Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

Пояснювальна записка

до бакалаврської дипломної роботи
бакалавр
освітньо-кваліфікаційний рівень

на тему «Розробка та наповнення сайту Інтернет магазину з використанням бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS»

Виконала: студентка 4 курсу, групи
1СІ-16б Кабак В.Д.

Напрям підготовки (спеціальність):
6.050201 – «Системна інженерія

Керівник: к.т.н., доц. Софина О.Ю.

Рецензент _____
(прізвище та ініціали)

Вінниця ВНТУ 2020

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет комп'ютерних систем і автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки (спеціальність) 151 – Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
Р. Н. Квєтний
«___» _____ 2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Кабак Вікторії Дмитрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та наповнення сайту Інтернет магазину з використанням бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS

керівник проекту (роботи) Софина О.Ю., к.т.н., доц. каф. АІІТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджена наказом вищого навчального закладу від " __ " лютого 2020 року №_

2. Строк подання студентом проекту (роботи)

3. Вихідні дані до проекту (роботи) WEB-сайт з наповненням; Середовище розробки Visual Studio та Visual Studio Code.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розкрити) сучасний стан питання та основні задачі роботи; технології розробки Інтернет магазину; розробка програмного забезпечення на прикладі Інтернет магазину; висновки; перелік джерел; посилання; додатки (схеми, алгоритми, лістинги програм).

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
блок-схема роботи Інтернет магазину, структура бази даних, діаграма послідовності роботи програми.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання отримав
1	Софина О.Ю., к.т.н., доц. каф. АІІТ		
2	Софина О.Ю., к.т.н., доц. каф. АІІТ		
3	Софина О.Ю., к.т.н., доц. каф. АІІТ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи (проекту)	Строк виконання етапів роботи (проекту)	Примітка
1	Обґрунтування доцільності розробки сайту Інтернет магазину		
2	Технології розробки Інтернет магазину		
3	Розробка програмного забезпечення на прикладі Інтернет магазину “Green Lotus”		
4	Попередній захист		
5	Остаточний захист		

Студент _____Кабак В.Д.
(підпис) (прізвище та ініціали)

Керівник роботи (проекту)_____Софина О.Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

В даній бакалаврській дипломній роботі було розглянуто основні поняття та структуру Інтернет магазинів і технології їх розробки. Проаналізовано архітектуру серверної частини, дизайн зовнішнього вигляду сторінок та наповнення контенту. Розробку Інтернет магазину виконано з використанням мови програмування JavaScript, бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS та об'єктно-орієнтованою системою керування базами даних MongoDB.

В результаті роботи було розглянуто методику створення Інтернет магазинів на прикладі Інтернет магазину “Green Lotus”.

ANNOTATION

In this bachelor's thesis the basic concepts and structure of online stores and technologies of their development were considered. The architecture of the server part, the design of the appearance of the pages and the content are analyzed. The development of the online store was performed using the JavaScript programming language, the JQuery library and the Bootstrap framework on the NodeJS platform and the object-oriented database management system MongoDB.

As a result, the method of creating online stores on the example of the online store "Green Lotus" was considered.

ЗМІСТ

ВСТУП

Актуальність. У наш час все більше людей віддають перевагу покупкам в Інтернеті перед звичайними. Беручи до уваги ситуацію, в якій опинився зараз весь світ, використання онлайн сервісів є як ніколи нагальним та навіть необхідним, особливо коли немає можливості відвідувати магазини вживу.

Загалом процес прийняття рішень покупців за останні роки різко змінився. Вони проводять досить багато часу досліджуючи та порівнюючи в Інтернеті потрібні товари, перш ніж проконсультуватись з продавцем особисто. Щоб залишатись актуальним для свого клієнта, магазини повинні адаптуватись під вимоги та потреби покупців, щоб вони в свою чергу із задоволенням залишались вдома та робили покупки в Інтернеті. І люди не просто купують з дому, вони роблять покупки в будь-якому місці, у якому є wifi або мобільне обслуговування. Зараз це відіграє дуже важливу роль у житті кожного, особливо людей похилого віку.

Проте наявність обох варіантів купівлі товарів та послуг, задовольняє потреби більш широкої клієнтської бази. Більшість підприємств роздрібної торгівлі мають клієнтів, які вважають за краще замовити все в Інтернеті. Інші вважають за краще щось побачити особисто, перш ніж купувати. Тоді є і ті клієнти, які не мають переваги, тобто роблять покупки відповідно до власного розкладу і бажань.

Завдяки інтернет магазинів прибуток більше не обмежується кількістю клієнтів, які можуть фізично відвідувати торгові приміщення. Магазин може вільно продавати в містах, штатах і навіть за кордоном, знімаючи всі географічні обмеження. Інтернет магазин також дозволяє обслуговувати покупців, яким зручніше переглядати та купувати в ті часи, коли торгові місця традиційно не відкриті. Інтернет покупки можуть заощадити час як для

покупця, так і для роздрібної торгівлі, зменшивши телефонні дзвінки про доступність, технічні характеристики, години роботи та іншу інформацію, яку легко можна знайти на сторінках компанії та товарів.

Метою дослідження є аналіз існуючих технологій програмування для розробки Інтернет магазину, що відповідає актуальним запитам та задоволенню потреб з безпосередньою можливість дистанційного замовлення та поліпшення інтерфейсу для зручності у користуванні, інтуїтивно зрозумілим, і орієнтованим на користувача.

Задачі дослідження:

- обґрунтування актуальності Інтернет магазину;
- розглянути основні поняття Інтернет магазинів;
- проаналізувати технологію створення зовнішнього вигляду Інтернет магазину;
- проектування структури Інтернет магазину;
- програмна реалізація прикладу Інтернет магазину;
- аналіз результатів тестування.

Об'єктом дослідження є процес створення Інтернет магазину та його наповнення.

Предметом дослідження є Інтернет магазин з технологіями програмування JavaScript, бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS.

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ САЙТУ ІНТЕРНЕТ МАГАЗИНУ

1.1 Основні поняття та структура Інтернет магазинів

1.1.1 Основні поняття Інтернет магазинів та веб-сайтів

Веб-сторінка – це документ, який може відображатися у веб-браузері, наприклад Firefox, Google Chrome, Opera, Microsoft Internet Explorer Apple Safari. Їх частіше називають просто "сторінками".

Веб-сайт – сукупність веб-сторінок, які згруповані разом і зазвичай з'єднані між собою різними способами. Часто його називають "веб-сайтом" або просто "сайтом".

Веб-сервер – комп'ютер, який розміщує веб-сайт в Інтернеті.

Пошукова система – веб-сервіс, який допомагає знаходити інші веб-сторінки, такі як Google, Bing, Yahoo або DuckDuckGo. До пошукових систем зазвичай доступний веб-браузер (наприклад, пошукова система безпосередньо в адресному рядку Firefox, Chrome тощо) або через веб-сторінку (наприклад, bing.com або duckduckgo.com).

Інтернет магазин – це веб-сайт, через який клієнти роблять замовлення. Він може представляти собою невеликий локальний магазин, великий торговий роздріб, магазин електронної комерції або особу, яка продає проекти через сторонній сайт. Інтернет магазин може працювати за низкою бізнес-моделей, включаючи бізнес-споживач, бізнес-бізнес або споживач-споживач.

Успішні Інтернет магазини та веб-сайти завжди орієнтовані на клієнтів у своїй основі. Якщо допомогти їм вирішити проблему або їхню потребу, вони можуть обрати саме ваш Інтернет магазин для здійснення своїх покупок

та з часом стати постійним клієнтом, що в свою чергу є безперечним успіхом. Кожен досвід роботи на веб-сайті повинен бути розроблений спеціально для вирішення задачі найбільш ефективним та орієнтованим на користувача шляхом.

Завжди слід ставити питання щоразу, коли є потреба додати чи оновити певні категорії, елементи, дизайн чи інтерфейс Інтернет магазину “Навіщо це?” “Чому потрібно додати це?” “Як корисно це буде для клієнтів?”. Варто бути уважним, адже при вирішенні однієї проблеми головне не створювати іншу в процесі. Варто також пам’ятати про принцип “простіше – краще”, при додаванні нових функцій тощо.

При створенні дизайну Інтернет магазину, виключна увага зосереджена на вмісті, структурі, потоці та функціональності без інших візуальних засобів, які б відволікали око.

Користувачі повинні мати можливість легко переходити по сайту незалежно від пристрою, на якому вони переглядають.

Все зводиться до того, щоб знайти своїх користувачів, включаючи розуміння їх цілей, навичок, уподобань та тенденцій. Коли дізнається інформація про свого користувача, слід враховувати наступне при розробці інтерфейсу:

– Зберігати інтерфейс простим. Кращі інтерфейси майже непомітні для користувача. Вони уникають зайвих елементів і чіткі в мові, якою вони користуються на етикетках та в обміну повідомленнями.

– Створювати послідовність та використання загальних елементів інтерфейсу користувача. Використовуючи загальні елементи у інтерфейсі, користувачі відчувають себе комфортніше та можуть швидше зробити покупку. Також важливо по можливості створювати підписані моделі, макети та дизайн на всьому сайті, щоб сприяти підвищенню ефективності.

– Цілісність у макетах сторінок. Ретельне розміщення предметів може допомогти привернути увагу до найважливіших відомостей та може

допомогти сканувати та читати. Стратегічно використовувати та спрямовувати увагу на елементи, використовуючи колір, світло, контраст та текстуру на свою користь.

– Використання типографіки для створення ієрархії та ясності. Ретельне обдумування, як використовувати шрифт і який саме. Різні розміри, шрифти та розташування тексту сприяють збільшенню масштабованості, розбірливості та читабельності.

– Слід переконатись, що система повідомляє, що відбувається. Завжди треба повідомляти своїх користувачів про місцезнаходження, дії, зміни стану чи помилки.

