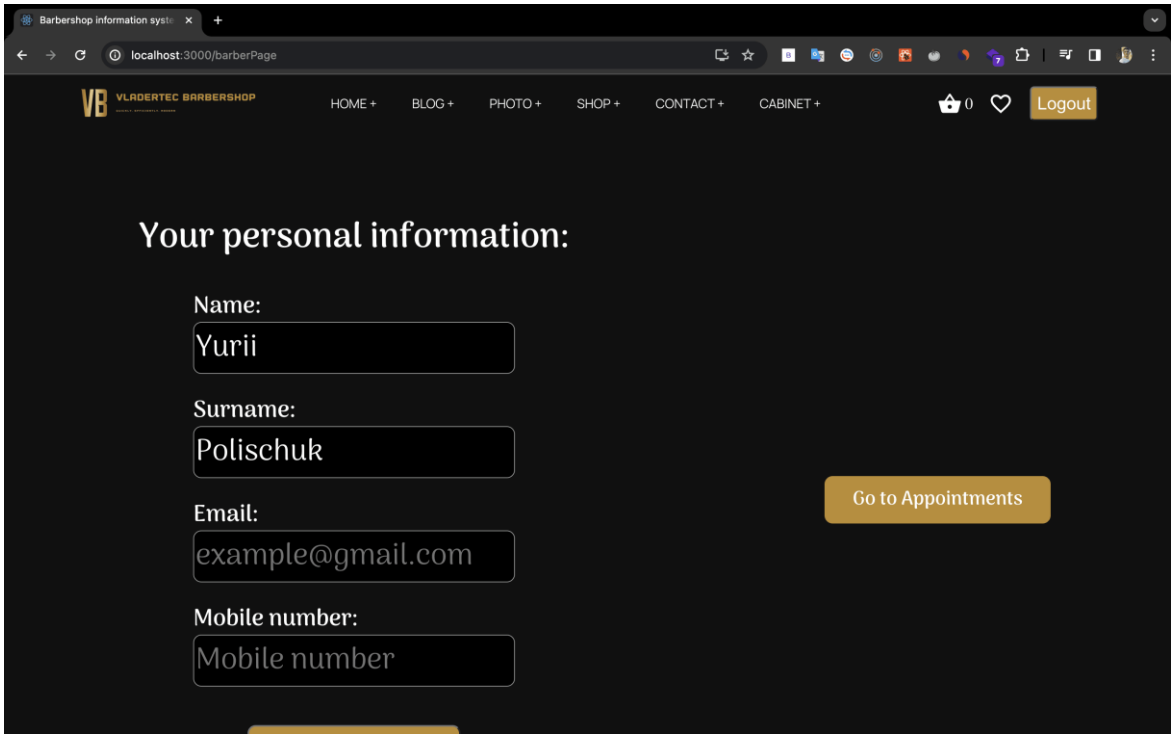

Рисунок 5.29 – Сторінка кабінету барберу

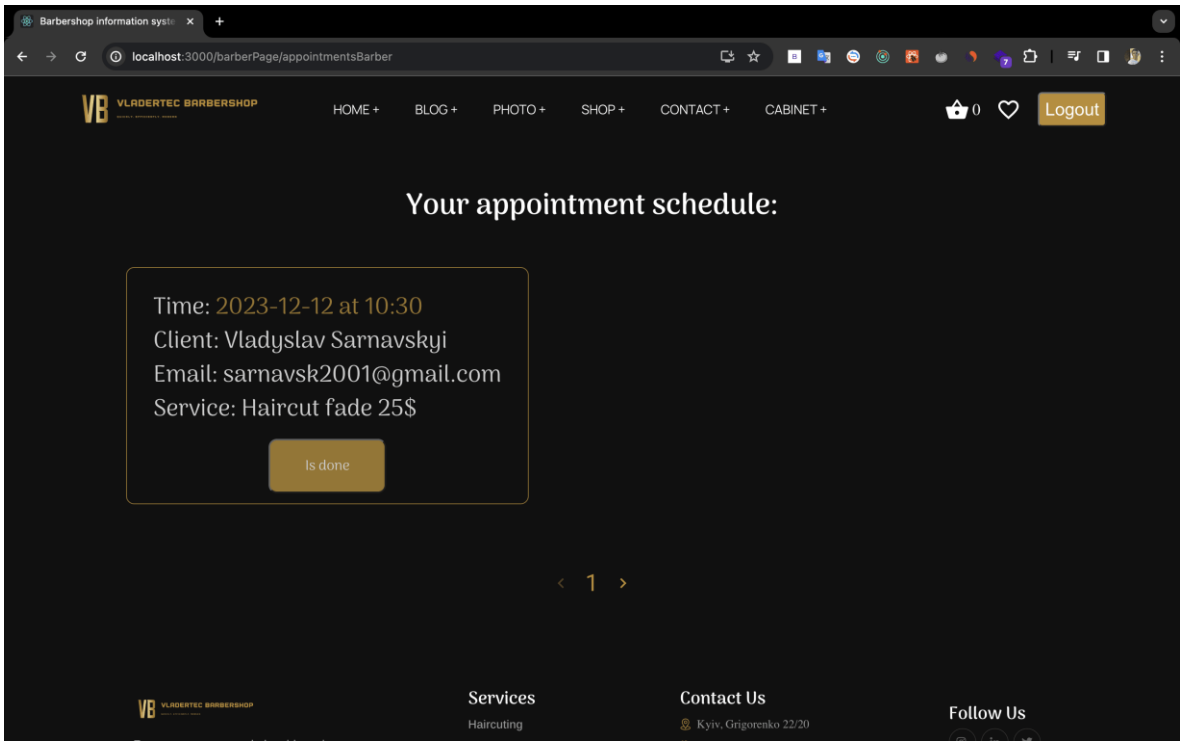

Рисунок 5.30 – Сторінка графіку візитів

5.5 Мобільне відображення інформаційної системи

Також, завдяки медіазапитам для кожного розміру пристрою, на стороні клієнтської частини, інформаційна система побудована по «респонсивним» принципап побудови.

