МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Програмне забезпечення інтернет-магазину товарів для косметологів. Клієнтська частина

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.21810802

Студент _____ В.О. Антонов

підпис

«__» _____ 2022 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2022 р.

Керівник канд. тех. наук, доцент _____ Г. В. Горбань

підпис

«__» _____ 2022 р.

Миколаїв 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії програмного
забезпечення, канд.техн.наук, доцент,
_____Є.О. Давиденко
«____»_____2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук

Антонов В'ячеслав Олександрович
.
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
«Програмне забезпечення інтернет-магазину товарів для косметологів. Клієнтська частина»
.
Затверджена наказом по ЧНУ від «01» грудня 2021 р. № _____

2. Строк представлення кваліфікаційної роботи «____» ___ 2022 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Вхідні дані до роботи − функціональні та нефункціональні вимоги до програмного забезпечення, відстеження помилок. Результат – функціюнуючий інтернет магазин.

4. Перелік питань, що підлягають розробці

- проєктування зовнішнього вигляду вебзастосунку;
- розгортання середовища розробки;
- проектування архітектури frontend;
- пошук та завантаження модулів і бібліотек, необхідних для реалізації завдання;
- розробка функціоналу сайту;
- програмна реалізація, тестування та відлагодження вебзастосунку інтернет-магазину.

5. Перелік графічних матеріалів:

Презентація

.

6. Завдання до спеціальної частини

Дослідження питань охорони праці, які безпосередньо пов'язані з діяльністю розробника програмного забезпечення .

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи канд. техн. наук, доцент Горбань Гліб Валентинович

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

Антонов В'ячеслав Олександрович

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «____» _____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: «Програмне забезпечення інтернет-магазину товарів для косметологів. Клієнтська частина»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	12.11.2021	13.11.2021	
2.	Огляд літератури за темою роботи	15.11.2021	17.11.2021	
3.	Аналіз предметної області	18.11.2021	21.11.2021	
4.	Вибір технологій	02.01.2022	28.04.2022	
5.	Моделювання та конструювання ПЗ	05.01.2022	30.05.2022	
6.	Кодування, тестування та апробація розробленого ПЗ.	07.01.2022	30.05.2022	
7.	Реалізація скриптів та запитів на сервер	15.02.2022	30.05.2022	
8.	Розробка спеціальної частини з охорони праці	10.05.2022	28.05.2022	
9.	Відгук керівника КРБ	08.05.2022	20.05.2022	
10.	Оформлення КРБ та презентації	21.05.2022	21.05.2022	
11.	Попередній захист	22.05.2022	22.05.22	
12.	Рецензування	14.06.2022	14.06.2022	
13.	Завершення оформлення КРБ та презентації	15.06.2022	19.06.2022	
14.	Захист кваліфікаційної роботи	28.06.2022	28.06.2022	

Розробив студент Антонов В’ячеслав Олександрович _____

(прізвище, ім’я, по батькові) *(підпис)*

«__» _____ 20__ р.

Керівник роботи

Канд. техн. наук, доцент Горбань Гліб Валентинович _____

(посада, прізвище, ім’я, по батькові) *(підпис)*

«__» _____ 20__

ЗМІСТ

Примечание [ПW1]:

Примечание [ПW2]:

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

VSC – Visual Studio Code

CSS – Cascading Style Sheets

SASS – Syntactically Awesome Stylesheets

SPA – Single page application

MPA – Multi page application

MVC – Model, View та Controller

CDN – Content Delivery Network

ВСТУП

Програмування клієнтської частини вебзастосунку є однією з основних частин в створенні сайту. Адже саме з нею користувач проводить більшість часу, взаємодіючи з різними елементами застосунку. Користувач не буде купувати товари в застарілому інтерфейсі. У теперішній час у сайту є 5 секунд для того, щоб залишити клієнта у себе на сторінці. За ці 5 секунд сторінка повинна повністю завантажитися та справити гарне перше враження на користувача. Не менш важливо створити сайт адаптивним. Тому що вже багато років мобільний трафік переважає на сайтах Інтернет-магазинів та й інших застосунків. У роки цифрової діджиталізації треба бути готовим до постійних оновлень інтерфейсу. З кожним роком виходять нові вимоги до побудови класів та інтерфейсу Front-End частини. Інтерфейси все більш прагнуть бути user-friendly, тому потребують оновлення.

Під час проєктування даної частини кваліфікаційної роботи необхідно дослідити такі задачі для реалізації клієнтської частини інтернет-магазину товарів для косметологів:

- вибір середовища розгортання та налаштування конфігурацій цього середовища для подальшого проєктування;
- дослідження існуючих модулів та бібліотек JavaScript;
- проектування та реалізація frontend-архітектури.

1 ВИКОРИСТАНІ ТЕХНОЛОГІЇ

1.1 Середовище розробки

Visual Studio Code було обрано головним середовище розробки клієнтської частини інтернет-магазину товарів для косметологів. VSC є дуже прогресивним і універсальним середовищем розробки. Він є безкоштовним та має багато плагінів для кожної мови програмування. Так як в проєкті використовується декілька мов програмування та багато технологій, то VSC є ідеальним вибором не тільки для цього застосунку, а й для сучасного frontend-розробника. Популярність VSC, полягає в тому, що в ньому є все, що будь-який програміст очікує від будь-якого редактора коду з деякими додатковими та корисними функціями. Він легкий, швидкий, з відкритим вихідним кодом та кросплатформний, а також інші цікаві функції дають йому додаткову перевагу перед іншим редакторами, такі як WebStorm або Eclipse [5].

Переваги Visual Studio Code:

– Кросплатформний;
– Підтримує багато мов програмування;
– Надає документацію по мовам програмування;
– Повністю безкоштовний;
– Багато додаткових плагінів від різних компаній які кожен день підтримують їх;
– Вбудована інтеграція Git;
– Кастомізація.

Варто зазначити, що часто плутають Visual Studio Code та Visual Studio, але це принципово різні середовища розробки, які можуть виконувати одні й ті самі функції але й мають багато своїх переваг.

Перелік VSC плагінів які були використанні для реалізації проекту:

– All Autocomplete;

– Auto Close Tag;
– Auto Complete Tag;
– Auto Rename Tag;
– Beautify;
– Code Runner;
– Import Cost;
– JavaScript (ES6) code snippets;
– Jshint;
– Live Server;
– Multiple clipboards for VSCode;
– Sass;
– Theme - Oceanic Next;
– Vscode-icons;
– ESLint.

1.2 HTML5

Жоден вебдодаток, як наслідок і Front-End програміст не може обійтися без HTML розмітки. Існує багато фрейморків, які мають свої мови розмітки, але всі вони потім перетворюються у HTML, як наприклад мова JSX у фрейморку ReactJS. Це відбувається тому що браузери розуміють тільки HTML як мову розмітки. HTML5 отримала ряд доповнень та нових можливостей. До складу робочої групи її розробників входять представники таких гігантів IT-сфери, як Google, Apple, IBM, Microsoft та інші.

Переваги HTML5 над попередніми версіями:

– Були додані нові теги і атрибути, які дозволяють значно простіше розробляти і просувати сайти;

– Додані теги структурування, які допомагають розділяти сторінки на зручні для роботи блоки (header, footer, section, nav, article, aside, figure);

– Прискорене завантаження сторінок сайту;

– Немає необхідності підключати сторонні розширення до веб-браузеру для відтворення аудіо і відео доріжок, досить використовувати призначені для цього теги.