– Поміркованість за замовчуванням. Ретельно продумавши та передчуваючи цілі, які користувачі наносять на ваш сайт, є можливість створювати типові параметри, які зменшують навантаження на покупця. Це стає особливо важливим, коли мова йде про дизайн форми, де може бути можливість попередньо обрати або заповнити деякі поля.

Є багато причин, чому клієнти сьогодні віддають перевагу покупкам в Інтернеті:

1. *Зручність*. Зручність – найбільша вигода. Немає жодних ліній, на яких потрібно чекати або торгові помічники, які чекатимуть, щоб допомогти вам у покупках, і є можливість зробити покупки за лічені хвилини. Інтернет магазини дають змогу здійснювати покупки цілодобово. Також це краще місце для придбання таких інформаційних продуктів, як електронні книги, які доступні миттєво, як тільки пройде оплата.

2. *Порівняння цін*. Порівнювати та досліджувати продукти та їхні ціни набагато простіше в Інтернеті. Крім того, клієнт має можливість ділитися інформацією та оглядами з іншими покупцями, які мають досвід роботи з продуктом чи роздрібною торгівлею.

3. *Без натовпу.* Є й такі клієнти, які ненавидять натовп, коли ходять по магазинах. Особливо під час фестивалів чи спеціальних заходів,

скупчення людей може бути величезним головним болем. Крім того, це, як правило, хаотичніше, чим більше натовп, тим більше змушує нас поспішати. До того ж для людей, які мають автомобіль, паркування стає величезною проблемою. Усі ці проблеми можна уникнути, коли ви здійснюєте покупки онлайн.

4. *Подарунки.* Людина може надсилати подарунки простіше. Робити приємно рідним та друзям легко, незалежно від того, де вони перебувають. Тепер не потрібно шукати привід для того, щоб не надсилати подарунок у різні свята.

5. *Система електронної комерції*. Ця перевага відноситься вже до самого магазину, вона забезпечує дані в реальному часі та аналітику щодо продуктів та клієнтів. Є можливість бачити, як люди взаємодіють із сайтом, які продукти їх цікавлять, що вони залишили у кошику та скільки тривала середня покупка. Це цінні показники, які дозволяють вносити корективи для задоволення потреб клієнта.

Для роботи в Інтернет магазині знадобиться каталог товарів, кошик для покупок та інші товари.

Для створеня працюючого та ефективного Інтернет магазину в першу чергу необхідно мати:

Веб хостинг

Веб-сервер розміщує Інтернет магазин, а хостинг електронної комерції надає функції, необхідні для створення, роботи та керування веб-магазином. Функції веб-хостингу включають програмне забезпечення кошика для покупок, протокол SSL, підтримка бази даних, послуги з обробки платежів, функції безпеки та інші функції. Хостинг електронної комерції – це послуга, яку пропонують постачальники послуг веб-хостингу на додаток до веб-сервера, необхідного для розміщення сайту.

Доменне ім'я

Для налаштування своєї присутності в Інтернет магазині потрібне доменне ім’я. Власник бізнесу реєструє доменне ім’я і посилає його на Інтернет магазин. Ім’я домену – це ідентифікатор магазину в Інтернеті.

Виділена IP-адреса

Веб-сервер Інтернет магазину має IP-адресу, яка дозволяє користувачеві підключитися до сервера. У свою чергу, Інтернет магазин шифрує дані, які перетікають між браузером та веб-сервером, використовуючи протокол SSL для захисту даних клієнтів. Приватний сертифікат SSL забезпечує безпеку клієнтів на веб-сайті.

Каталог товарів

Каталог товарів – це віртуальна вибірка, яка надає клієнтам перелік доступних товарів та їх описи, їх класифікацію, а також функцію пошуку. Каталог складається із сторінок категорій та сторінок із переліком продуктів. Використовуючи каталог товарів, клієнт може замовити товари, здійснити оплату, отримати доступ до обслуговування клієнтів, надати зворотній зв'язок та виконувати інші функції.

Програмне забезпечення для кошика для покупок

Програмне забезпечення для кошика покупок або програмне забезпечення для електронної комерції працює на веб-сайті. Програмне забезпечення підтримує каталог Інтернет магазину та обробку замовлень. Це програмне забезпечення можна придбати у різних постачальників або найняти розробника, для створення кошика для покупок.

Інтерфейс продавця

Кошик взаємодіє з рахунком продавця у фінансовій установі, яка потребує обробки кредитної картки через Інтернет у режимі реального часу. Продавець отримує торговий рахунок, необхідний для платіжної системи, у банку. Платіжна система може інтегруватися з платіжною системою.

Інтернет-процесор оплати

Інтернет-процесор платежів дозволяє Інтернет магазину приймати платежі кредитною карткою. Шлюз платежів перевіряє дані кредитної картки та обробляє транзакцію. Після зняття суми платежу за плату на обробку шлюз закидає залишки на банківський рахунок Інтернет магазину.

Калькулятор витрат на доставку

Вартість доставки може бути розрахована після того, як клієнт помістить товар у кошик для покупок. Після або до завершення замовлення калькулятор визначає плату за доставку на основі критеріїв, введених замовником Інтернет магазину. Наприклад, витрати на доставку можуть бути розраховані на основі ваги, місця призначення, країни та інших критеріїв.

Розрахунок податку

Продаж в Інтернеті не завершений, поки податки не будуть обчислені. Менеджер сайту Інтернет магазину періодично оновлює ставки податку. Можна також придбати програмне забезпечення, яке автоматично оновлює ставки податку.

1.1.2 Структура Інтернет магазинів

Найважливіші веб-сторінки Інтернет магазину, які перші приходять в голову кожному – це домашня сторінка, категорія та сторінка товарів, кошик і сторінка оформлення замовлення. Ці веб-сторінки є обов’язковими, наведені на рисунку 1.1. Та важливо одразу розуміти при створенні, що погана навігація виступає стримуючим фактором для клієнтів.

Здається, так просто створити Інтернет магазин; просто поєднати домен, хороший хостинг, систему управління вмістом і тему. Низькі накладні витрати створюють дуже маленький бар’єр для входу – не потрібно платити високу оренду, якщо у вас немає потреби у фізичному магазині.

Але насправді створити хороший Інтернет магазин досить важко, а зробити його успішним та зручним для користувачів ще складніше.

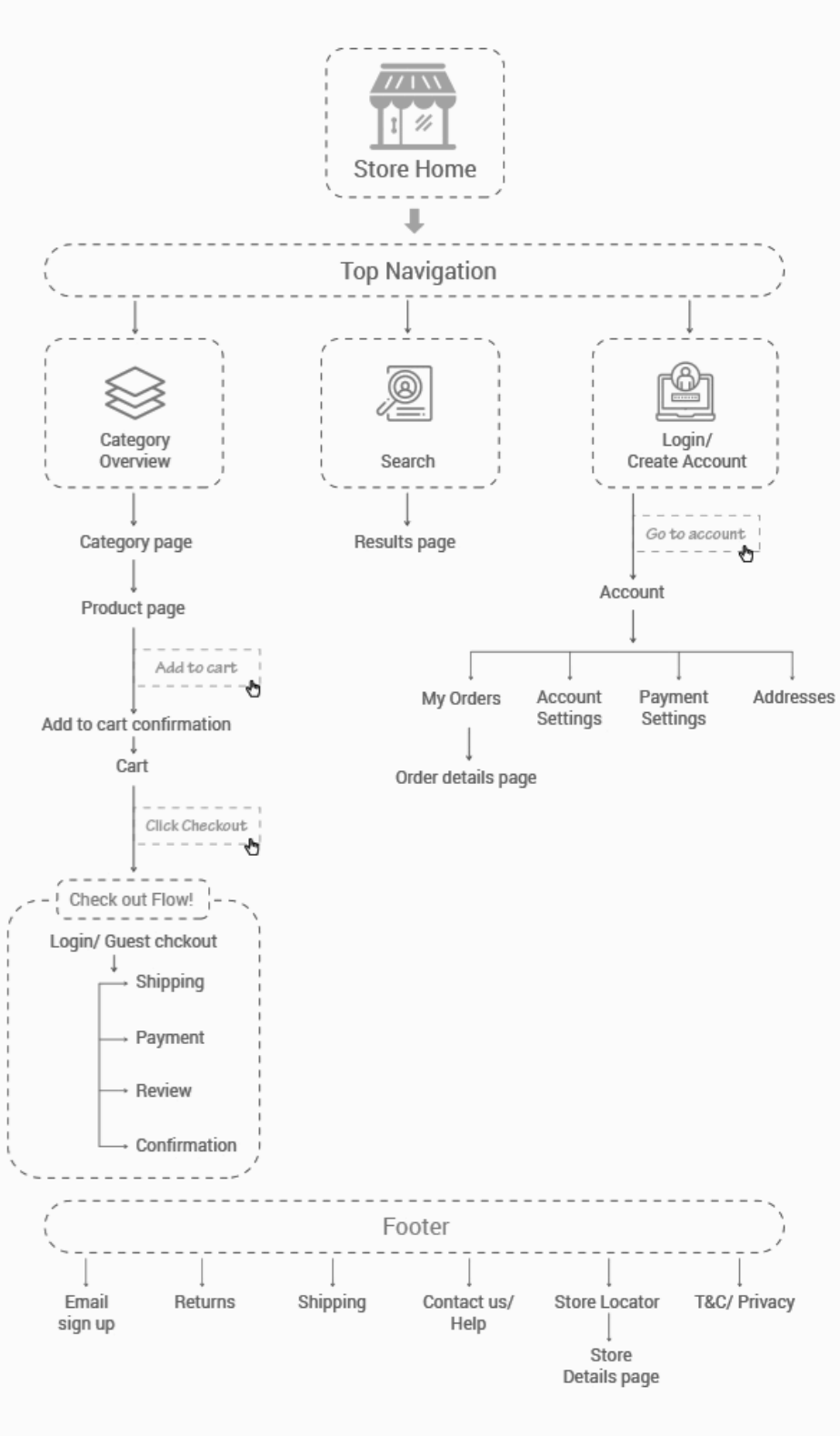


Рисунок 1.1 – Структура Інтернет магазину

Домашня сторінка

Домашня сторінка – це обличчя компанії, Інтернет магазину, тому її слід робити легко читабельною і зрозумілою. Клієнти цінують свій дорогоцінний час, тому вони не витратять кілька хвилин на те, щоб “розшифровувати”, що магазин продає або які послуги надає. Потрібна лаконічність та прямолінійність, передаючи найважливішу інформацію своїм клієнтам. Не варто повідомляти одразу ж про те, де знаходиться бізнес чи скільки в компанії менеджерів з продажу, цю інформацію можна розмістити на сторінці “Про нас”. Звичайно, є деякі винятки, але мається на увазі про загальні ситуації.