«Респонсивна» інформаційна система – це система яка має здатність реагувати на різни типи пристроїв та їх екранів (роздільну здатність екранів), за допомогою медіазапитів в SCSS. Вони вказують різні стилі та макети в залежності від типу пристрою. Також забезпечують зручне та оптимальне використання системи, ефективну взаємодію з будь-яких пристроїв, та відображення інтуїтивно-зрозумілого інтерфейсу.

Тому весь функціонал і візуал системи підлаштований під мобільну роздільну здатність екрану всіх типів мобільних пристроїв (крім тих, у яких ширина екрану менше 280 пікселів, адже воно фізично не зможе коректно відображатись на девайсі), а продемонстровано на прикладі популярного на сьогодні моделі iPhone 6,7,8 Plus. Всі сторінки інформаційної системи в мобільному відображенні зображені на рис. 5.31-5.33.

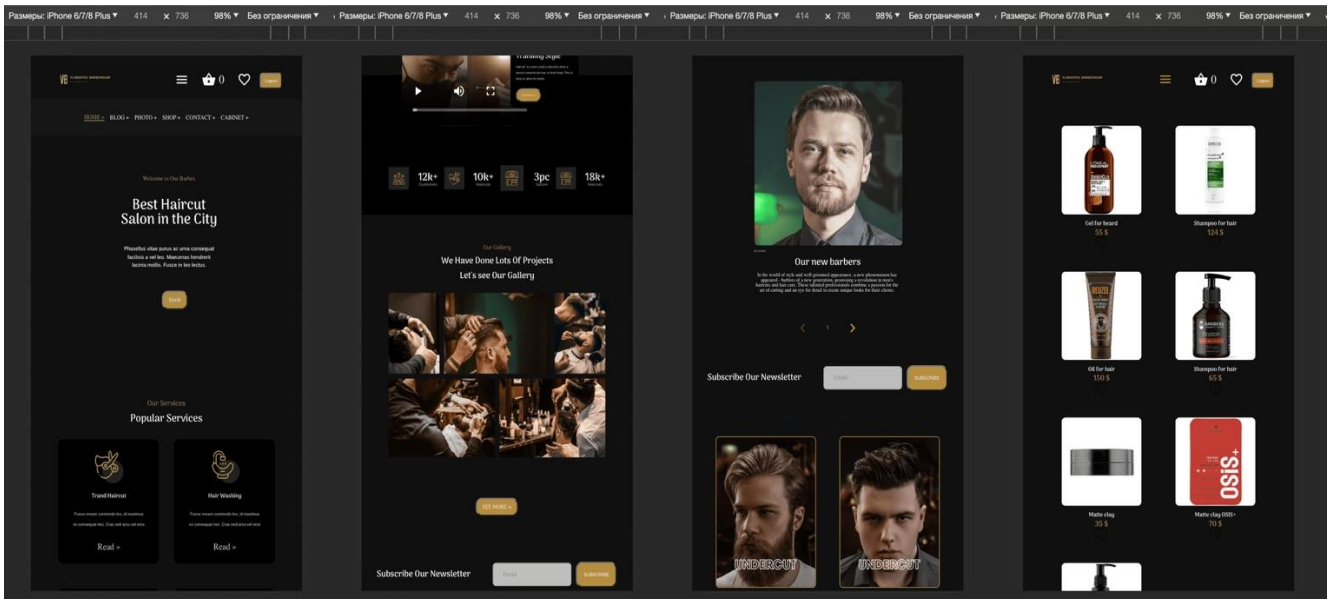

Рисунок 5.31 – Сторінки системи під роздільну здатність екрану iPhone 6,7,8 Plus

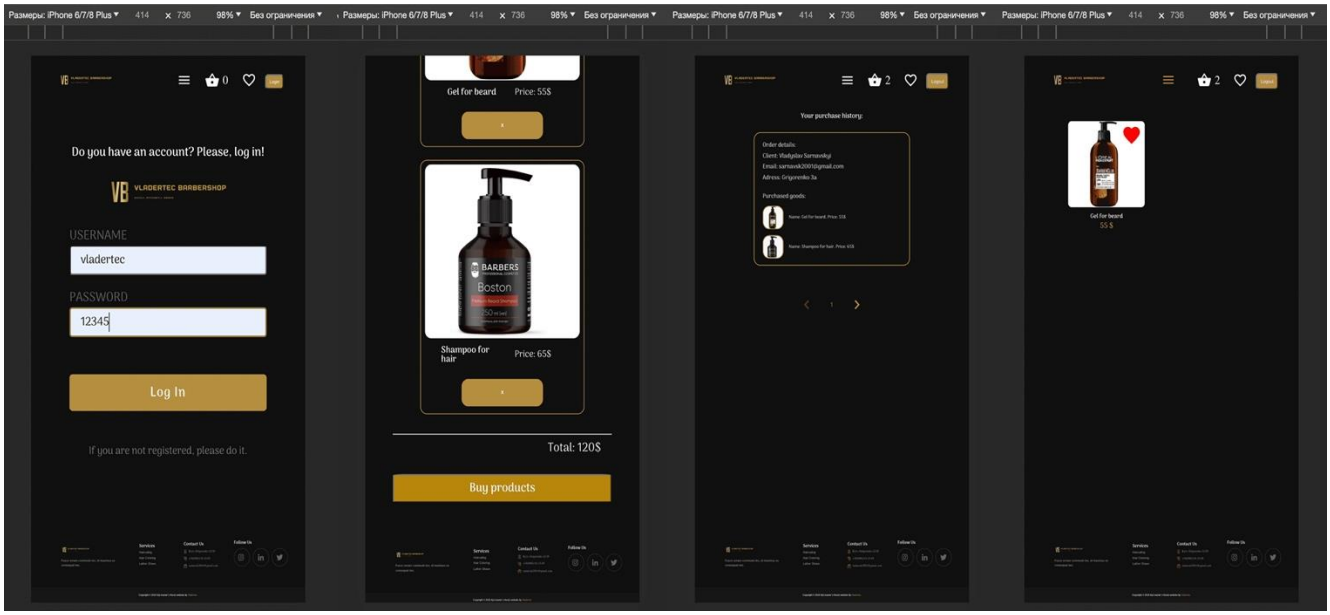

Рисунок 5.32 – Сторінки системи під роздільну здатність екрану iPhone 6,7,8 Plus

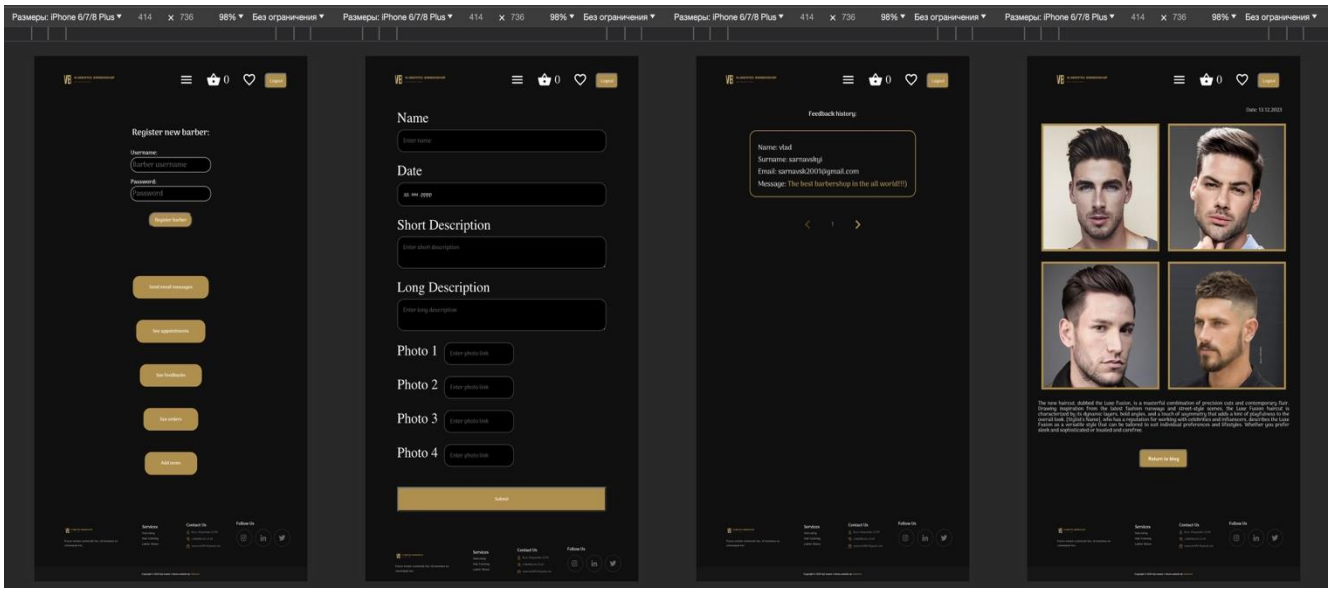

Рисунок 5.33 – Сторінки системи під роздільну здатність екрану iPhone 6,7,8 Plus

5.6 Висновки до розділу 5

Огляд інформаційної системи для мережі бербершопів на технологій стека «MERN» демонструє важливий крок у автоматизації більшості бізнес-процесів та підходу управління в цілому.

Інформаційна система пропонує не лише свої послуги, а й асортимент своїх продуктів для спеціалізованого доглядом за волоссям чи лицем. Це створює додатковий дохід для компанії.

Завдяки функціям перегляду історії всіх послуг, замовлень та пропозицій, менеджер може оптимізувати графік персоналу, покращити процес обслуговування клієнтів, і найголовніше – підвищити конкурентоспроможність мережі барбершопів.

6 ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

6.1 Тестування в інформаційних системи, їх види

Надійність та правильне виконання програми є важливими характеристиками для будь-якої інформаційної системи. Тому, тестування є невід'ємним етапом створення програмного забезпечення. Завдяки тестам, розробники та тестувальники не лише забезпечують правильне виконання програми, а й гарантують стабільність в її роботі з користувачами.