Рисунок 1.1 – Логотип HTML5

1.3 CSS, SASS та SCSS

Якщо HTML – це скелет сайту, то CSS вважають зовнішнім виглядом. Це мова стилів, за допомогою якої описують вигляду документів (як і де відображати елементи веб-сторінки), написаних мовами розмітки даних. Одна з головних переваг використання CSS - це можливість розділити зміст сторінки від її оформлення. Таке розділення дозволило покращити сприйняття та доступність змісту, забезпечити більшу гнучкість та контроль за відображенням змісту в різних умовах, зробити зміст більш структурованим та простим, прибрати повторення та ін. Власне це ж і була основна мета створення цієї технології. SASS є дуже потужною та модифікованою версією CSS. Вона дозволяє робити більше роботи пишучи, менше коду. Це в декілька разів пришвидшує роботу програміста. Тому усі сучасні програмісти використовують SASS. Для того щоб використовувати цю скриптову метамову, потрібно мати компілятор. Тому що все одно потім увесь SASS код перетворюється у CSS. Також у SASS є свій “діалект” це

SCSS. Під діалектом мається на увазі різновид мови. Відмінності між ними невеликі, проте порушення правил синтаксису не дозволить скомпілювати файл. У SASS-синтаксисі немає фігурних дужок, вкладеність елементів у ньому реалізовано за допомогою відступів, а стильові правила обов’язково відокремлені новими рядками. Незалежно від синтаксису, SCSS сумісний з CSS. Тобто будь-який CSS обов’язково буде дійсним SCSS-кодом [1].

Рисунок 1.2 – Логотип SASS

1.4 JavaScript та jQuery

JavaScript – є “м’язами” цього проекту. JavaScript – це мова програмування, яка є на даний момент одноосібним лідером і монополістом у frontend-розробці. Інтерпретована, об'єктно-орієнтована мова з функціями першого класу, найвідоміша скриптова мова для вебсторінок, але також використовується в багатьох не браузерних оточеннях. JavaScript запускається на стороні клієнта Інтернету, який може використовуватися для створення/програмування того, як вебсторінки будуть поводитися при настанні будь-яких подій. JavaScript легко вивчити, а також це потужна скриптова мова, що широко використовується для контролю поведінки вебсторінок. [3]. JavaScript має дуже багато фрейморків, плагінів та бібліотек, які модифікують і так дуже потужну і сучасну мову програмування. Одним з цих бібліотек є jQuery. JQuery — це швидка, невелика і багатофункціональна

бібліотека JavaScript. Вона значно спрощує такі речі, як обхід і маніпуляції з документами HTML, обробку подій, анімацію та AJAX-запити, завдяки простому у використанні API, який працює в багатьох браузерах. Завдяки поєднанню універсальності та розширюваності jQuery змінив спосіб написання JavaScript мільйонами людей. JQuery хоча і стара бібліотека, але дуже потужна. Вона може відправляти та отримувати AJAX-запити, як усі сучасні бібліотеки такі як Vue Js, ReactJS, AngularJS. На рис. 1.3 зображено рейтинг кращих JavaScript бібліотек. JQuery вже не займає лідерські позиції але все тримається у середині. На першому місці бібліотека Axios яка використовується у багатьох сучасних проектах та цей застосунок не є виключенням. Завдяки неї за 1 строчку коду можно відправити або отримати запит з серверу.

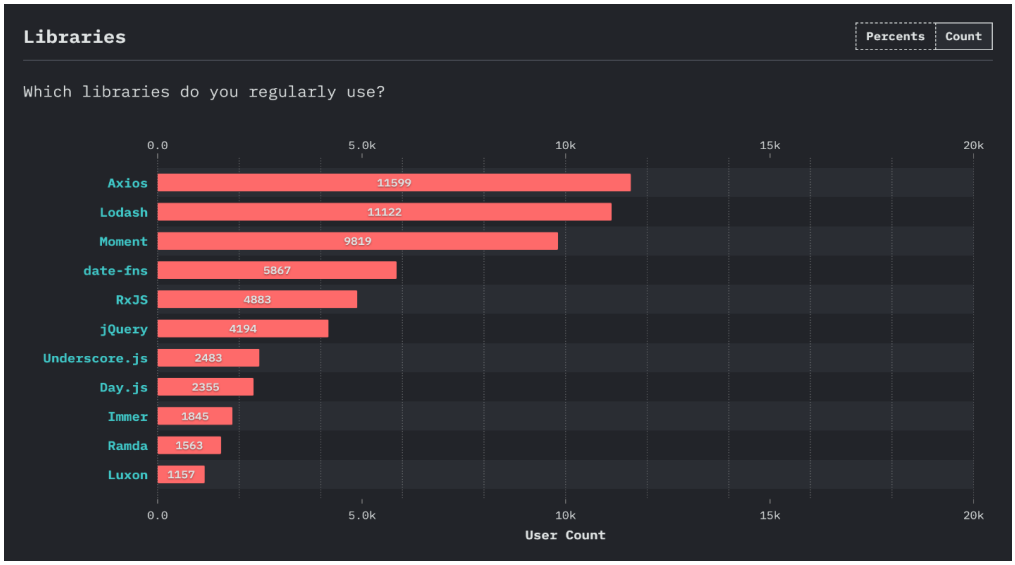

Рисунок 1.3 – Рейтинг JavaScript бібліотек

На рис. 1.4 зображено рейтинг кращих JavaScript-фреймворків. Можна побачити, що ReactJs та Svetle отримують лідерські позиції.

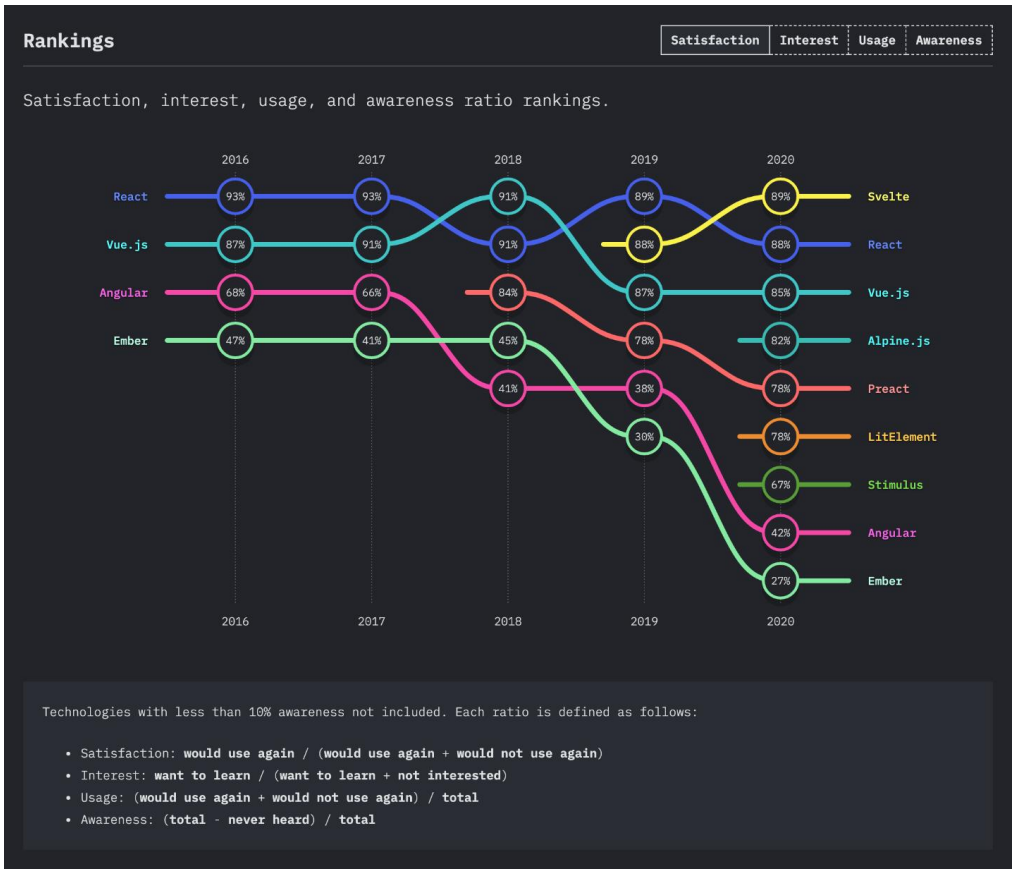

Рисунок 1.4 – Рейтинг JavaScript фреймворків

Слід не плутати JavaScript та Java. Мовою Java реалізується backend-частина застосунку, а мовою JavaScript реалізується frontend-частина.