Хороша домашня сторінка також посилається на категорії, щоб полегшити навігацію по сайту та забезпечити безперебійну роботу користувачів. Також можуть бути банери зі спеціальними пропозиціями та акціями.

Іноді також потрібно встановити контакти або навіть онлайн-чат, щоб швидко відповісти на всі запитання користувачів. Чати в реальному часі мають найвищий рівень задоволеності для будь-якого каналу обслуговування клієнтів – 73%, тому варто подумати про використання його у магазині.

Сторінки категорій

Слід подумати про свій Інтернет магазин як офлайн-магазин на мить; сторінки категорій схожі на менші відділи в ній, тому клієнтам треба показати, що у Інтернет магазині на полицях: розмістити зображення, додати до кошика і, можливо, ще якусь корисну інформацію (короткий опис, рейтинг тощо). Також можна додати параметри сортування (за бестселерами, підрахунками відгуків, ціною, новинками тощо) або прокрутку Ajax. Якщо вирішили піти на сортування, варто переконатись, що правильно встановилені всі канонічні та навігаційні теги, щоб не обтяжувати Google дублюючим вмістом. Якщо обрана нескінченна прокрутка для своїх категорій, треба дотримуватися інструкцій Google.

Деякі магазини використовують швидкий перегляд, який також чудово працює. Однак вони зменшують кількість переглянутих сторінок за сеанс, оскільки клієнту не потрібно відкривати кожен продукт на окремій вкладці.

Все залежить від асортименту продукції та клієнтів, яких обслуговують.

Сторінка продукту

Клієнт переходить через сайт, щоб перейти на сторінку продукту. Його намір – знайти щось цікаве і, можливо, зробити покупку. Остаточне рішення значною мірою залежить від якості та кількості наданої інформації. Але це не означає, що повинно буде написано опис у 1000 слів (можна зберегти слова для керівництва користувача або навіть повідомлення в блозі про продукт); Головна мета – показати всю релевантну інформацію якомога швидше, тому вона повинна бути організована в зручному для користувача порядку.

Ось обов'язкові компоненти сторінки продукту:

– *Зображення*

Використовути треба високоякісні зображення із збільшенням та підбором кольорів, остання актуально для продуктів, що настроюються. Це слід робити, щоб забезпечити хороший досвід користувачеві.

– *Опис*

Можна надати короткий опис для швидкого сканування та більш довгий опис з технічними деталями. Слід вирішувати на власний розсуд, які параметри форматування та відображення використовувати, але це має виглядати природно на сторінці. Більше того, доведеться наполегливо писати описи продуктів, вони повинні бути унікальними.

– *Ціна*

Більшість Інтернет магазинів показують ціни, і не дивно, люди вважають за краще знати, чого чекати. Особливо це стосується грошей, ніхто не любить ризикувати необгрунтованими своїми тяжко заробленими активами.

Якщо це продукт зі знижкою, можна прорекламувати цю інформацію, порівняти ціну продажу з діючою.

– *Додати в кошик*

Насправді, головна мета – не показувати свій дизайн, фотографії чи щось подібне. Головна мета – продати.

Крім того, можна відобразити параметри "Додати для порівняння" та "Додати до списку бажань".

– *Перехресні продажі*

Чудово, якщо клієнт купує товар, але він міг би придбати більше! Тож мета тут – запропонувати більше продуктів, які відповідають смаку вашого клієнта. Можна показати пов’язані товари та/або “Клієнти, які придбали цей товар, також купили”. Ця методика може збільшити середню вартість замовлення у Інтернет магазині.

Спеціальні пропозиції та рекламні пропозиції

Уявімо, що вже є веб-сайт з ідеальними сторінками шанувань, категорій та продуктів. Тепер потрібно залучати нових клієнтів і змушувати старих клієнтів повертатися. Слід розробити маркетинговий план і запустити акції.

– *Цільові сторінки*

Якщо запускається нова колекція чи спеціальна категорія у Інтернет магазині, можна створити акцію та зробити це голосно! Для цього може знадобитися гарна цільова сторінка;

Дизайн та повідомлення, яке передає цільову сторінку, значною мірою залежить від бізнесу та цілей, які потрібно досягти.

– *Купонні коди*

Можна давати клієнтам знижки в різні події та свята, які можуть бути глобальними (наприклад Різдво), місцевими (наприклад День Перемоги) або навіть особистими (дні народження, N замовлення тощо).

– *Ігри*

Гейміфікація відіграє все більш важливу роль в електронній комерції, і треба навчитися її використовувати. Можна створювати різні ігри, які допомагають клієнтам заробляти бали або купони. Людям подобається розважатися, просто потрібно знайти правильний підхід.

Відстеження та аналіз

Коли вкладається багато часу та зусиль у маркетинг та на сайт загалом, хочететься знати, як вони впливають на користувачів і бачити, як клієнти подорожують сайтом. Найпростіший спосіб отримати цю інформацію – це налаштувати свій код Google Analytics. Після того, як це зроблено, ось ще кілька важливих речей:

Слід створити 3 перегляди: 1-й із необробленими даними; 2-й з різними фільтрами (наприклад, обмеження IP-адреси); 3-й для тестування нових речей (фільтри, групування тощо).

Постановка цілей: додавання в кошик, покупка, підписка на розсилку та ін.

Налаштування відстеження електронної комерції.

Звіти Google Analytics за замовчуванням охоплюють майже все, що потрібно. Але якщо є якісь особливі питання, можна створити власні панелі інструментів, звіти.

Коли сайт показує звернення до відвідувачів, вони швидше купують товар. Або коли показує відносний вміст і тд, що шукають кінцеві користувачі, збільшується ймовірність високого рейтингу Інтернет магазину. Суть полягає в тому, що потрібно швидко та легко показувати інформацію відвідувачам та пошуковим системам.

Кращі практики структуризації Інтернет магазину

Щоб допомогти пошуковим системам швидше та простіше сканувати, потрібна структура веб-сайту, яка спирається на вміст, написаний контекстуально та пов'язаний між собою. Коли сканер читає веб-сайт, ця

структура допомагає боту будувати контекстні внутрішні посилання. Ідеальна структура Інтернет магазину для кращого рейтингу SEO представлена на рисунку 1.2.

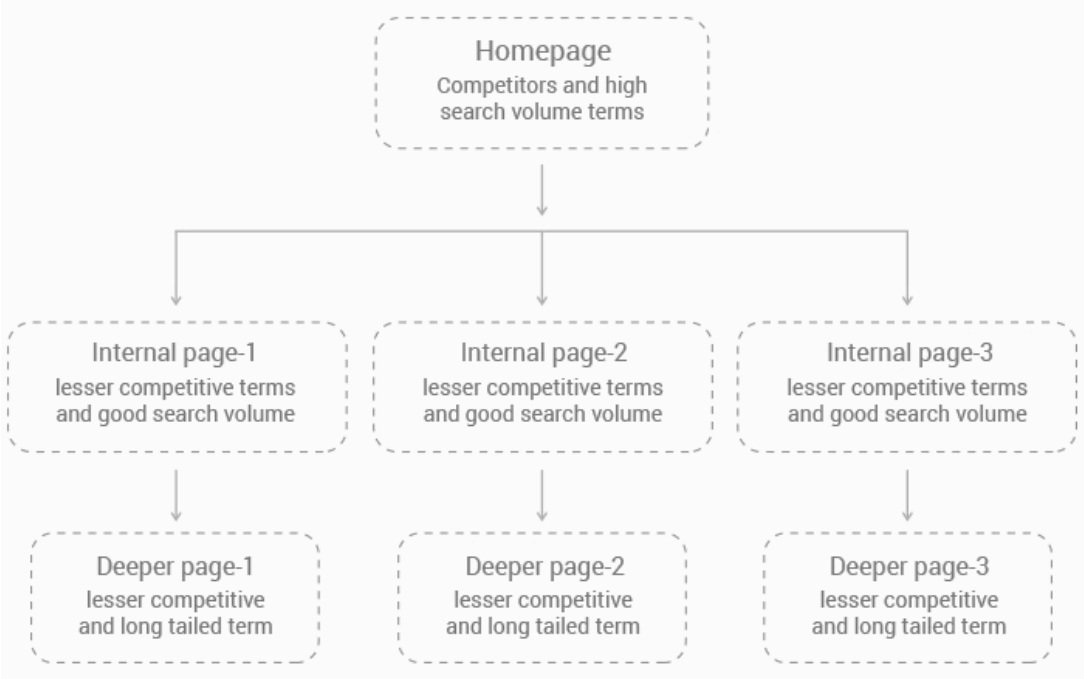


Рисунок 1.2 – Ідеальна структура інтернет магазину

Деякі з найкращих практик такі:

– більш конкурентоспроможні умови на головних сторінках;

– довгохвості терміни на більш глибоких сторінках;

– структура верхнього рівня, яка допомагає підсилити ключові слова на веб-сайті, щоб пошукові системи знали глибше, ніж домашня сторінка;

– пов’язати авторитетні сторінки з тими сторінками, які потрібно ранжирувати;

– пов’язати мапи сайту домашньої сторінки на внутрішні сторінки (категорія та підкатегорія);

– сегментайія HTML-карт сайтів з їх систематикою для індексації сторінок (рисунок 1.3).