Тестування – це процес ідентифікації та перевірки значень системи або частини системи, щоб визначити, чи ці значення відповідають вимогам.

Тестування включає в себе різні методики та види, такі як: функціональне тестування, інтеграційне тестування, валідація значень тощо [22]. Детальніше, опис кожної з видів продемонстрований у нижче наведеній табл. 6.1.

Таблиця 6.1 – Опис видів тестування

№	Вид тестування	Опис видів тестування
1.	Функціональне тестування	Дозволяє визначити саме функціональні характеристики програми (чи відповідають вони вимогам)
2.	Інтеграційне тестування	Визначається та перевіряється взаємодія компонетів одним з одним (можуть бути і модулі програми) чи корректно вони разом відпрацьовують.
3.	Одиничне тестування (Unit-тестування)	В даному випадку перевіряється тільки один компонент або модуль, а саме його: функції, методи тощо.
4.	Валідація на вимоги	Інформаційну систему в цілому валідують на визначених вимогах чи функціях.

№	Вид тестування	Опис видів тестування
5.	Функціональне тестування безпеки	Щоб забезпечити безпеку даних проводяться тести на вразливість програми (перевірку на несанкціонований доступ, перехоплення даних)
6.	Тестування продуктивності	Систему навантажують та визначають оцінку цієї роботи, та скільки було витрачено ресурсів для цієї імітації навантаження.
7.	Мобільне тестування	Функції інформаційної системи тестують на різних ОС та девайсах, задля корректного відображення функціоналу та правильної її працездатності.
8.	Автоматизоване тестування	Залучаються спеціальні інструменти та скрипти, які виконують тестові сценарії щодо вимог. Це робить процес тестування простішим та ефективнішим.

6.2 Огляд технологій тестування інформаційної системи, їх особливості

В наш час сфера тестування швидко та ефективно розвивається, додаються нові функції, зміни, які допомагають та полегшують роботу розробникам або тестувальникам. Тому, потрібен огляд та аналіз популярних технологій, щоб виявити їх переваги та недоліки, обравши найкращу та оптимальну бібліотеку або фреймворк для написання тестів.

Jest – це сучасна та потужна бібліотека для тестування мови програмування JavaScript в якій наявні ефективні інструменти для аналізу та перевірки функціональності програми [23].