1.5 Thymeleaf

Thymeleaf у проекті використовується для того щоб поєднати backend-частину із frontend-частиною. Так як ці частини написані на різних мовах програмування, то їм потрібний спільний шаблонізатор. Thymeleaf — це сучасний серверний механізм шаблонів Java як для веб, так і для автономних середовищ. Основна мета Thymeleaf – привнести елегантні природні шаблони у робочий процес розробки – HTML, який можна правильно

відображати у браузерах, а також працювати як статичні прототипи, забезпечуючи більш міцну співпрацю в командах розробників. Завдяки модулям для Spring Framework, як інтеграції хосту з нашими інструментами, і підключенням вашої власної функціональності, Thymeleaf ідеально підходить для сучасної веброзробки HTML5 JVM — хоча він може зробити набагато більше [4]. Шаблони HTML, написані на Thymeleaf, все ще виглядають і працюють як HTML, дозволяючи фактичним шаблонам, які запускаються у вебзастосунку, продовжувати працювати як корисні артефакти дизайну.

Рисунок 1.5 – Логотип Thymeleaf

1.6 Bootstrap

Bootstrap — це відкритий і безкоштовний HTML, CSS і JS-фреймворк, який використовується веброзробниками для швидкого адаптивного дизайну вебсайтів і вебзастосунків [4]. Фреймворк Bootstrap використовується по всьому світу не тільки незалежними розробниками, а іноді і цілими компаніями. Основна область його застосування – це фронтенд розробка сайтів та інтерфейсів адміністративних панелей. Серед аналогічних систем (Foundation, UIkit, Semantic UI, InK та ін.) фреймворк Bootstrap є найпопулярнішим. Це пов'язано з тим, що він дозволяє верстати сайти в кілька разів швидше, ніж на чистому CSS і JavaScript. А в нашому світі час –

це дуже цінний ресурс. Ще один аспект – доступність. Вона зводиться до того, що надає можливість навіть веб-розробнику-початківцю (без глибоких знань і достатньої практики) створювати досить якісні макети. Фреймворк Bootstrap – це набір набір CSS та JavaScript файлів. Щоб його використовувати ці файли, необхідно просто підключити до сторінки. Після цього стануть доступні інструменти фреймворку: колонкова система (сітка Bootstrap), класи і компоненти [12]. В даному проекті цей фреймворк використовується 4 версії. Так як багато плагінів цеї версії Bootstrap написані на JQuery та й не має багато переваг 5 версії перед 4 саме для цього проєкту.

1.7 Ajax запити

AJAX означає асинхронний JavaScript і XML. У двох словах, це використання об’єкта XMLHttpRequest для зв’язку з серверами. Він може надсилати та отримувати інформацію в різних форматах, включаючи JSON, XML, HTML та текстові файли. Найпривабливішою характеристикою AJAX є його «асинхронна» природа, що означає, що він може спілкуватися з сервером, обмінюватися даними та оновлювати сторінку без необхідності оновлювати сторінку [6].

Дві основні функції AJAX дозволяють робити наступне:

- робити запити на сервер без перезавантаження сторінки;
- отримувати та працювати з даними з сервера.

В проекті AJAX запити будуть надсилатися за допомогою бібліотеки jQuery. Ці запити є основою SPA-застосунків. SPA (односторінковий застосунок) — це реалізація вебпрограми, яка завантажує лише один вебдокумент, а потім оновлює основний вміст цього окремого документа за допомогою API JavaScript, таких як XMLHttpRequest та Fetch, коли має бути показаний інший вміст.

Таким чином, це дозволяє користувачам використовувати вебсайти без завантаження нових сторінок із сервера, що може призвести до підвищення продуктивності та більш динамічного досвіду, з деякими недоліками, такими як SEO, більше зусиль, необхідних для підтримки стану, впровадження навігації та значущої продуктивності. моніторинг. Саме за допомогою перелічених вище фреймворків Vue Js, ReactJS, AngularJS створюються SPA-застосунки. Вебзастосунок Інтернет-магазину товарів для косметологів буде MPA. Багатосторінкові програми, або MPA, щоразу запитують відтворення нової сторінки з сервера в браузері. Вони ідеально підходять для більших застосунків, ніж SPA, і через кількість вмісту вони мають різні рівні інтерфейсу. Вони часто відомі як «традиційний» спосіб розробки застосунків, і хоча існує кілька рівнів інтерфейсу користувача, нещодавно це було вирішено завдяки розробкам AJAX. Однак AJAX дозволяє передавати велику кількість складних даних між серверами та браузерами; це також створює певні проблеми. Завдяки можливості передавати дані є новий рівень складності, який часто кидає виклик розробникам JavaScript. Модель класичних застосунків для мережі (зліва) в прямому порівнянні із застосуванням AJAX (праворуч) можна побачити на рис. 1.6.

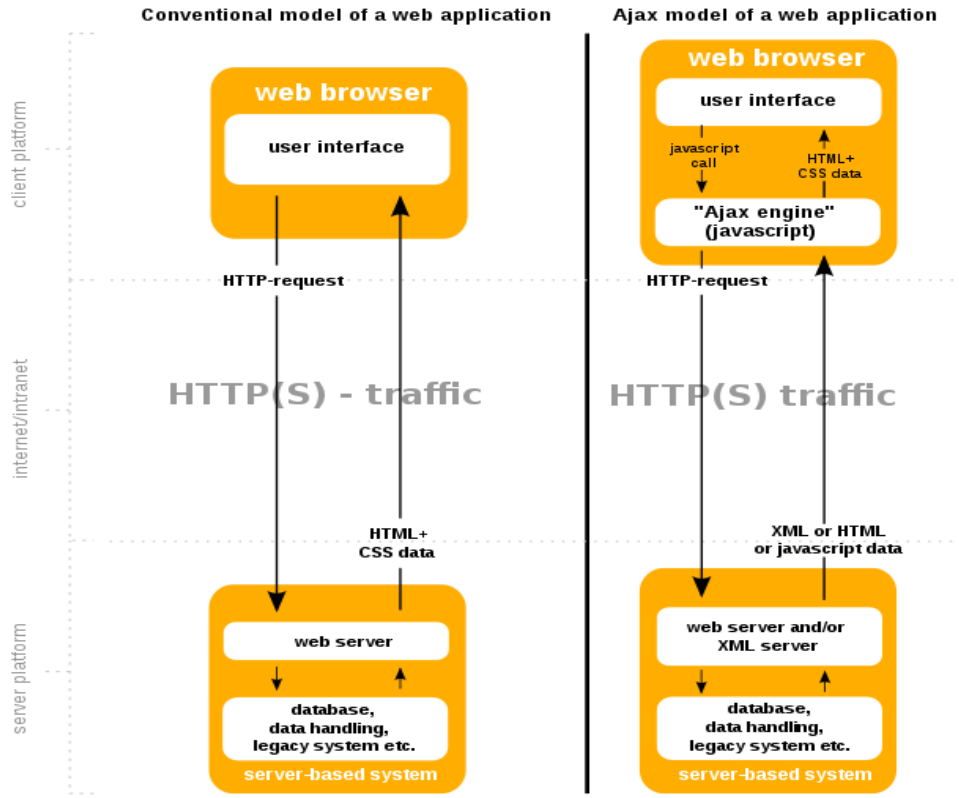

Рисунок 1.6 – Модель порівняння класичного застосунку та застосунку із використанням AJAX запитів

1.8 MVC модель

MVC це архітектурна схема, яка складається з трьох компонентів Model, View та Controller, що ефективно відокремлює бізнес логіку від користувальницького інтерфейсу програми[7]. MVC складається з:

- **Модель:** простими словами, Модель містить дані про програму. Тут вказуються дані, які мають бути важливими для відображення вимог щодо доступу та інших перевірок;

- **Вид:** Перегляд відображає інформацію в компоненті Модель. Будь-який запит від користувача також розпізнається та надсилається до компонента Controller;
- **Контролер:** Контролер відповідає за надання інформації, присутньої у Моделі, компоненту «Вид» та інтерпретацію відповідей користувачів, які розпізнаються компонентом «Перегляд».