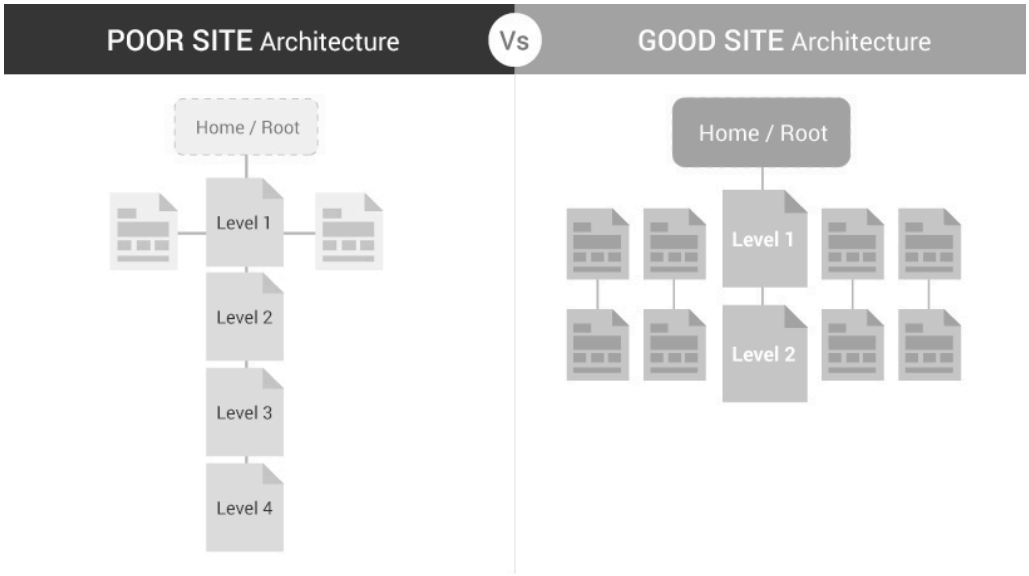


Рисунок 1.3 – Порівняння архітектури сайтів

Щоб клієнти легше знаходили інформацію, потрібна чітка навігація.

Використати можна структуру категорії вищого рівня, яка, ймовірно, буде відокремлена за визначеним їх видом. Це дозволяє клієнтам легко знати, який шлях їм потрібно пройти. Він уточнює, яка інформація надається на веб-сайті.

Потрібно направляти далі, щоб глибоко орієнтуватися в конкретній інформації. Будь-то товар чи послуга, нехай вони детально вписуються в деталі. Можна меншити кількість зображень для відображення на домашній сторінці. Нехай клієнт переходить на певну сторінку продукту, а потім йому надається опис, характеристики тощо.

Інтернет магазин вимагає вразити своїх клієнтів дизайном та структурно організованою інформацією для їх навігації. Як надається правильний спосіб шукати інформацію, знайдеться правильний шлях для досягнення постійних клієнтів.

1.1.3 Класифікація Інтернет магазинів

Класифікують Інтернет магазини по різних критеріях: по моделі бізнесу, по відношенню з постачальниками, по товарному асортименту, по методах роздрібного продажу товарів в мережі.

Інтернет магазини, частіше всього, класифікують за наступними ознаками:

- за методом роздрібного продажу товарів у мережі;
- за бізнес-моделлю, за взаємовідносинами з постачальниками;
- за ступенем автоматизації.

За методом роздрібного продажу товарів у мережі розрізняють Інтернет магазини; веб-вітрини, торгові системи; торгові ряди; контентні проекти (споживацькі енциклопедії, системи Інтернет-замовлень товарів тощо).

За ступенем автоматизації серед торгових систем електронних магазинів розрізняють веб-вітрини та власне Інтернет магазини.

Характерною рисою Інтернет магазину є повна автоматизація системи обробки замовлень, завдяки чому можна працювати індивідуально з кожним зареєстрованим покупцем.

2 ТЕХНОЛОГІЇ РОЗРОБКИ ІНТЕРНЕТ МАГАЗИНУ

2.1 Вибір мови програмування

JavaScript – це динамічна мова програмування. Він легкий і найчастіше використовується у складі веб-сторінок, реалізація яких дозволяє сценарію на стороні клієнта взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об’єктно-орієнтованими можливостями.

Спочатку JavaScript був відомий як LiveScript, але Netscape змінив свою назву на JavaScript. Вперше JavaScript з'явився в Netscape 2.0 в 1995 році під назвою LiveScript. Ядро мови загального призначення вбудовано в Netscape, Internet Explorer та інші веб-браузери.

Специфікація ECMA-262 визначала стандартну версію основної мови JavaScript.

- JavaScript – це легка, інтерпретована мова програмування;
- призначена для створення програм, орієнтованих на мережу;
- доповнює і інтегрується з Java;
- доповнює та інтегрується з HTML;
- відкрита та кросплатформна.

Клієнтський JavaScript

JavaScript на стороні клієнта – це найпоширеніша форма мови. Сценарій повинен бути включений у HTML-документ або посилатися на нього, щоб код інтерпретувався браузером.

Це означає, що веб-сторінка не повинна бути статичною мовою HTML, але вона може включати програми, які взаємодіють з користувачем, контролюють браузер та динамічно створюють вміст HTML.

Механізм клієнтського боку JavaScript надає багато переваг перед традиційними скриптами на сервері CGI. Наприклад, можна використовувати JavaScript, щоб перевірити, чи користувач ввів дійсну електронну адресу в поле форми.

JavaScript код виконується, коли користувач подає форму, і лише якщо всі записи дійсні, вони будуть передані на веб-сервер.

JavaScript може використовуватися для лову подій, ініційованих користувачем, таких як натискання кнопок, навігація посилання та інші дії, які користувач ініціює явно або неявно.

Переваги використання JavaScript:

1. Менше взаємодії з сервером.

Ви можете перевірити введення даних користувача перед тим, як відправити сторінку на сервер. Це економить трафік сервера, що означає менше навантаження на ваш сервер.

2. Негайний відгук відвідувачів.

Їм не потрібно чекати перезавантаження сторінки, щоб побачити, чи забули щось ввести.

3. Підвищена інтерактивність.

Можна створювати інтерфейси, які реагують, коли користувач наводить на них мишку або активує їх за допомогою клавіатури.

4. Багатші інтерфейси.

Можна використовувати JavaScript для включення таких елементів, як компоненти перетягування та повзунки, щоб надати відвідувачам свого сайту багатий інтерфейс.

Також JavaScript використовується в AJAX, популярному підході до побудови призначених для користувача інтерфейсів веб-додатків, що полягає в «фоновому» асинхронному обміні даними браузера з веб-сервера. При оновленні даних веб-сторінка не перезавантажується повністю і інтерфейс

веб-додатка стає швидше, ніж це відбувається при традиційному підході (без застосування AJAX).

2.2 Технології програмування

У даному Інтернет магазині будуть використовуватись такі технології: Node.js, Express.js, MongoDB, JQuery, Bootstrap, Pug.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих стосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше JavaScript застосовувався для обробки даних в браузері користувача, то Node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

До найважливіших переваг Node можна віднести:

- дозволяє створювати додатки в режимі реального часу (наприклад, чати чи ігри) блискавично;
- робить можливим кодування в JavaScript як для клієнта, так і для сервера;
- підвищує ефективність процесу розробки, оскільки він заповнює розрив між розробниками інтерфейсу та бекенда;
- постійно зростаючий NPM (Node Package Manager) надає розробникам безліч інструментів і модулів для використання, тим самим додатково підвищуючи їх продуктивність;
- код виконується швидше, ніж на будь-якій іншій мові;
- Node ідеально підходить для мікропослуг, які є популярним рішенням серед корпоративних програм.

Express.js, або просто *Express* – програмний каркас розробки серверної частини веб-застосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-застосунків і API. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний, але має велику кількість плагінів, що підключаються.

MongoDB – об'єкнто-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Переваги використання MongoDB:

1. MongoDB забезпечує просту установку.

За допомогою хостингу A2 можна використовувати інсталяцію MongoDB одним натисканням через Webuzo. Це безкоштовна програма, сумісна з багатьма шаблонами Операційної Системи (ОС). Якщо програміст не клієнт A2 хостингу, знадобиться скористатися майстром установки, який постачається в комплекті з MongoDB. Він доступний для macOS, Linux, Windows та Docker.

2. База даних без схем.

База даних MongoDB не має схеми. Схема – це структура таблиць і стовпців, що використовуються в багатьох базах даних.

Використовуючи щось на зразок SQL, потрібно визначити схему, перш ніж навіть можна буде налаштувати свою базу даних. На щастя, MongoDB усуває це, дозволяючи розширити універсальність для зберігання різних типів файлів та швидкого доступу до них.

На відміну від баз даних, які покладаються на SQL, MongoDB – це об’єктно-орієнтована СУБД. Коротше кажучи, він використовує Документи, організовані у Колекції, а не таблиці.

Завдяки цій системі MongoDB забезпечує чітке збільшення швидкості для своїх користувачів. Завдяки базі даних без схем, все, що потрібно зробити, – це індексувати точку даних для її отримання. Одне тільки це може зняти велику частину напруги обладнання, оскільки воно вимагає меншої потужності для обробки.

3. Пропонує високу продуктивність.

Продуктивність бази даних може вплинути на швидкість веб-сайту чи програми. Це може або покращити досвід користувачів (UX), або викликати у користувачів головний біль, поки вони чекають завантаження вмісту.

Зокрема, це не дозволяє приєднуватися до таблиці, як це робить SQL. За допомогою SQL можна приєднати внутрішній, лівий (зовнішній), правий (зовнішній) та повний (зовнішній). Ці процеси зберуть дві таблиці частково або повністю.