Jest володіє інструментами та функціями для інтеграційних тестів, одиничного тестування (unit-тестування), функціональних тестів, тестування API. Також у випадку розробленої мною інформаційної системи на стеці «MERN»

наявна підтримка тестування Redux (стейт-менеджеру) та React компонентів (підтримка snapshot-тестування).

Переваги бібліотеки тестування Jest:

– React – головною перевагою є вбудована широка підтримка тестування та аналізу React-компонентів (snapshot-тестування, що перевірку тестування змін при рендеру) та бібліотеки React в загальному;

– мокування (mocking) – засоби можуть легко імітувати функції та об’єкти для їх тестування на справність та коректність роботи, що робить процес тестування зручним та продуктивним;

– продуктивність – має високу оптимізацію та продуктивність під час виконання тестування;

– інструменти – володіє багатьма сучасними інструментами та інтуїтивно зрозумілим інтерфейсом для створення ефективних та продуктивних тестів;

– автоматичне виконання тестів – бібліотека швидко визначає та запускає тести (є паралельне запускання тестів, що пришвидшує їх виконання та робить результат тестування ефективним);

– assertions – підтримка вбудованих тверджень для оптимізації створення та збору тестових сценаріїв;

– популярність – наявність широкої спільноти розробників, що впливає на якість документації, обговорень в групах, широкого використання;

– комплексне тестування – підтримка великого спектру різних видів тестів для обширного тестування;

– інтеграція – легко інтегрується з іншими сучасними та популярними інструментами (Babel, Webpack), що зручно при створенні великого проекту.

Недоліки бібліотеки тестування Jest:

– ресурсомісткість – при паралельному тестуванні може використовуватись великий об’єм пам’яті;

– React Native – хоч і Jest має тісну інтеграцію і вбудовані інструменти для взаємодії з React, але для React Native потрібні додаткові конфігурації та налаштування;

– оновлення – на деяких версіях можуть бути проблеми зі стабільністю й так само під час оновлення версії.

Mocha – це продуктивний та ефективний фреймворк для тестування мови програмування JavaScript на правильність виконання роботи та задач програми. Має простоту у виконанні, розширюваність, швидкість та популярність серед розробників [24].

У порівнянні з минулою технологією тестування Jest, головна відмінність в тому, що Mocha не має вбудованого функціоналу для мокування (це засоби, які можуть легко імітувати функції та об’єкти для їх тестування на справність та коректність роботи).

Переваги фреймворку тестування Mocha:

– розширюваність – фреймворк може легко розширюватись через власні плагіни або ж ефективно інтегруватись із сторонніми;

– асинхронність – підтримка асинхронності, що дозволяє писати асинхронні тести;

– популярність – наявність широкої спільноти розробників, що впливає на якість документації, обговорень в групах, широкого використання;

– продуктивність – має високу оптимізацію та продуктивність під час виконання тестування;

– паралельність виконання – є підтримка паралельного виконання тестів, що пришвидшує їх розробку та робить результат тестування ефективним;

– гнучкість – є різні стилі тестів, що дозволяє писати гнучкі тести для перевірки системи (Mocha підтримує Behavior-Driven Development (BDD) – реалізація тестів під очікувані вимоги системи та Test-Driven Development (TDD) – спочатку реалізація тестів для нового функціоналу, а потім власне написання коду під цей тест).

Недоліки фреймворку тестування Mocha:

– веб-розробка – для веб-розробки потрібні додаткові бібліотеки, адже немає вбудованої підтримки тесті для веб-застосунків.

– автоматичне виконання – фреймворк не має вбудованих інструментів для автоматичного визначення та запуску тестів;

– мокування – немає засобів, які можуть легко імітувати функції та об’єкти для їх тестування на справність та коректність роботи;

– швидкість – у деяких випадках швидкість фреймворку може поступатись швидкості іншим інструментам тестування (особливо коли є велика кількість тестів);

Jasmine – це потужний та ефективний фреймворк для тестування мови програмування JavaScript на правильність виконання роботи та задач програми. Відзначається своєю ефективністю та простістю, що дозволяє розробникам та тестувальникам створювати продуктивні тест й перевіряти коректність виконання програми до заданих вимог.

Також, головною особливістю є розвинутий синтаксис під Behavior-Driven Development (BDD) – реалізація тестів під очікувані вимоги системи для забезпечення якості коду.

Переваги фреймворку тестування Jasmine:

– BDD (Behavior-Driven Development) синтаксис – є головною перевагою, адже весь інструментарій спеціалізований під даний стиль тестів, роблячи тести простими у створенні та легкими у розумінні;

– мокування (mocking) – засоби можуть легко імітувати функції та об’єкти для їх тестування на справність та коректність роботи, що робить процес тестування зручним та продуктивним;

– assertions – підтримка вбудованих тверджень для оптимізації створення та збору тестових сценаріїв;

– популярність – наявність широкої спільноти розробників, що впливає на якість документації, обговорень в групах, широкого використання;

– асинхронність – підтримка асинхронності, що дозволяє писати асинхронні тести;

– продуктивність – має високу оптимізацію та продуктивність під час виконання тестування;

– незалежність тестів – дозволяє реалізовувати повністю незалежні тести, що забезпечує надійність та ізоляцію тестів.

Недоліки фреймворку тестування Jasmine:

– неінтегрованість – головним недоліком є погана інтегрованість з популярними бібліотеками та фреймворками для створення інтерфейсу користувача;

– автоматичне виконання – фреймворк не має вбудованих інструментів для автоматичного визначення та запуску тестів;

– швидкість – у деяких випадках швидкість фреймворку може поступатись швидкості іншим інструментам тестування (особливо при великої кількості тестів);

– автоматичне виконання – фреймворк не має вбудованих інструментів для автоматичного визначення та запуску тестів;

– вузьконаправленість стилю написання – так як використовується Behavior-Driven Development (BDD) синтаксис, він є специфічним, тому не всі розробники та тестувальники його використовують його, що може бути зайвою складністю в написанні тестів.

6.3 Реалізація тестування в інформаційній системі

Після огляду та аналізу популярних технологій тестування для виявлення їх переваг та недоліків, була обрана найкраща та оптимальна бібліотеку написання тестів для веб-застосунку – Jest.