Ця модель має такі переваги:

- MVC має архітектуру для надання декількох переглядів;
- Розробка застосунку, який завантажує надзвичайно швидкіші темпи;
- Модифікація клієнтського інтерфейсу не впливає на бізнес логіку;
- Розробка великих програм з певною структурою.

Схематичне зображення MVC моделі зображено на рис. 1.7.

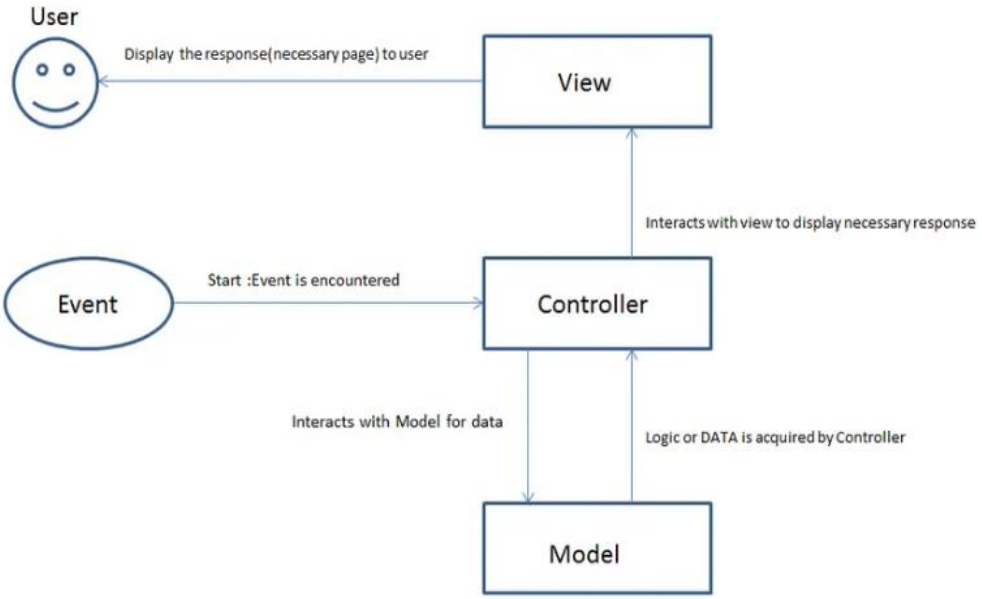

Рисунок 1.7 –зображення MVC у діаграмі

Висновки до розділу 1

В результаті проробленої роботи було виявлено, що для реалізації вебзастосунку – кращим та актуальним рішенням є мова програмування JavaScript. Обрана мова програмування є об’єктно-орієнтованою, що дозволяє розробляти модулі для багатого використання коду. Також необхідно залучитись до додаткових бібліотек та фреймворків: Bootstrap, jQuery, Thymeleaf та SCSS. Було визначено основні переваги та причини використання цих бібліотек. Додаток буде будуватися відповідно з технологією MVC та із використанням AJAX запитів.

2 РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

2.1 Адаптивна верстка та структура проекту

Кодування кожного вебзастосунку починається із верстки сторінок. Із використанням шаблонізатора Thymeleaf, верстка ділиться на модулі. Кожен модуль можливо застосовувати повторно на інших сторінках. Це дуже скорочує та спрощує програмний код застосунку. Застосунок має дві частини: адміністративна та клієнтська. Кожна частина має свою папку з файлами, які належать до певної частини застосунку. Клієнтська частина має 32 зверстані HTML-сторінки. Серверна частина має 41 зверстану HTML сторінку.

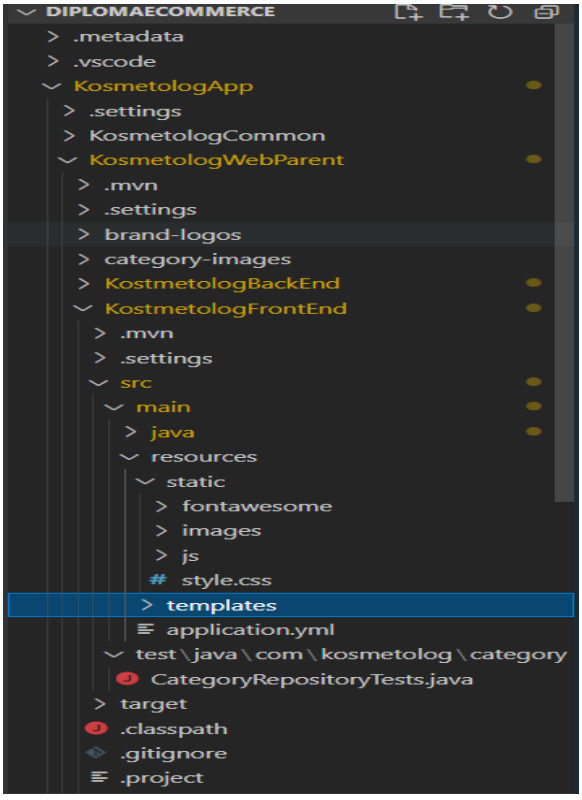

Рисунок 2.1 – Файлова структура застосунку

Всі ці сторінки використовують шаблони Thymeleaf. Наприклад на головній сторінці є header та search nav який використовується безліч разів на інших сторінках. У файлі index.html це займає всього 2 строчки, що зображено на рис. 2.2.

```
<div th:replace="navigation :: header_menu"></div>
<div th:replace="navigation :: search_nav"></div>
```

Рисунок 2.2 – Шаблони header та search nav

Але звичайно ж є header та search nav не може вміститись у дві строчки. Сам код цього шаблону написаний у navigation.html що й видно на рис. 2.2. Фрагменти header та search nav займають 64 строчки. Якщо на кожній сторінці використовувати повторно усі ці строчки замість шаблонів, то файли будуть розростатися у геометричній прогресії, а це буде дуже сильно впливати на швидкість застосунку. Це один шаблон, а їх ще багато і тих які використовуються на усіх сторінках застосунку.

```
<div th:fragment="header_menu">
    <nav class="navbar navbar-expand-lg bg-main-nav navbar-light">
        <div class="container">
        <a class="navbar-brand" th:href="@{/}">
            <img th:src="@{${SITE_LOGO}}" />
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-targ
            <span class="navbar-toggler-icon"></span>
        </button>
        <div  class="collapse navbar-collapse" id="topNavbar">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="">Покупки</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/login}">Вхід</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/register}">Реєстрація</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="">Контакти</a>
                </li>
                <th:block sec:authorize="isAuthenticated()">
                    <li class="nav-item">
                        <b><a class="nav-link" th:href="@{/account_details}"
                                sec:authentication="principal.fullName"></a></b>
                    </li>
```

Рисунок 2.3 – Частина коду елемента header розташованого в navigation.html

Також кожна сторінка має шаблоний header у якому підключені усі додаткові бібліотеки. Він міститься тільки у fragments.html, що зображено на рис. 2.4. В усіх інших файлах(де це потрібно) він використовується шаблоном що зображено на рис. 2.5.

```
<head th:fragment="page_head(title, remove)" th:remove="${remove}">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0, minimum-scale=1.0">

    <title>[[${title}]] - [[${SITE_NAME}]]</title>

    <link rel="stylesheet" type="text/css" th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" />
    <link rel="stylesheet" type="text/css" th:href="@{/fontawesome/all.css}" />
    <link rel="stylesheet" type="text/css" th:href="@{/style.css}" />
    <script type="text/javascript" th:src="@{/webjars/jquery/jquery.min.js}" ></script>
    <script type="text/javascript" th:src="@{/webjars/bootstrap/js/bootstrap.min.js}" ></script>
</head>
```