Знову ж таки, можливість приєднання до потрібних таблиць є важливою особливістю для реляційних баз даних для об’єднання рядків та обміну інформацією. І все-таки без тієї самої структури в MongoDB немає таблиць, до яких можна приєднатися.

Хоча це може здатися недоліком, виконання з’єднань може бути інтенсивним процесом, який може негативно вплинути на продуктивність. Покладаючись на Документи, заощаджується простір та ресурси.

4. Економічно ефективні СУБД.

Якщо переглянути параметри SQL, найпростіша версія для розробників – безкоштовна. Однак якщо треба чогось більш всебічного і придбається сервер SQL через, Microsoft, ціни можуть швидко збільшитися в тисячі. Насправді найдорожчий план коштує 14 256 доларів.

Що стосується економічності, MongoDB часто перевершує. Це не тільки безкоштовна СУБД з відкритим кодом, але й пропонує підтримку декількох мов.

Десять з цих мов є офіційними, але є багато інших, які були створені великою спільнотою MongoDB. Можна помітити із програмним забезпеченням з відкритим кодом, навколишня спільнота активна та охоче ділиться з іншими користувачами, що є безперечною перевагою.

Крім того, MongoDB має можливість легко поширюватися на декілька серверів, що може дати можливість рости. Також він має повну підтримку індексу та динамічні запити, що дозволяють знаходити дані за будь-якими критеріями. Якщо потрібно використовувати велику кількість даних, щоб забезпечити ефективність роботи веб-сайту чи програми, то це одне з найкращих доступних рішень.

jQuery – популярна JavaScript-бібліотека з відкритим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресігом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Основна перевага використання Bootstrap як системи CSS полягає у заохоченні послідовності використання внутрішніх інструментів. Bootstrap підтримує найновіші версії Google Chrome, Firefox, Internet Explorer, Opera та Safari. Він навіть надає підтримку ще в Internet Explorer 8, тобто Bootstrap пропонує ширшу доступність з самого початку.

2.3 Середовище програмування

Visual Studio Code – це легкий, але потужний редактор вихідного коду, який працює на десктопі та доступний для Windows, macOS та Linux. Він оснащений вбудованою підтримкою JavaScript, TypeScript і Node.js і має багату систему розширень для інших мов (таких як C ++, C #, Java, Python, PHP, Go) та режимів виконання (таких як .NET і Unity).

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підкаски. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як мало енергозатратне рішення, що дозволяє обійтися без повного інтегрованого середовища розробки.

За основу для Visual Studio Code використовуються напрацювання вільного проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проекту Atom на сайті Visual Studio Code і в прес-релізі і в офіційному блозі не згадується.

2.4 Розробка структури та алгоритмів

Даний приклад Інтернет магазину побудований на архітектурі клієнт-сервер. Клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Така архітектура визначає такі типи компонентів:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Роль – це функція сервера (наприклад поштовий, контролер домена тощо). Один сервер може відігравати як одну так і декілька ролей одночасно.

Веб-сервер – сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дволанкова клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта.

Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп’ютери, мобільні телефони та ін.

Блок-схема роботи Інтернет магазину наведена на рисунку 2.1.

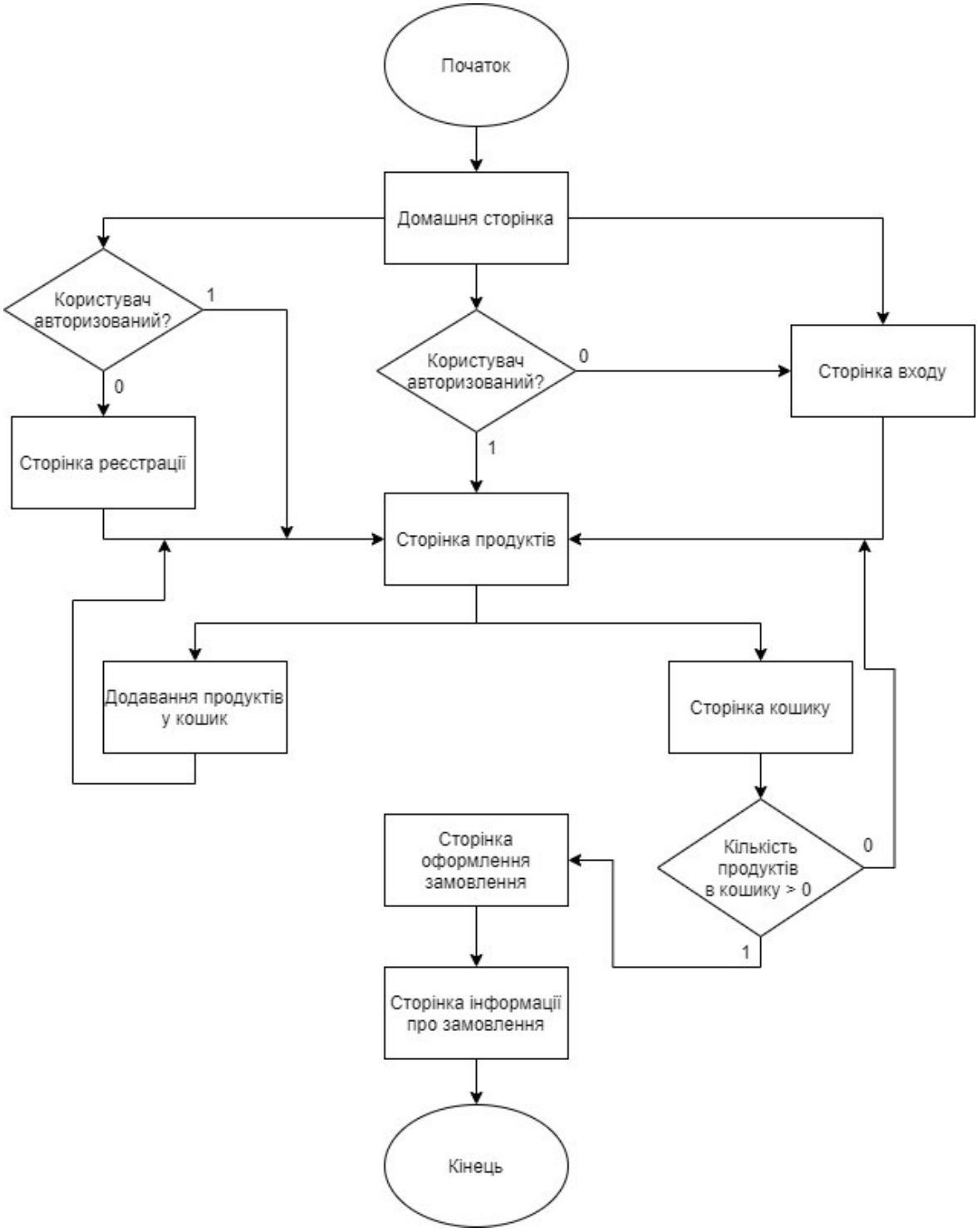


Рисунок 2.1 – Блок-схема роботи Інтернет магазину

Схема алгоритму додавання товару у кошик наведена на рисунку 2.2.

Рисунок 2.2 – Схема алгоритму додавання товару у кошик

Схема алгоритму оформлення замовлення наведена на рисунку 2.3.

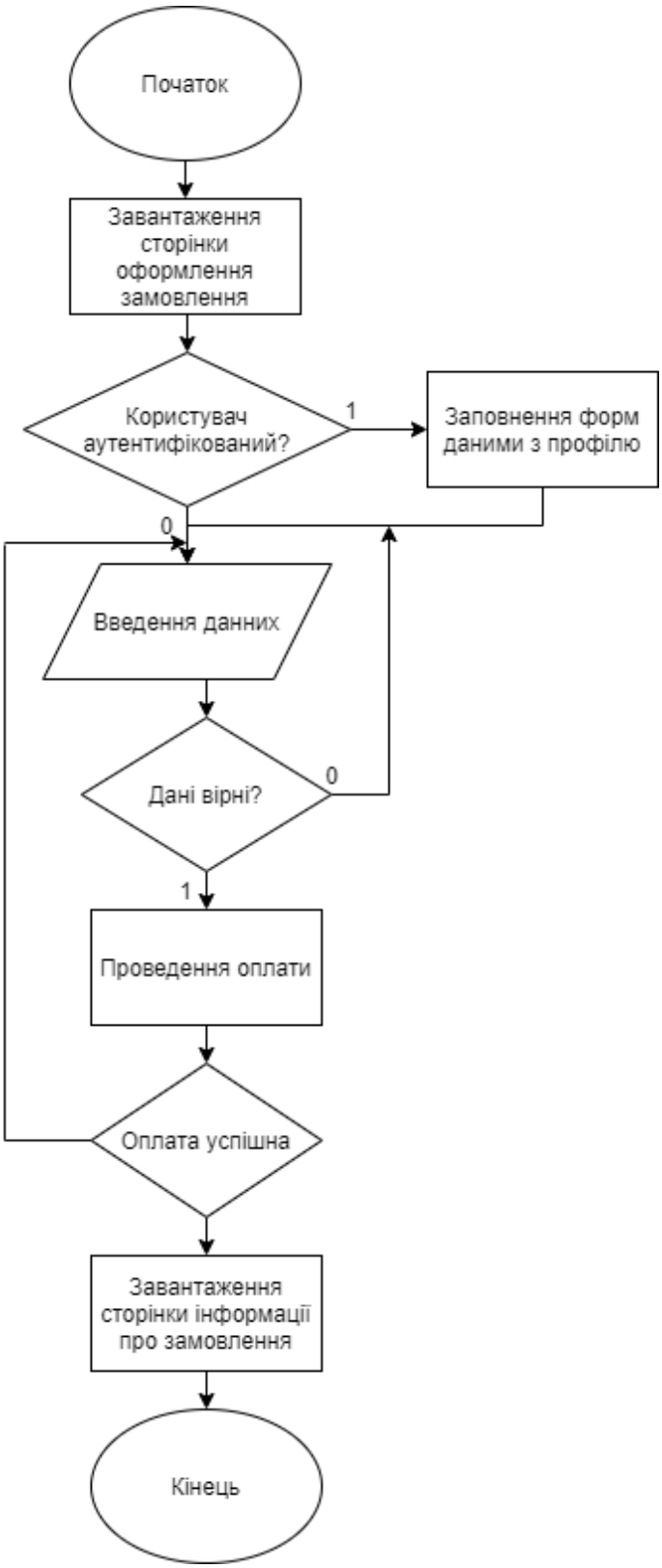


Рисунок 2.3 – Схема алгоритму оформлення замовлення

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ПРИКЛАДІ ОНЛАЙН МАГАЗИНУ “GREEN LOTUS”

3.1 Створення програмного модуля Інтернет магазину

Створення програмного модуля можна розділити на 3 частини:

- клієнтська частина;
- серверна частина;
- база даних;

Клієнтська частина охоплює формування розмітки у шаблонізаторі Pug, додавання стилів за допомогою Bootstrap.