Redux є важливою частиною системи, адже зберігає в собі стан системи в одному централізованому об’єкті під назвою store. Інформація в цих станах використовується на всіх рівнях вкладеності незалежних компонентів, що забезпечує зручний, швидкий та ефективний доступ до інформації для її відображення у системі.

Для роботи з бібліотекою, спочатку потрібно її встановити через пакет менеджерів NPM, командою: npm install --save-dev jest. Щоб почати тестування, потрібно створити файл для цього, але в назві має міститись слово test (наприклад:

MainPage.test.jsx чи reducer.test.js). Після цього можна приступати до роботи створення тестів.

Завдяки Jest було зроблено написання тестів для кожного підоб'єкту в store та автоматизовано їх. Це одиничне тестування (Unit-тестування), де перевіряється тільки один компонент або модуль, а саме його: функції, методи тощо. На рис. 6.1 зображено Unit-тестування Redux-Store.

Рисунок 6.1 – Unit-тестування Redux-Store

Jest також має вбудовані інструменти, широку підтримку тестування та аналізу React-компонентів. Основним інструментом для взаємодії з компонентами стала бібліотека @testing-library/react [25], Jest – автоматизував процес тестування.

Тому, один з прикладів написаних мною – це тестування правильності рендеренгу та взаємодії з компонентом (з компонентом модального вікна, яке повинно відображатись при додаванні товару в кошик). Я використав функціонал Jest й створив моки та бібліотеку @testing-library/react (задля отримання доступу до DOM-елементів або ж елементів React). Приклад компонентного тестування продемонстрований на рис.6.2.

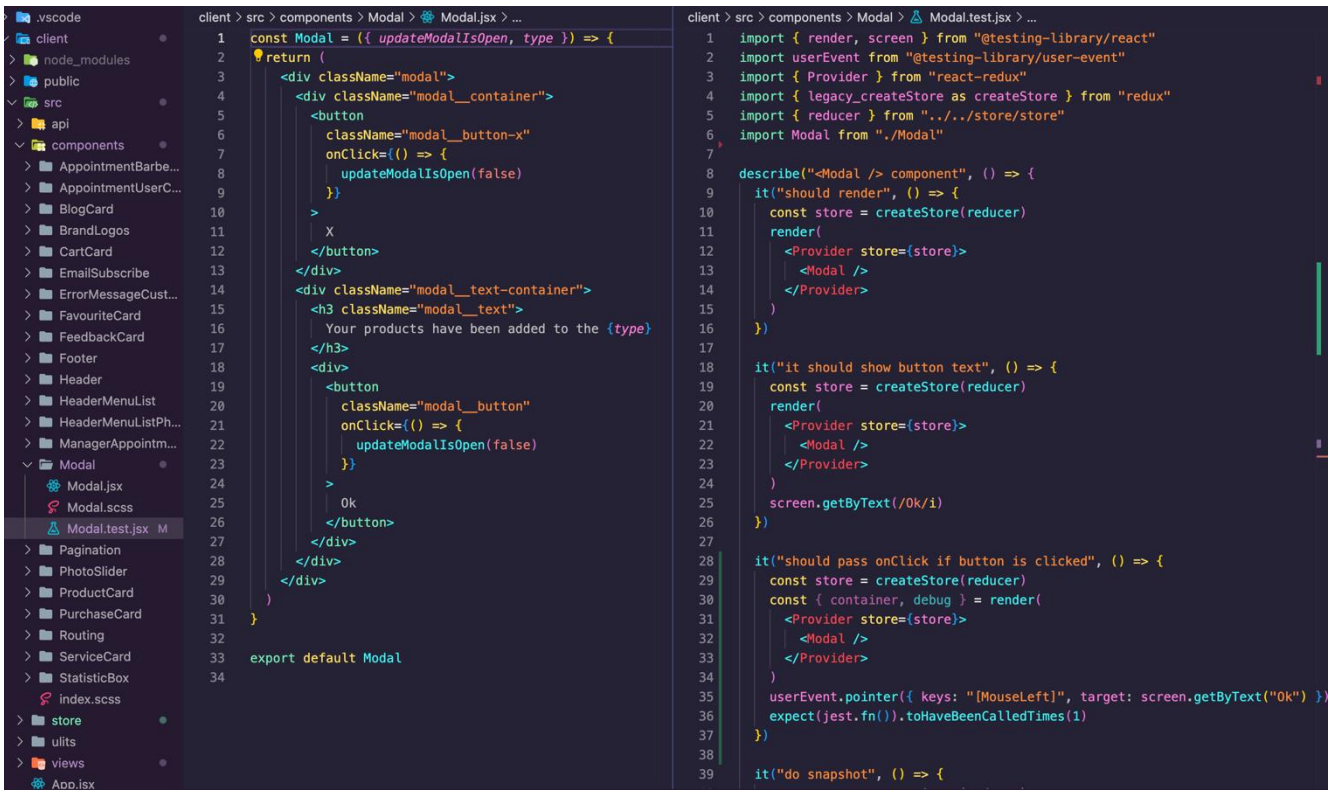

Рисунок 6.2 – Компонентне тестування

Щоб перевірити роботу та результат тестування, в командному рядку (CLI) – потрібно ввести команду: npm test. Після чого можемо побачити результат тестування, який продемонстровано на рис. 6.3.

Рисунок 6.3 – Результати тестування бібліотеки Jest

6.4 Висновки до розділу 6

Вибір даної бібліотеки демонструє орієнтацію на сучасні практики для тестування інформаційних систем. Як було зазначено у розділі 6.3, Jest характеризується своєю простотою, швидкістю, продуктивністю, сучасністю та тісною інтеграцією з бібліотекою React, що і стало головним рішенням, щодо обрання даної технології для тестування системи.

Результатом використання стало створення ефективних та продуктивних тестів, які відпрацювання згідно заданим вимогам та зробили систему надійною та стабільною. А автоматизація даних тестів полегшила продуктивність створення системи.

7 СТАРТАП-ПРОЕКТ

7.1 Опис ідеї проекту

При виконанні даної частини стартап-проекту був огляд та аналіз ідеї, результатом якого є наведені таблиці:

– опису суті ідеї та її вміст;

– напрямки використання ідеї (табл. 7.1);

– головні переваги для користувача;

– відмінності рішень, які вже існують.