Рисунок 2.4 –Head в fragments.html

```
<head th:replace="fragments :: page_head('Shopme', 'none')" />
```

Рисунок 2.5 – Шаблон тегу head в index.html

Крім додаткових бібліотек jQuery та Bootstrap, до тегу head слід підключити файл зі стилями style.css та файл із шрифтами all.css. При завантаженні проекту на хостинг ці файли будуть мінімізуватися. Це означає, що кожен пробіл, який займає 1 байт вільної пам’яті, буде видалений. Це економить місце на сервері та сторінка завантажується набагато швидше. Це робить автоматично плагін для VSC під назвою SASS. Файли додаткових бібліотек вже мінімізовані. Вони підключаються із CDN-серверу. Це зручно робити, тому що не треба завантажувати ці файли на власний сервер, але це може негативно вплинути на швидкодію. Але не в усіх випадках, тому треба протестувати швидкодію завантаження сторінки вебзастосунку після завантаження на хостинг. Із використанням класів Bootstrap було зверстано багато елементів. Декілька з них можливо побачити на рис. 2.6. А саме навігаційне меню, слайдер, форма пошуку товарів та вибірка категорії товарів.

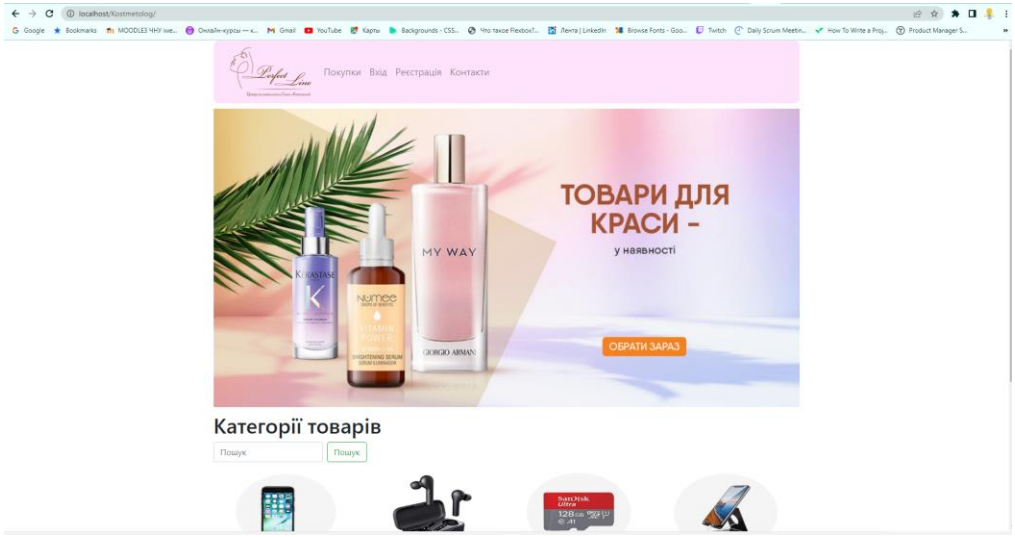

Рисунок 2.6 – Зверстана головна сторінка

Ключовим фактором у написанні розмітки сторінок – це можливість їх до адаптації під різні девайси. Тому що перегляд Інтернет-магазинів у браузерах із використанням мобільних девайсів вже багато років переважають перегляд із використанням комп'ютерів. Усі сторінки є повністю адаптивними під різні типи девайсів. Тестування цих сторінок описано у розділі 2.3 під назвою тестування користувацького інтерфейсу. На рис 2.8 зображений приклад відображення головної сторінки на одному із найпопулярніших девайсів IPhone XR із розмірами 414 на 896 пікселів. Уся адаптивність у Bootstrap реалізований із використанням технології flex-box яка міститься у HTML5 та використання стилей @media у CSS. Сайт заповнений тестовими картинками.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Рисунок 2.7 – Сітка класів адаптивності у Bootstrap

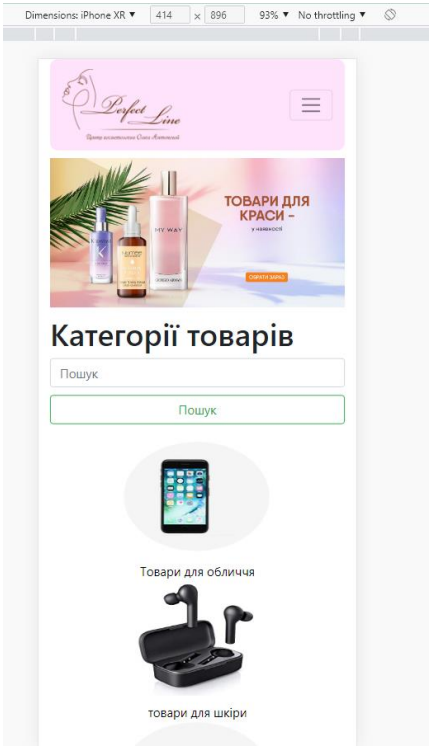

Рисунок 2.8 – Приклад відображення головної сторінки на IPhone XR із розмірами екрану 414 на 896 пікселів

2.2 Реалізація скриптів та запитів на сервер

Усі скрипти реалізовані за допомогою бібліотеки jQuery. Кожен скрипт поділений на модулі. У кожному модулі є реалізація певних функцій схожих за змістом. Усього таких модулів 14. Вони складаються з файлів, які розташовані у frontend та backend-частинах. Скрипти які розташовані у frontend-частині:

1. add_to_cart.js – скрипт із однією функцією додавання товару до корзини;

2. common_customer_form.js – скрипт із перевірками на збіг паролю користувача та додаванням країни користувача;

3. common_modal.js – скрипт із функціями помилкових модальних вікон. А саме коли щось піде не так у застосунку, то буде працювати ці функції. Він приймає зміну, в якій буде міститися помилка, тому при різній помилці буде інше модальне вікно;

4. quantity_control.js – скрипт перевірки на кількість доданих продуктів у кошик. У кошик не можна додати більше 5 продуктів;

5. quantity_control.js – найбільший скрипт у frontend частині. Він має багато функцій пов’язаних із кошиком. Серед них є AJAX запити на видалення товарів з кошика, додавання кількості продуктів.

Скрипти які розташовані у backend-частині:

1. common_form_country_state.js – скрипт загрузки міст із країни яку обрав користувач. Міста загружаються із серверу який відповідє JSON файлом із містами певної країни;

2. common_form.js – скрипт загрузки та перевірки на розмір картинки яку користувач завантажує на сервер;

3. common_list.js – скрипт із модальним вікном підтвердженням на видалення користувача, категорії, товару.

4. common.js – скрипт який працює при виході з аккаунту. При виході з акаунту скриваються навігаційні меню які доступні авторизованому користовачу;

5. countries_setting.js – великий скрипт із багатьма функціями та AJAX запитами. Завдяки цьому скрипту можливо додавати та оновлювати інформацію про нові країни на сервер завдяки AJAX запитам, та інші маніпуляції;

6. product_form_details.js – скрипт додавання або видалення опису товару;

7. product_form_images.js – скрипт додавання або видалення фото товару. Адміністратор має можливість завантажувати безліч фото товару;

8. product_form_overview.js – скрипт із формою додавання нового товару у вебзастосунок із перевірками на коректність заповнених даних, щоб сервер отримав коректні дані;

9. states_setting.js – скрипт із можливістю додавання міста доставки або проживання користувача. Скрипт використовує AJAX запити.

Усі функції розбиті у модулі для того щоб можливо було читати код. Якщо усі функції були б в одному файлі, то працювати та тестувати вебзастосунок було б набагато складніше. Кожну функцію можливо використовувати повторно у інших модулях.