Даний програмний модуль забезпечує коректну роботу з відображенням елементів на сторінці та контенту, також обробку запитів юзера, виконання реєстрації або перевірку даних при вході в особистий кабінет.

3.2 Структура бази даних

У даній роботі для того, щоб Інтернет магазин ефективно працював, зберігав дані про товари, які були додані до кошика та безпосередньо інформацію про користувача, який бажає зробити покупку, потрібно створити три таблички, з яких і формується структура бази даних.

Таблиця Products, у якій зберігаються дані про товари, які є в наявності у Інтернет магазині.

Таблиця Shopping Cart, у якій зберігаються ті товари, які були додані користувачем до свого кошика.

Таблиця Users, у якій зберігаються дані зареєстрованих користувачів.

Ці таблиці пов’язані між собою так, що певним таблицям Users надані свої таблиці Shopping Carts, у яких зберігаються додані покупки, в свою чергу таблиці Products і Shopping Cart пов’язані між собою так, що таблиця Shopping Cart може зберігати велику кількість значень у таблиці Products. Схема наведена у додатку В.

3.3 Збірка програми та приклади розробленого Інтернет магазину

Під час розробки даної роботи, серверна частина програми цілком відповідає на запити користувача, що працює із сайтом Інтернет магазину.

Створений Інтернет магазин “Green Lotus” відповідає поставленим цілям і вимогам. Інтерфейс побудований з урахуванням потреб кінцевих користувачів.

Коли клієнт потрапляє до нашого Інтернет магазину, перш за все він опиняється на головній сторінці (рисунок 3.1).

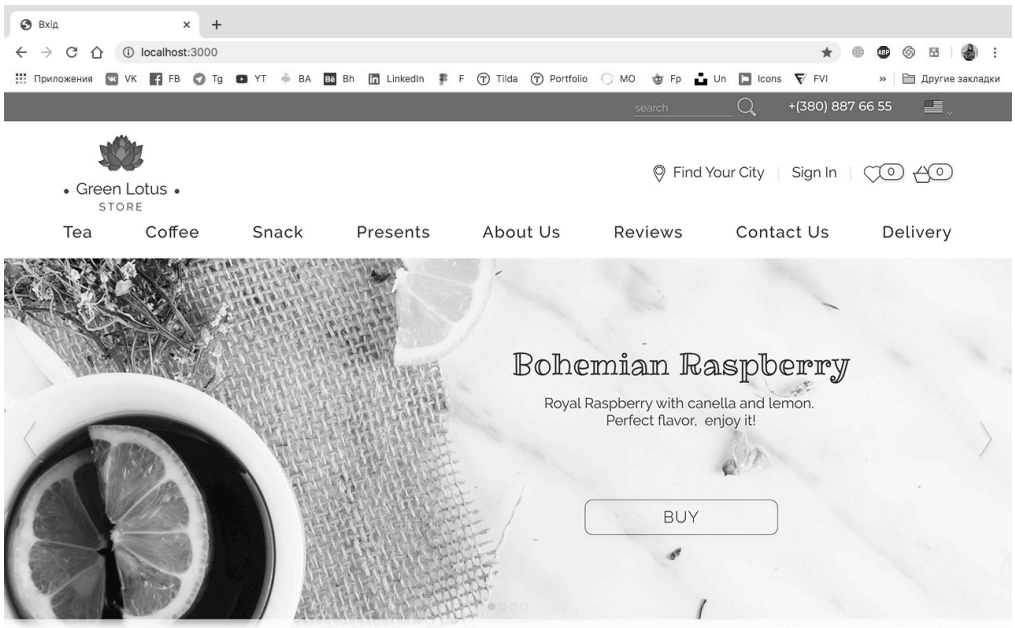


Рисунок 3.1 – Головна сторінка Інтернет магазину

Далі користувач може вибрати категорію з товарами, які його цікавлять та перейти безпосередньо до списку цих товарів (рисунок 3.2).

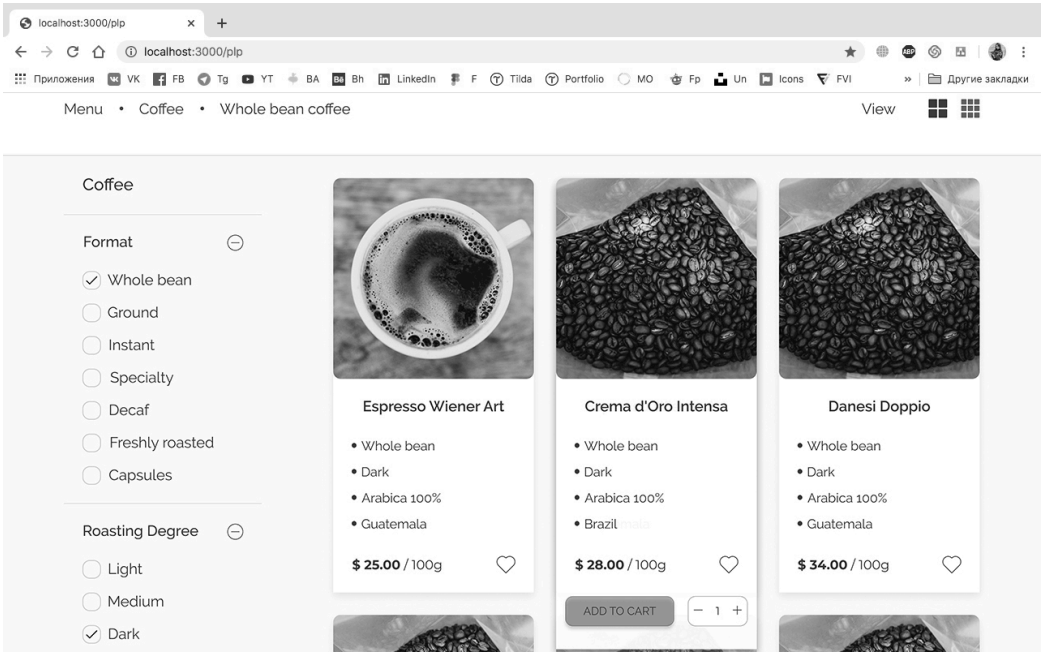


Рисунок 3.2 – Список товарів

Користувач також може перейти до самої картки товару (рисунок 3.3).

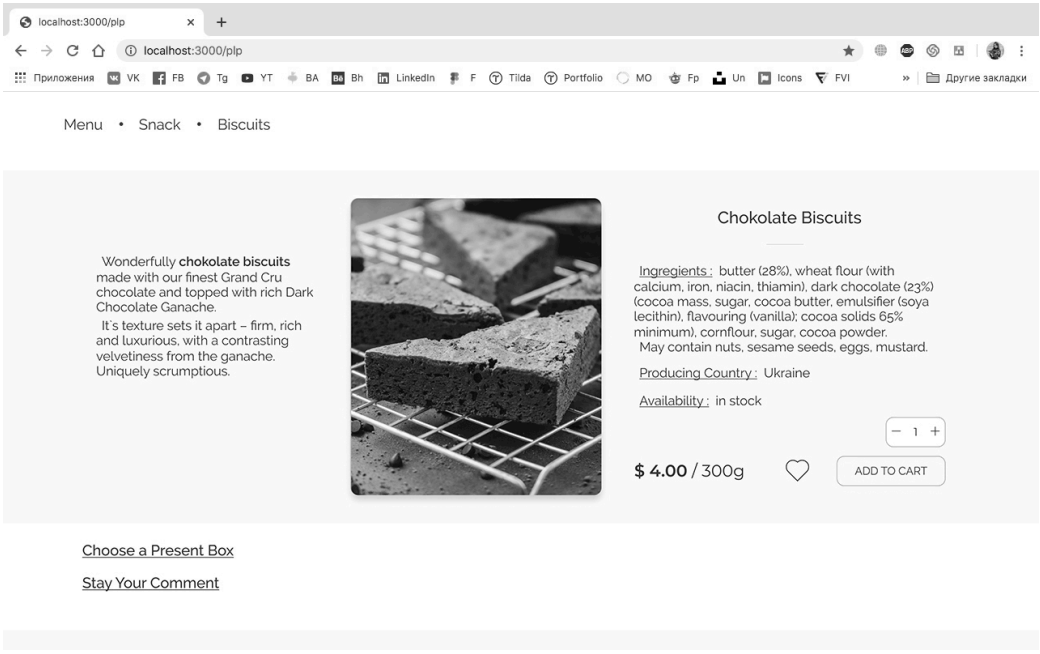


Рисунок 3.3 – Картка товару

Щоб користувач міг додати певний товар до кошика, йому потрібно зареєструватись (рисунок 3.4).