Передбачаються такі етапи для аналізу техніко-економічних переваг ідеї порівняно з пропозиціями конкурентів:

– визначено техніко-економічні характеристики ідеї;

– визначено конкурентів (існуючі аналоги на ринку) та збір техніко-економічних показників;

– порівняння гірших, нейтральних та кращих показників із проектом конкурента (табл. 7.2).

Таблиця 7.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигода для користувачів
Розроблена інформаційна система може використовуватись у барбершопах для автоматизації процесів (онлайн-запис чи онлайн-замовлення	Оптимальне управління графіками та записами	Зручність та швидкість обслуговування.
	Повідомлення та нагадування	Уникання забуття за візитів чи замовлень, отримання своєчасної інформації (знижки, новини тощо)
	Збереження історії клієнтів (історія записів,	Зручний доступ до історії, що дає

Зміст ідеї	Напрямки застосування	Вигода для користувачів
товарів) та підвищення ефективності управління.	візитів, кошика чи улюблених товарів)	можливість відстежувати результати.
	Безпека та конфіденційність	Відчуття довіри та безпеки.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W слабка сторона	N нейтральна сторона	S сильна сторона
	Мій проект (інформаційна система)	GC Barbershop (інформаційна система)			
Простота використання	Інтерфейс системи є інтуїтивно-простим та зрозумілим.	Інтерфейс сис-теми є інтуїтивно-простим та зрозумілим.		+	
Наповненість інформацією	Є багато другорядних сторінок з детальною та обширною інформацією.	Обширна інформація наявна тільки на головній сторінці.			+
Мобільне відображення	Весь функціонал і візуал системи підлаштований під мобільну роздільну здатність екрану всіх типів	Весь функціонал і візуал системи підлаштований під мобільну роздільну здатність екрану всіх типів		+	

Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W слабка сторона	N нейтральна сторона	S сильна сторона
	Мій проект (інформаційна система)	GC Barbershop (інформаційна система)			
	мобільних пристроїв.	мобільних пристроїв.			
Дизайн	Дизайн є мінімалістичним	Дизайн є мінімалістичним		+	
Онлайн-запис	Онлайн запис проводиться через вбудований внутрішній функціонал системи.	Онлайн запис проводиться через сторонню технологічну базу запису-онлайн «Alte-gio».			+
Онлайн-замовлення	Наявний функціонал для замовлення товарів	Наявний функціонал для замовлення товарів		+	
Швидкодія системи	Загрузка сторінок вібувається швидко, адже використовується React.	Загрузка сторінок вібувається швидко (особливо головна сторінка через завантаження відео)			+

7.2 Технологічний аудит ідеї проекту

В даній частині стартапу виконаний технологічний аудит, щоб дізнатись ефективність реалізації інформаційної системи.

Таблиця 7.3 – Технологічна здійсненність проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Розроблена інформаційна система може використовуватись у барбершопах для автоматизації процесів (онлайн-запис чи онлайн-замовлення товарів) та підвищення ефективності управління.	Мова JS	Наявна	Доступна
	База даних MongoDB	Наявна	Доступна
	Середовище виконання Node.js	Наявна	Доступна
	Бібліотека React	Наявна	Доступна
	Фреймворк Express	Наявна	Доступна
	Фреймворк Mongoose	Наявна	Доступна
	Бібліотека Redux	Наявна	Доступна
	Бібліотека React Router	Наявна	Доступна
	Бібліотека Jest	Наявна	Доступна

Технологічними технологіями були обрані: мова програмування JS, база даних MongoDB, середовище виконання Node.js, фреймворки Express та Mongoose, бібліотеки React, Redux, React Router та Jest. За аналізом табл. 7.3 була визначена можливість технічної реалізації проекту (інформаційної системи для барбешопів).

7.3 Аналіз ринкових можливостей запуску стартап-проекту

В даній частині стартап-проекту визначаються потенційні ринкові можливості, які потрібні для успішного впровадження проекту на ринку та ринкових загроз, які заважатимуть даному релізу стартап-проекту. Завдяки цьому аналізу, можна спланувати шляхи розвитку проекту. Спочатку був проведений в табл. 7.4 аналіз попиту.

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проекту

Показники стану ринку (найменування)	Характеристика
Кількість головних гравців, од	10
Загальний обсяг продаж, грн/ум.од	35000
Динаміка ринку (якісна оцінка)	Зростає
Наявність обмежень для входу (вказати характер обмежень)	-
Специфічні вимоги до стандартизації та сертифікації	-
Середня норма рентабельності в галузі (або по ринку), %	20%

За результатами табл. 7.4, ринок є привабливим для входження. Також визначені (табл. 7.5) потенційні групи клієнтів, їх вимоги до товару.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Автоматизація процесів роботи барбешопу, розрекламування та підвищення якості управління.	Барбершопи, салони краси та компанії, які надають послуги краси.	Молодь: швидкість системи, мобільне відображення. Старші групи: використовують дзвінки.	Зручність, ефективність, швидкість, надійність.

Також, проведено аналіз ринкового середовища, а саме – фактори загроз та можливостей при впровадженні проекту. Зображено на табл. 7.6, 7.7.

Таблиця 7.6 – Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Технічні збої	Програмні збої	Резервне копіювання даних, регулярне огляд програми.
Конкуренція	Вихід на ринок більш ефективної інформаційної системи та яка має кращий функціонал та характеристики.	Оптимізація, модернізація,

Таблиця 7.7 – Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Поява нових технологій	Нові технології та інструменти на ринку, які реалізують ефективну та сучасну інформаційну систему	Засвоєння нових технологій та оптимізація системи ними
Конкуренція	Відсутність проектів-аналогів на ринку	Розрекламування

Надалі був проведений аналіз пропозиції: у табл. 7.8 наведені загальні риси конкуренції.