Для того щоб сервер не отримував хибні запити або запити із невірними даними у вебзастосунку реалізована валідація форм на клієнтській частині. Це не навантажує сервер і прискорює сайт. У HTML5 валідацію деяких форм можливо об’явити у самому тегу. Наприклад при вводі даних типу електронної пошти, номеру телефону є відповідна властивість, що відповідає CSS псевдокласам :valid та :invalid. Приклад властивостей зображено на рис. 2.9.

```
<input type="email" name="email" class="form-control" placeholder="E-mail" required />

<input type="password" name="password" class="form-control" placeholder="Password" required />
```

Рисунок 2.9 – Приклад властивостей які мають валідацію

Це дуже зручно, тому що не треба писати додатковий код мовою JavaScript.

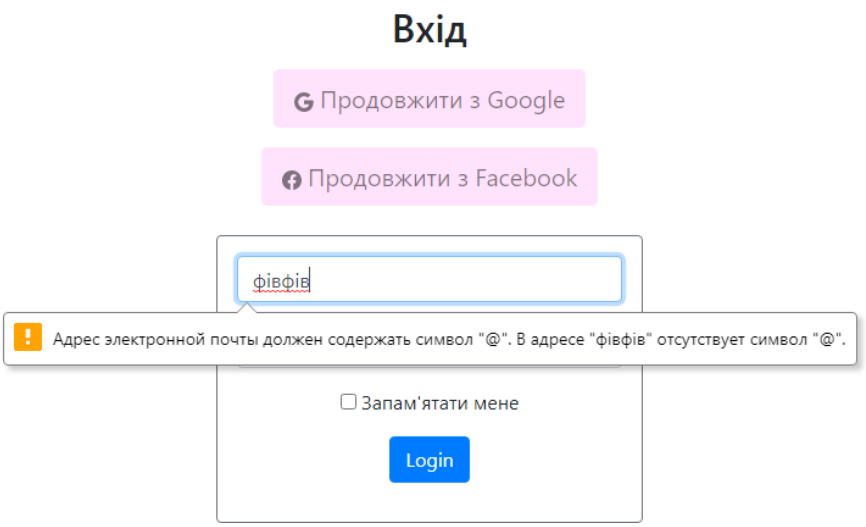

Рисунок 2.10 – Приклад валідації за допомогою властивостей тегів HTML

Також є функції валідації, що написані за допомогою бібліотеки jQuery. Наприклад, у файлі common_form.js перевіряється розмір картинки товару, яку адміністратор завантажує на сервер. Перевірка відбувається у функції checkFileSize яка приймає параметр вхідного файлу. Розмір файлу не повинен бути більший за змінну MAX_FILE_SIZE яка зберігається на сервері.

```
25  function checkFileSize(fileInput) {
26      fileSize = fileInput.files[0].size;
27
28      if (fileSize > MAX_FILE_SIZE) {
29          fileInput.setCustomValidity("You must choose an image less than " + MAX_FILE_SIZE + " bytes!");
30          fileInput.reportValidity();
31
32          return false;
33      } else {
34          fileInput.setCustomValidity("");
35
36          return true;
37      }
38  }
```

Рисунок 2.11 – Функція checkFileSize яка перевіряє розмір файлу

2.3 Тестування клієнтської частини

Тестування вебзастосунку є важливим процесом у розробці вебзастосунку. Історія налічує багато провальних проєктів які випускалися з

недостатнім тестуванням. На тестування принято не виділяти багато коштів та часу, але це дуже шкодить кінцевому продукту. Вебзастосунок пише людина, а не машина, тому неможливо обійтись без помилок. Навіть якщо програму пишуть кращі у світі програмісти, то це не привід для пропуску стадіїї тестування. Ще є теорія, що коли програміст пише свій участок коду, то він не бачить помилки, навіть тестуючи сам свій код. Це може бути зв’язано з тим, що розробник має дещо інші навички тестування та бачення коду.

Усі тестування клієнтьскої частини будуть проводится методом білої скриньки, так як є доступ до коду. Також тестування буде мануальним. Це означає що тести будуть проводить особисто тестувальником, а не написаним застосунком для тестування.

Тестування користувацького інтрефейса буде проходити за такими критеріями:

- Інтерфейс продукту відповідає прототипам;
- Типографіка;
- Відповідність стилю;
- Адаптивність;
- Відповідність стандартам;
- Використання функціоналу;
- Перевірка орфографії;
- Перевірка полів та стандартних елементів;
- Елементи інформування.

Спочатку треба порівняти інтерфейс вебзастосунку із прототипами які описані у розділі 2. Інтрефейс збігається із прототипом.

Треба перевірити усі сторінки на адаптивність на усіх розмірах екрану. Вона є дуже енергозатратною, так як треба передивлятися усі сторінки, під кожним розміром. Тестування адаптивності буде проводитись із використанням Chrome Developer Tools, що інтегровані у браузер Google

Chrome. Сайт заповнений тестовими даними, але це не буде негативно впливати на тестування.. Із зменшенням розміру екрану, зменшується сітка товарів. На рис. 2.12, 2.13, 2.14 буде наведена сторінка товарів на різних популярних девайсів:

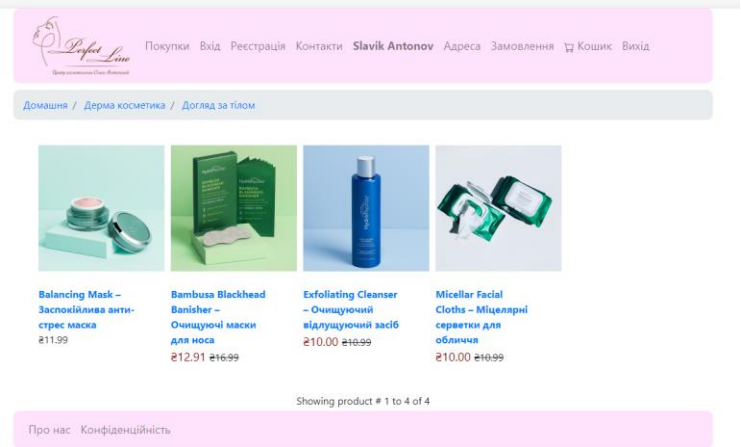

Рисунок 2.12 – Приклад адаптивності на комп’ютері із розміром екрану 1440 на 935 пікселів

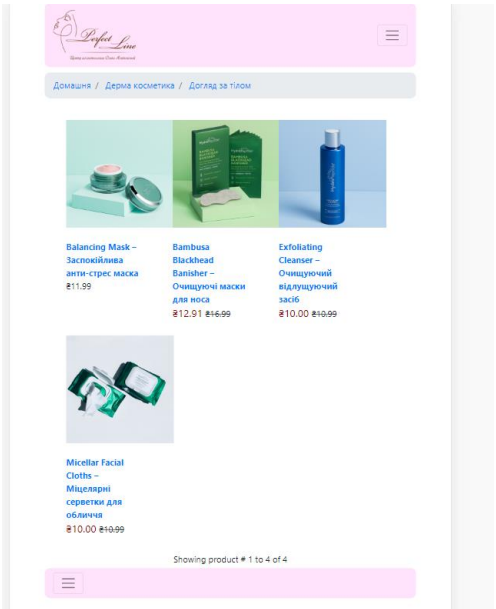

Рисунок 2.13 – Приклад адаптивності на планшеті Ipad Air із розміром 820 на 1180 пікселів

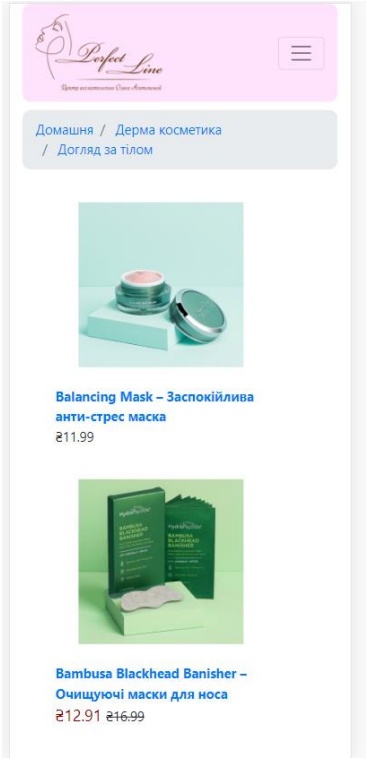

Рисунок 2.14 – Приклад адаптивності на телефоні Sumsung Galaxy S20 із розміром 412 на 915 пікселів

Текст є добре читабельним на основному фоні та інших елементах. Також, чітко зрозуміло де заголовки, а де основний контент. Не використовується понад два шрифти.