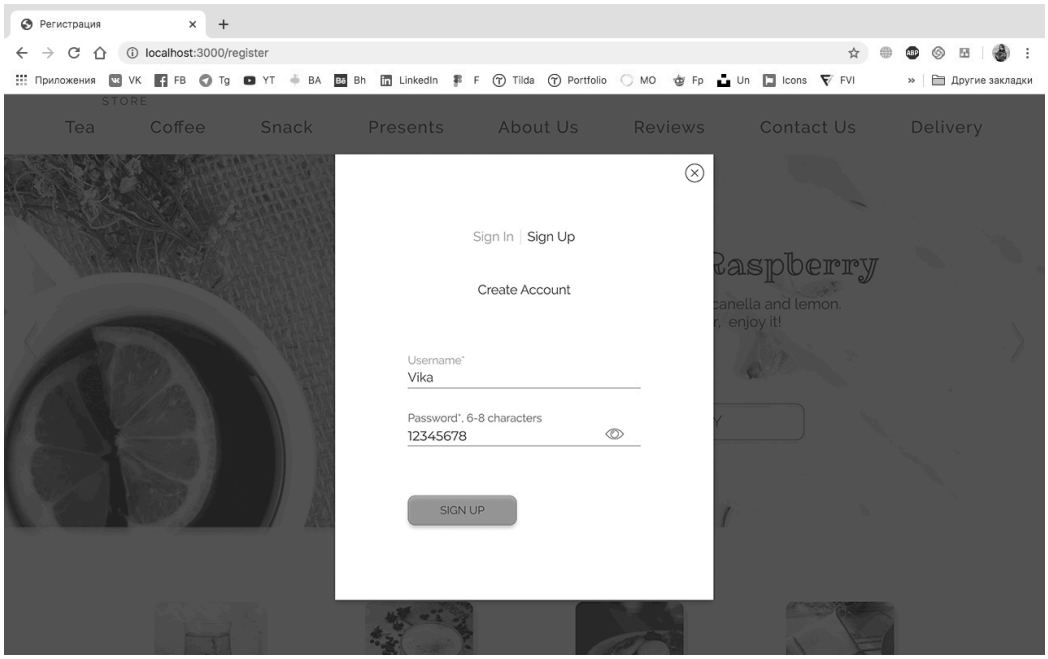


Рисунок 3.4 – Реєстрація користувача

Запис нового користувача було додано до бази даних (рисунок 3.5).

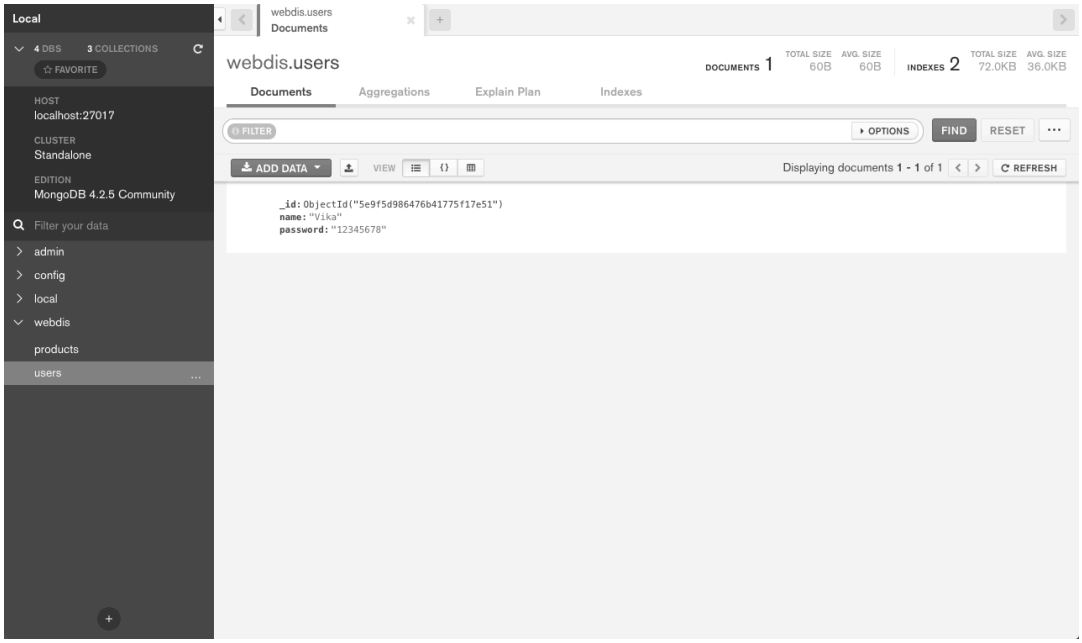


Рисунок 3.5 – Колекція користувачів

Якщо ж користувач вже був зареєстрований, йому потрібно авторизуватись для того, щоб далі мати змогу купувати (рисунок 3.6).

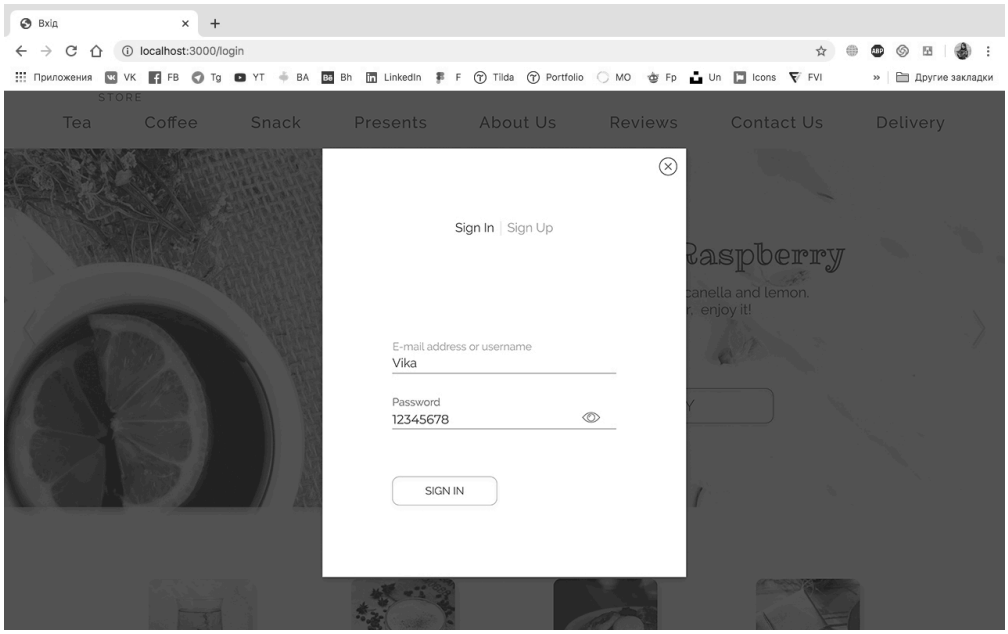


Рисунок 3.6 – Авторизація користувача

Після чого клієнт може також зайти до персонального кабінету (рисунок 3.7).

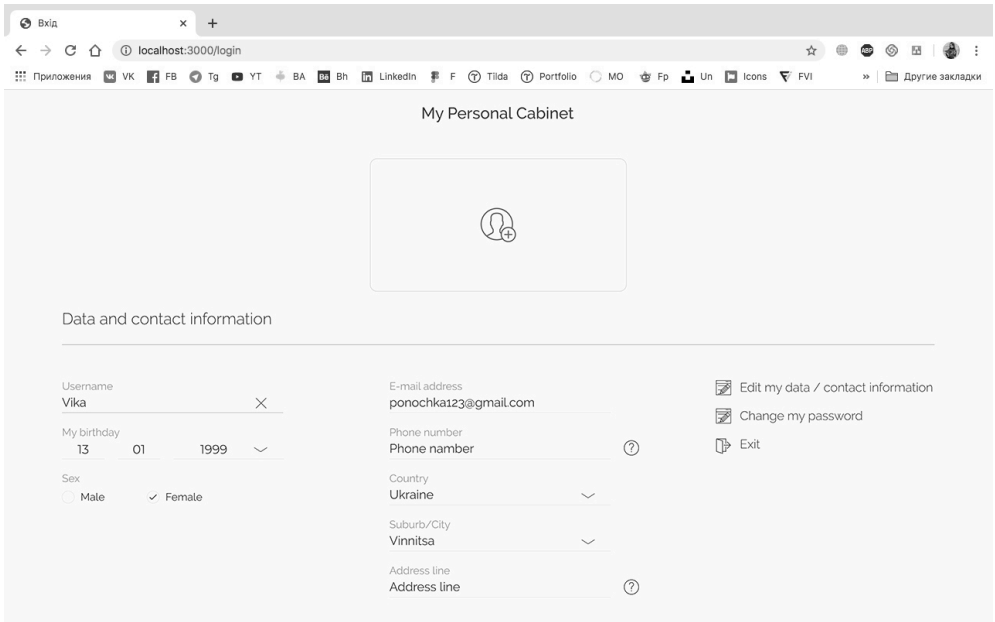


Рисунок 3.7 – Персональний кабінет користувача

Після авторизації користувач може вільно додавати потрібні йому товари до кошика (рисунок 3.8).