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємництва
Тип конкуренції – монополістична	Багато виробників, які продають схожі товари, моють невелику частку ринку та самі впливають на ціну свого товару.	Підтримка актуальності, новизни та якості товару (інформаційної системи)
Рівнень конкурентної боротьби – міжнародний	Компанії-конкуренти, які можуть бути з інших країн	Розробки універсальної інформаційної системи, яка легко може адаптуватись до інших галузей.
Галузева ознака – внутрішньогалузева	Спеціалізується на розробці інформаційних рішень для конкретних бізнесів – барбершопів, салонів краси та спа	Постійне оновлення і модифікація продукту (інформаційної системи)
Конкуренція за видами товарів – товарно-видова	Конкуренція між якістю, швидкістю, видами, ефективністю інформаційних систем.	Потрібна реалізація системи без недоліків, які присутні у продуктах конкурентів.
За характером конкурентних переваг – нецінова	Вдосконалення технологій, щоб зробити низьку собівартість продукту.	Інновації, функціональність, модульність, зручність продукту.
За інтенсивністю – не марочна	Наявність бренду, який має незначну роль у використанні інформаційної сиситеми	Рекламні кампанії

У табл. 7.9 був проведений аналіз конкуренції у галузі (за моделлю М. Портера).

Таблиця 7.9 – Аналіз конкуренції за М. Портера

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Перелік прямих конкурентів	Бар'єри входження у ринок	Фактори сили постачальників	Фактори сили споживачів	Фактори загроз продуктів-замінників
	GC барбешоп	Вже існуючі рішення	-	Якість продукту, надійність, зручність	Авторитет, функціональність
Висновки	Є багато конкурентів,	Потрібен аналіз інформаційної системи конкурентів. Вихід на ринок – 5 місяців.	-	Завдяки клієнтам, товари повинні відповідати таким вимогам як: надійність, швидкість, ефективність тощо	

На основі аналізу конкуренції, проведеного в табл. 7.9, обґрунтовується перелік факторів конкурентоспроможності у табл. 7.10.

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування
Функціональність та модульність	Дозволяє ефективно управляти своїм бізнесом та керувати клієнтськими записами.
Актуальність	Впровадження нових технологій, робить систему більш ефективною та швидкою для клієнта.
Надійність	Безпечне зберігання всієї інформації клієнта
Простий інтерфейс	Робить користування системою простою та зручною
Швидкодія	Система швидше реагує на дії користувача

Із зазначиними факторами конкурентоспроможності (табл. 7.10), також був проведений аналіз сильних та слабких сторін проекту (табл. 7.11).

Таблиця 7.11 – Аналіз сильних та слабких сторін проекту

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
		-3	-2	-1	0	1	2	3
Функціональність та модульність	18						+	
Актуальність	19					+		
Надійність	15			+				
Простий інтерфейс	18				+			
Швидкодія	19						+	

Кінцевим етапом ринкового аналізу можливостей є складання SWOT-аналізу (табл. 7.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Таблиця 7.12 – SWOT-аналіз

Сильні сторони	Слабкі сторони
Швидкість	Авторитет
Ефективність	
Функціональність	
Можливості	Загрози
Масштабування	Конкуренція
Покращення функціоналу	

Завдяки SWOT-аналізу створено альтернативи ринкової поведінки для впровадження на ринок та час їх ринкової реалізації з баченням конкурентних проектів. Визначені альтернативи були проаналізовані у табл. 7.13.

Таблиця 7.13 – Альтернатива ринкового впровадження проекту

Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Запуск маркетингової кампанії	80%	2 місяці
Розвиток та вдоскона-лення користовацького інтерфейсу системи	65%	4 місяці
Створення програми лояльності для клієнтів	75%	3 місяці

Тому, завдяки цьому порівняння було обрано першу альтернативу.

7.4 Аналіз ринкової стратегії проекту

Розробка стратегії ринку передбачає в проведеня вибору цільових груп потенційних споживачів (табл 7.14) та їх аналіз.

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Компанії, які пов'язані з наданням послуг краси	Високий	Високий	Сильна	Середня складність
Приватні підприємства, діяльність яких пов'язана з веб-застосунками	Високий	Високий	Сильна	Складно

Для взаємодії в даних сегментах ринку розроблена основна стратегія конкурентної поведінки у табл 7.15

Таблиця 7.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Запуск маркетинг-гової кампанії	Визначити потреби груп потенційних споживачів, та зробити рекламу під групи	Ціна, якість, функціональність, дизайн інформаційної системи	Стратегія диференціації

Далі було обрано стратегію конкурентної поведінки, яку продемонстровано у табл. 7.16.

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів чи існуючих?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Забирати існуючих	Так: якість, функціональність, структру інформаційної системи	Зайняття конкурентної ніші

Також, на підставі всіх даних досліджень була розроблена стратегія позиціонування, яка представлена в табл. 7.17.

Таблиця 7.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Швидкість, безпека, ефективність, зручність, гарний дизайн, наповненість інформацією	Стратегія диференціації	Позиція заснована на основі порівняння продуктів з іншими конкурентами	Швидкість Функціональність Зручність

Після завершення аналізу була визначена система стратегічних рішень ринкової поведінки компанії (напрямки діяльності проекту на ринку).

7.5 Розроблення маркетингової програми стартап проекту

Була сформована у табл. 7.18 маркетингова концепція товару, який отримає споживач.

Таблиця 7.18 – Визначення ключових перевагконцепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
Зручність	Сайт повинен бути інтуїтивно-простим для розуміння	Заощаджує час в провадженні та користуванні системою
Функціональність	Функціонал повинен бути обширним та надійним	Розширений функціонал дає змогу користувачу вбудовані функції для запису чи купівлі спеціалізованих товарів
Дизайн	Мінімалістичний дизайн	Дизайн повинен викликати відчуття довіри та враження
Швидкість	Швидка обробка запитів та реагування функціоналу системи	Використання оптимальних та швидких алгоритмів, бібліотек, інших інструментів.

Після маркетингової моделі, проект буде захищено від копіювання. Наступним кроком є розроблення трирівневої маркетингової моделі товару, яка описана в табл. 7.19.