Потрібно перевірити чи дизайн продукту відповідає вимогам одного стилю який має компанія PerfectLine. На сайті переважає світлі тона, що притаманно косметологічному центру та косметиці.

Потрібно переконатися, що інтерактивні елементи інтерфейсу поводяться належним чином: кнопки реагують на натискання, налаштування

працюють правильно тощо. Всі елементи повинні бути достатнього розміру, щоб користувач міг без проблем їх використовувати.

Потрібно перевірити текст на орфографічні помилки, оскільки це може зіпсувати репутацію розробників. Також, через помилки в тексті користувачі можуть неправильно зрозуміти призначення того чи іншого елементу.

Тестування того як поле буде поводитись при введенні некоректних даних, якщо вводиться довга назва, при виділенні даних тощо. Вигляд, положення і реакція чек-боксів, радіо-кнопок, полів для спецінформації (номер кредитної карти) та ін.

Перевірка вигляду і розміщення всіх повідомлень про помилки, сповіщень та інші речі, які відносяться до елементів інформування. Приклад модального вікна з помилкою зображено на рис. 2.15.

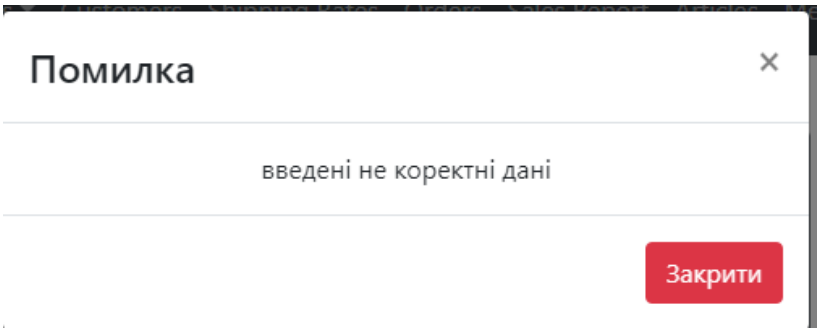

Рисунок 2.15 – Приклад елементу інформування адміністратора о некоректних даних

Усі частини сайту перевірені по усім критеріям тестування користувацького інтерфейсу зазначених вище. На рис. 2.16, 2.17, 2.18, 2.19, 2.20 зображено підтвердження працюючих функцій додатку.

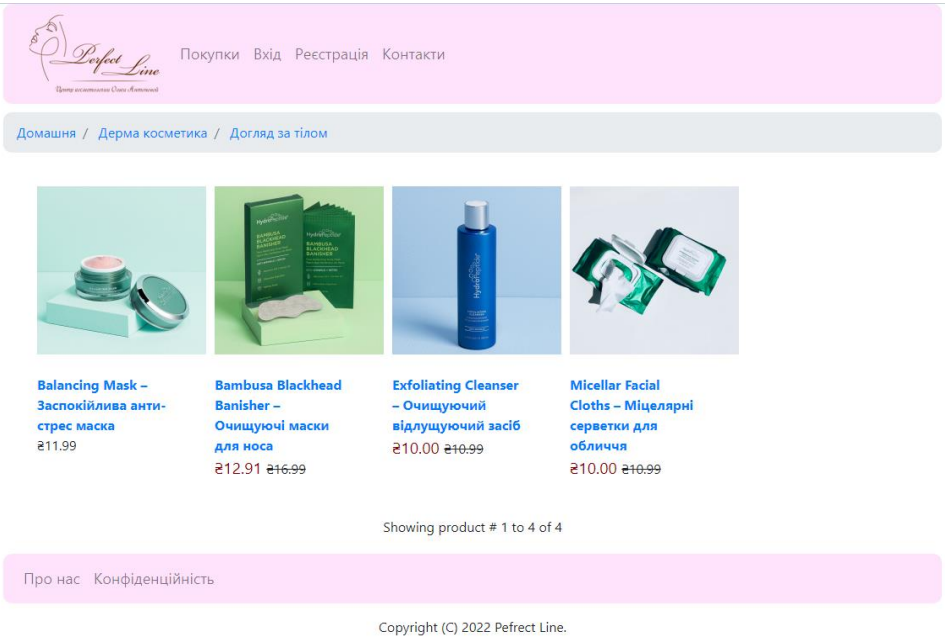

Рисунок 2.16 – Приклад відображення категорії “Догляд за тілом”

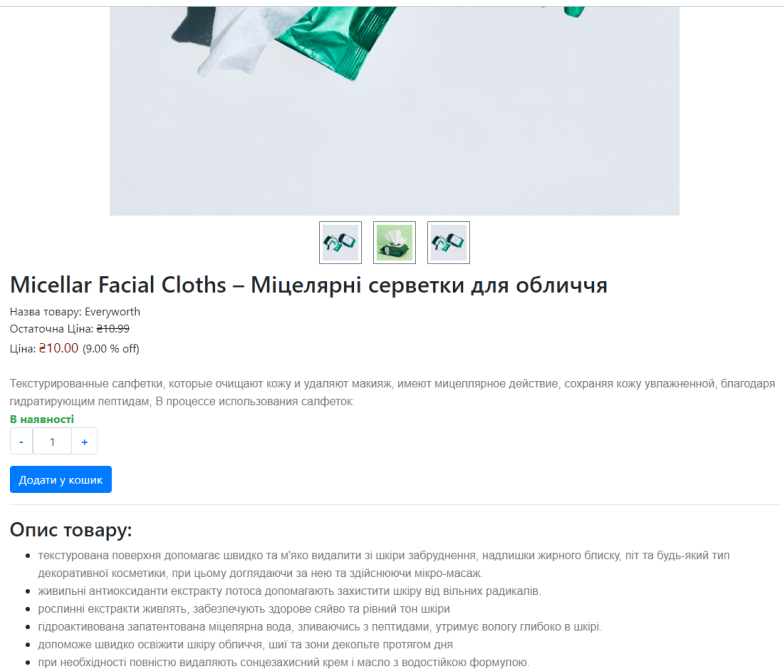

Рисунок 2.17 – Приклад відображення товару “Micellar Facial Cloths”