Таблиця 7.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>Товар за задумом</td><td colspan="3">Зручність, функціональність та швидкість отримання практичного результату</td></tr>
<tr><td>Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td rowspan="4">Товар із підкріпленням</td><td>Система володіє обширним функціоналом, який допомагає автоматизувати процеси та оптимізувати роботу закладу, що підвищує конкурентоспроможність закладу</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: система має авторизацію від несанкційонованого доступу</td></tr>
<tr><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: Vladertec</td></tr>
<tr><td rowspan="2">Товар із підкріпленням</td><td colspan="3">До продажу: відсутнє</td></tr>
<tr><td colspan="3">Після продажу: підтримка інформаційної системи (можливість модифікації системи)</td></tr>
<tr><td colspan="4">Вихідний код є закритим</td></tr>
</table>

У табл. 7.20 продемонстровано визначення меж встановлення ціни.

Таблиця 7.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
25000-27000 грн	34000 грн	3 групи мають достатній рівень	Створення інформаційної системи – 35000 грн, підтримка 2500 грн в місяць

Наступний крок – це визначення оптимальної системи збуту, яке золбражено в табл. 7.21.

Таблиця 7.21 – Формування систем збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Цільовими клієнтами є компанії у сфері краси чи обслуговування, які хочуть автоматизувати та оптимізувати процес роботи	Встановлення контактів з замовниками (користувачами) та підтримки зворотного зв’язку з ними для вирішення проблем чи питань	Один (адже розробник передає одразу товар (інформаційну систему) замовнику/споживачу)	Наявний прямий канал збуту, що дозволить мінімізувати витрати

Останнім етапом є розроблення концепції маркетингових комунікацій у табл. 7.22.

Таблиця 7.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Цільовими клієнтами є компанії у сфері краси чи обслуговування, які хочуть автоматизувати та оптимізувати процес роботи	Телефонний зв’язок, зустрічі, онлайн-зустрічі, інтернет, література	Позиція порівняння фірми з товарами конкурентами	Створити репутацію як надійна та продуктивна фірма, збільшення замовлень, збільшення прибутку фірми.	Ваш бізнес – наша турбота, інновації – наше керівництво.

7.6 Висновок до розділу 7

В даному розділі роботи був проведений аналіз інформаційної системи для мережі барбершопів у формі стартап-проекту. Як результат, було визначено що на ринку є конкуренція зі схожими послугами, але їх рішення можуть мати відмінності, тому вихід на ринок повинен бути з виваженою стратегією.

Створення інформаційної системи для барбешопів є обґрунтованим рішенням, адже система зможе здобути свою цільову аудиторію, надаючи ефективні інструменти для управління барбешопом та автоматизацію всіх процесів роботи закладу.

ВИСНОВКИ

Розробка інформаційної системи для мережі барбершопів на стеці технологій «MERN» є обґрунтованим рішенням та вдалим вибором. Стек містить в собі популярні на сьогодні можливості та технології, які дозволяють швидко, оптимально та ефективно розробити інформаційну систему.

Спочатку був здійснений аналіз предметної області, де були описані мета та постановка задачі, проблематика області, поняття інформаційної системи в цілому та її види та вимоги.

Після цього, проведений огляд всіх сучасних технологій, середовищ виконання, баз даних, що вказало на їх переваги та недоліки. Як результат цих досліджень – було обрано саме стек технологій «MERN» для розробки інформаційної системи для мережі барбершопів.

Власне розробка системи визначається високою ефективністю, швидкодією та продуктивністю. У стеці «MERN»: MongoDB – відповідає за гнучкість та безпеку даних, Express – ефективно обмінюється даними та реалізує логіку серверної частини, React – відображує надійні, швидкі та прості у розумінні компоненти, а Node.js – з’єднує всі частини стека разом, реалізуючи оптимальну роботу системи.

Завдання стартап проекту полягало у проведенні маркетингову аналізу перспектив реалізацій інформаційної системи для мережі барбершопів. Були визначені: слабкі та сильні сторони ідей проекту, технологічний аудит ідеї створення цієї системи, аналіз ринкових можливостей, створення ринкової стратегії та маркетингової програми.

Тобто, дана інформаційна система не тільки автоматизує всі внутрішні процеси роботи бізнесу, а й завдяки інтуїтивно-зрозумілому інтерфейсу та мінімалістичному дизайну привабить нових потенційних клієнтів, що підвищить конкурентоспроможність бізнесу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Інформаційні системи. Інформаційні технології [Електронний ресурс] – Режим доступу до ресурсу: https://ivanytskyi.blogspot.com/p/httpsdocs.html.

2. Функціональні та нефункціональні вимоги [Електронний ресурс] – Режим доступу до ресурсу: https://visuresolutions.com/uk/requirements-management-traceability-guide/functional-vs-non-functional-requirements/.

3. Інтегроване середовище розробки [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/hyperskill/ide.html#gsc.tab=0.

4. Git та Github [Електронний ресурс] – Режим доступу до ресурсу: https://sebweo.com/scho-take-git-ta-github-kerivnitstvo-dlya-pochatkivtsiv/.

5. React Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn.

6. Angular Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs.

7. Vue.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/guide/introduction.html.

8. Node.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/docs/latest/api/.

9. Deno Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.deno.com/runtime/manual/.

10. GraalVM Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.graalvm.org/latest/docs/.

11. Система управління базами даних [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Система_управління_базами_даних.

12. Типи баз даних: особливості, відмінності та приклади [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/types-of-databases/.

13. MySQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/.

14. Oracle Database Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/database/.

15. Neo4j Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://neo4j.com/docs/.

16. MongoDB Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/.

17. Що таке npm і навіщо він потрібен [Електронний ресурс] – Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/271-chto-takoe-npm-i-zachem-on-nuzhen.

18. Express.js – фреймворк для веб-застосунків [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/uk/.

19. Вступ до Mongoose для MongoDB та Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://code.tutsplus.com/uk/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527a.

20. Redux Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org/tutorials/fundamentals/part-1-overview.

21. Material UI – Overview [Електронний ресурс] – Режим доступу до ресурсу: https://mui.com/material-ui/getting-started/.

22. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/.

23. Jest Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://jestjs.io/docs/getting-started.

24. Mocha Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://mochajs.org/api/mocha.js.html.

25. React Testing Overview [Електронний ресурс] – Режим доступу до ресурсу: https://uk.legacy.reactjs.org/docs/testing.html.