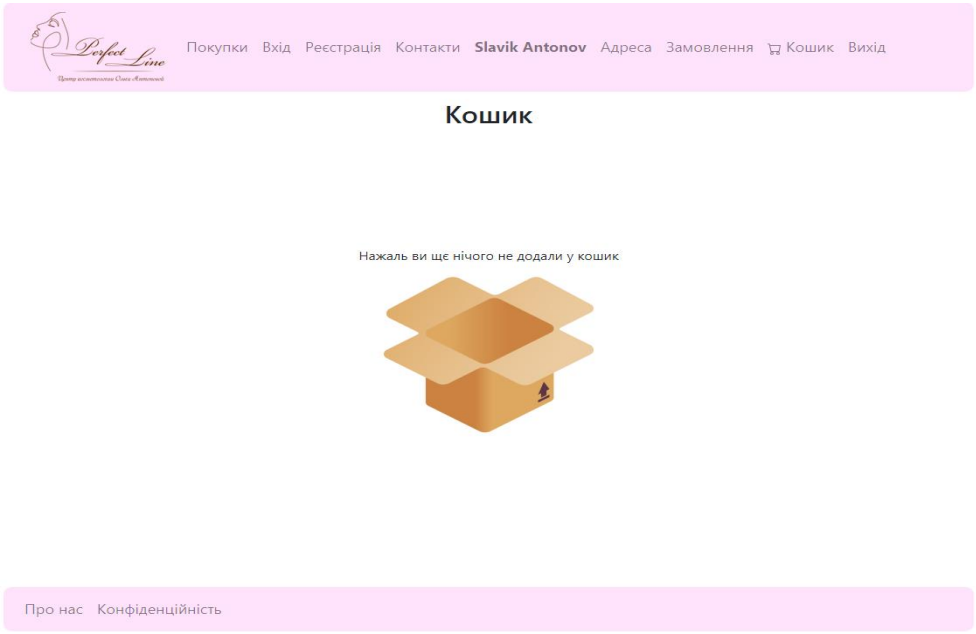

Рисунок 2.18 – Приклад відображення порожнього кошика

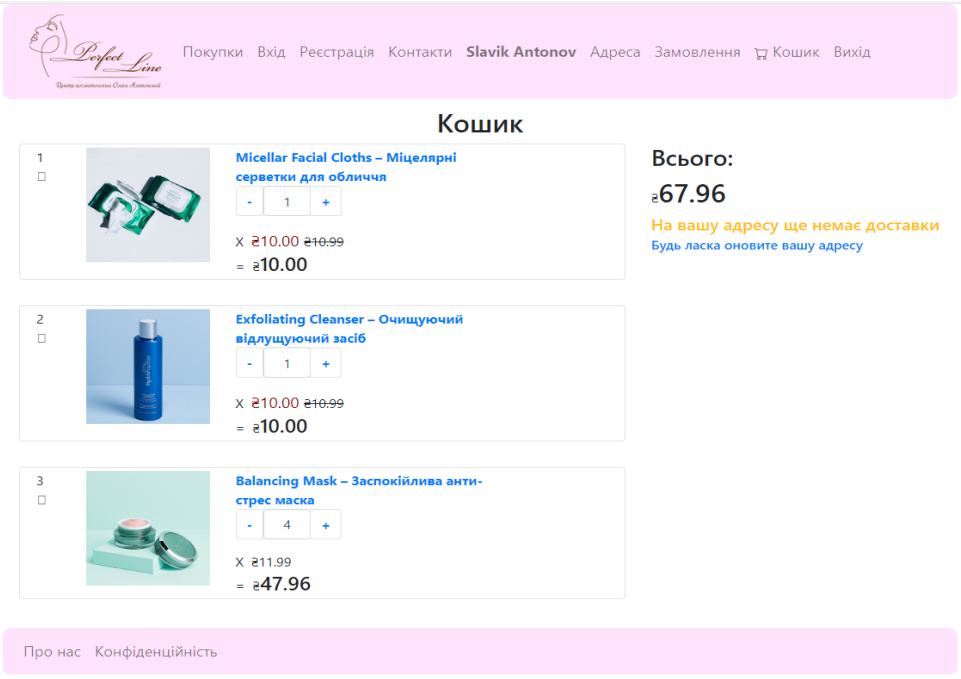

Рисунок 2.19 – Приклад відображення кошика із доданим товаром

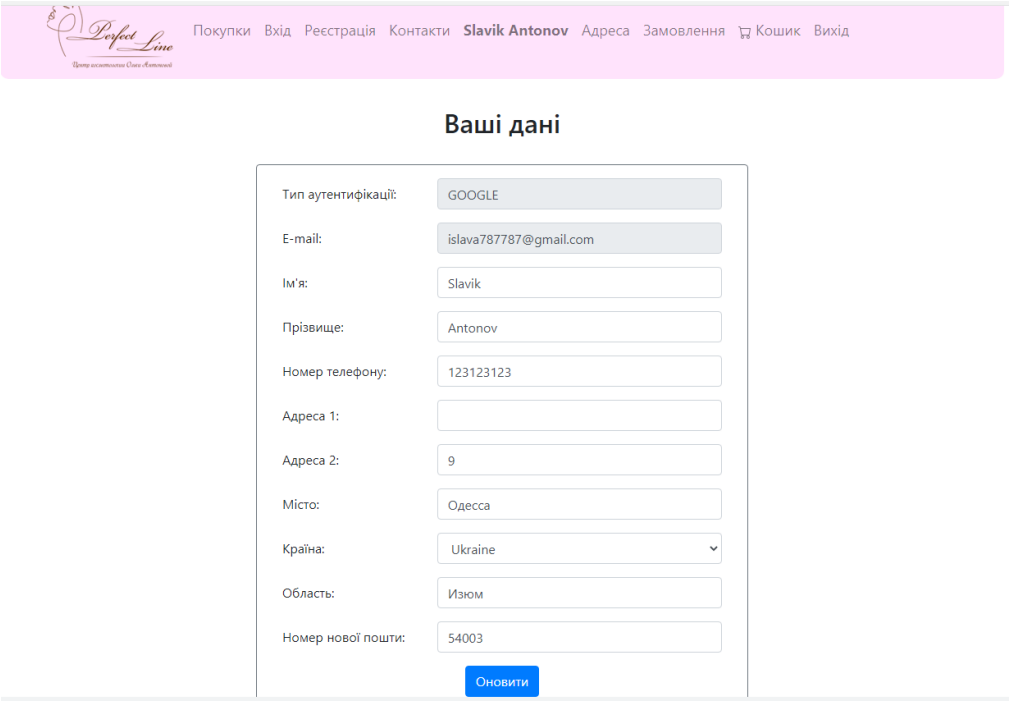

Рисунок 2.20 – Приклад відображення даних користувача

Висновки до розділу 2

Реалізація клієнтської частини вебзастосунку була поділена на 3 чатсини: верстка, написання скриптів та тестування. Із використанням додаткових бібліотек було побудовано адаптивний, user-friendly (дружелюбний до користувачів) користувацький інтерфейс. Його було протестовано за 9 пунктами. Критичних помилок не виявлено.

ВИСНОВКИ

В даній частині кваліфікаційної роботи бакалавра було реалізовано клієнтську частину вебзастосунку: Інтернет-магазину товарів для косметологів.

Дана частина роботи полягала в аналізі існуючих клієнтських моделей та обранні оптимальної для виконання задачі. Було обрано такі інструменти побудови застосунку: HTML5, CSS, SASS, Bootstrap, jQuery, JavaScript, Thymeleaf.

Задачі, які були виконані в процесі дослідження:

- ознайомлення з методами реалізації MVC в jQuery;
- розробка дизайну інтерфейсу користувача;
- розгортання середовища розробки VSC;
- проектування архітектури frontend;
- пошук та завантаження модулів і бібліотек, необхідних для реалізації завдання.

Для виконання поставленої задачі було реалізовано 14 модулів, кожен з яких реалізує власну функціональність.

В результаті виконання даної частини дипломної роботи було створено вебдодаток з користувацьким інтерфейсом, який було протестовано та перевірено за стандартами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Інформація про SASS: https://tokar.ua/read/6672 (дата звернення: 07.06.2022).

2. Інформація про Thymeleaf: https://www.thymeleaf.org/ (дата звернення: 07.06.2022).

3. Інформація про JavaScript: https://developer.mozilla.org/ru/docs/Web/JavaScript/About_JavaScript (дата звернення: 07.06.2022).

4. Інформація про Bootstrap: https://getbootstrap.com/ (дата звернення: 07.06.2022).

5. Інформація про VSC: https://open.zeba.academy/pochemu-vscode-populyaren/ (дата звернення: 07.06.2022).

6. Інформація про AJAX запити: https://itchief.ru/javascript/ajax-introduction (дата звернення: 07.06.2022).

7. Інформація про MVC модель: https://uk.wikipedia.org/wiki/ (дата звернення: 07.06.2022).