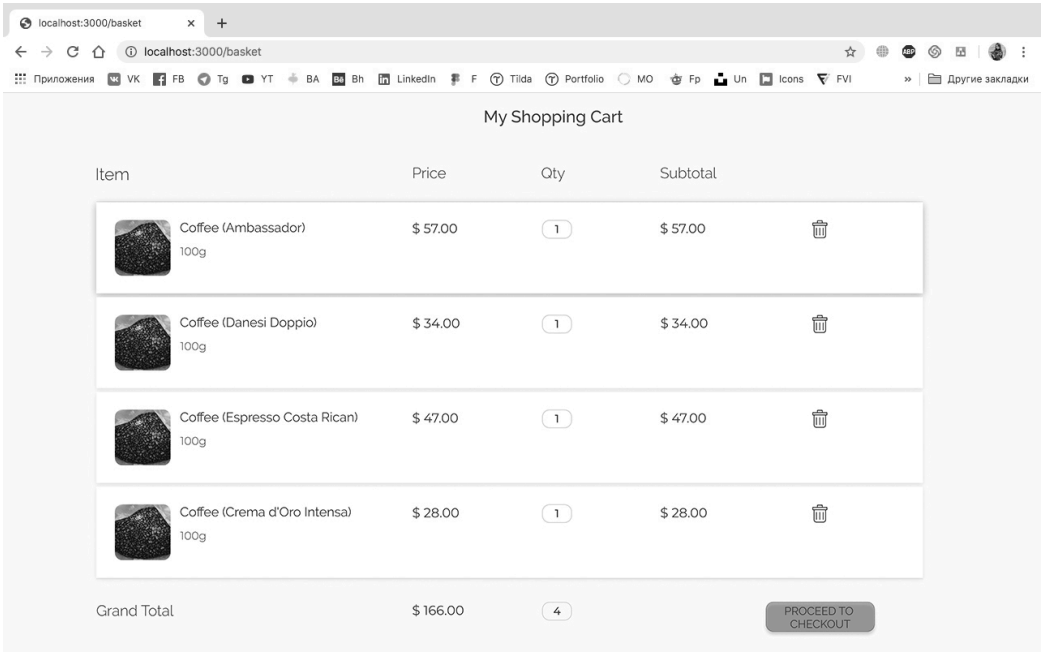


Рисунок 3.8 – Кошик користувача

Тоді ж база даних наповнена декількома тестовими продуктами (рисунок 3.9).

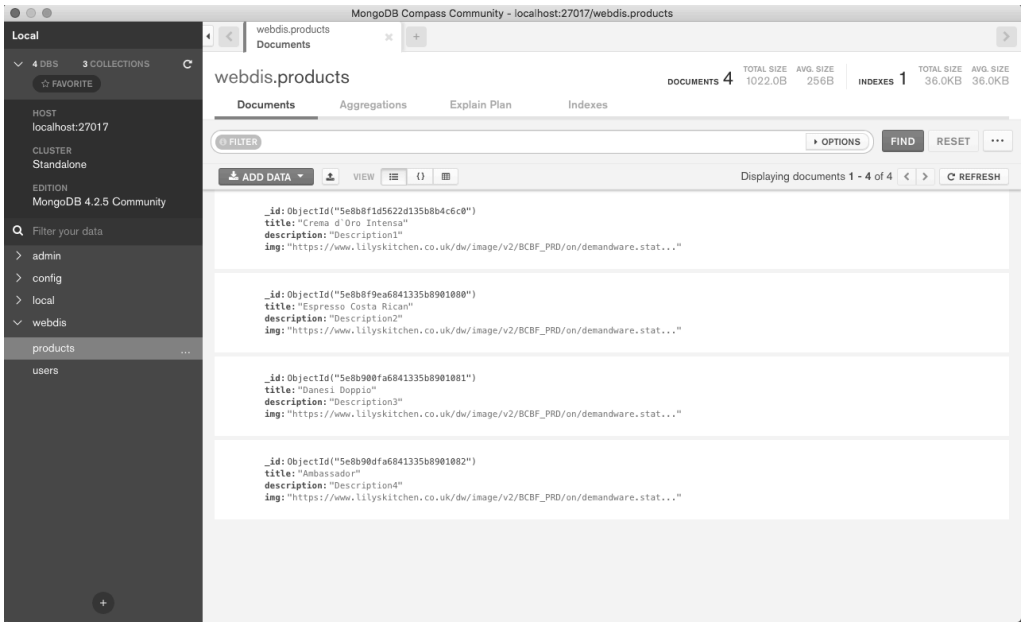


Рисунок 3.9 – База даних наповнена декількома товарами

Далі користувач може спокійно офурмлювати та проплачувати свої замовлення (рисунок 3.10).

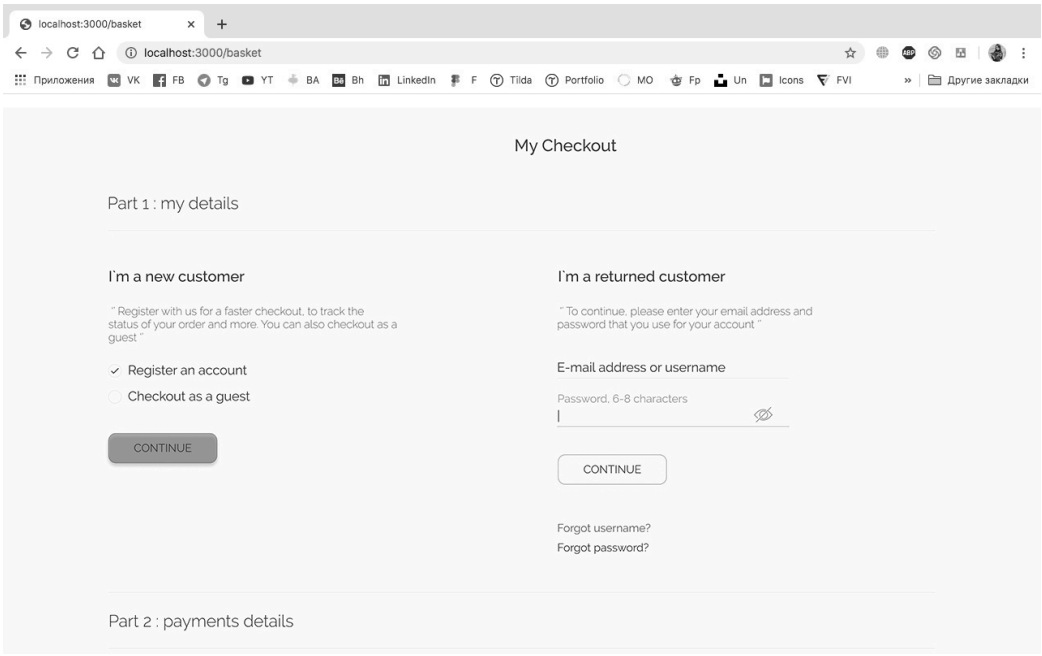


Рисунок 3.10 – Оформлення замовлення

3.4 Тестування Інтернет магазину

Після проведення тестуванн було перевірено роботу Інтернет магазину та всіх його компонентів, швидкість обробки даних, навігацію, простоту у пошуку потрібної позиції, працездатність, сумісніть з браузером, єдність та дружність дизайну, функціональність.

При розробці головним чином враховувалось :

- працездатність Інтернет магазину;
- легкість навігації та пошуку;
- коректне додавання товару до кошика;
- інтуїтивно зрозумілий інтерфейс сайту;
- швидкість завантаження сторінок.

Загалом тестування дало позитивний результат. Звичайно є маленькі ньюанси, які будуть в подальшому доопрацювані, але на даний момент, сайт є цілком задовільним та швидким з боку клієнта та сервера.

ВИСНОВКИ

У даній роботі було проаналізовано актуальність теми, особливо в момент карантину потреба в онлайн шопінгу значно зросла, беручи до уваги і те, що останнім часом все більше людей різного віку віддають перевагу все ж онлайн покупкам, в період ізоляції це скоріше вимушені дії. Тому різного роду компанії активно роблять акцент безпосередньо на онлайн магазинах. І щоб ринок не перенасичувався, Інтернет магазин має бути максимально простим у користуванні та швидким в обробці запитів, адже користувач – нетерплячий, тому з легкістю може звернутись до іншого магазину. До того ж велику роль грає і інтуїтивний інтерфейс, саме це як найкраще характеризує будь-який сайт, його зрозумілість та зручність.

Для успішного просування Інтернет магазину, варто проводити аналіз цільової аудиторії, що в свою чергу дає більше розуміння на кого слід орієнтуватись при наповненні категорій товару. Можливо для більшого заохочення впроваджувати купонні коди, ігри, які допомагають клієнтам заробляти бали або купони. Людям подобається розважатися, важливо знайти правильний підхід.

Також визначено та описано основні етапи, з яких складається процес розробки програмного модуля. Дослідження структури веб-сайту та головних складових для ефективної роботи Інтернет магазину для користувачів.

Проаналізовано переваги та недоліки таких технологій як: Node.js, Express.js, MongoDB, JQuery, Bootstrap, Pug та середовища програмування Visual Studio Code.

Проведено розробку структури, алгоритмів програми та тестування Інтернет магазину, що дало гарний результат.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Structuring an Online Store [Електронний ресурс] – Режим доступу: https://www.i95dev.com/how-to-structure-an-online-store/
2. Importance of E-Commerce and online shopping and why to sell Online. [Електронний ресурс] – Режим доступу: https://medium.com/@nyxonedigital/importance-of-e-commerce-and-online-shopping-and-why-to-sell-online-5a3fd8e6f416
    1. Визначення веб-сайтів [Електронний ресурс] – Режим доступу: http://www.webtec.com.ua/uk/articles/index/view/2011-05-05/web-site
2. Браун Э. Изучаем JavaScript. Руководство по созданию современных WEB-сайтов / Э. Браун. – 2017. – 363с. – ISBN: 978-617-7812-55-4.
3. MongoDB [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/MongoDB
4. Node.js [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Node.js
5. JQuery [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/JQuery
6. Express.js [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Express.js
7. Bootstrap [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Bootstrap
8. Грамотная структура интернет-магазина [Електронний ресурс] – Режим доступу: http://prodex.ua/blog/gramotnaya-struktura-internet-magazina/
9. What Makes a Website User-Friendly? [Електронний ресурс] – Режим доступу: https://www.intechnic.com/blog/what-makes-a-website-user-friendly/
10. Веб зсередини [Електронний ресурс] – Режим доступу: http://www.zhu.edu.ua/mk_school/mod/page/view.php

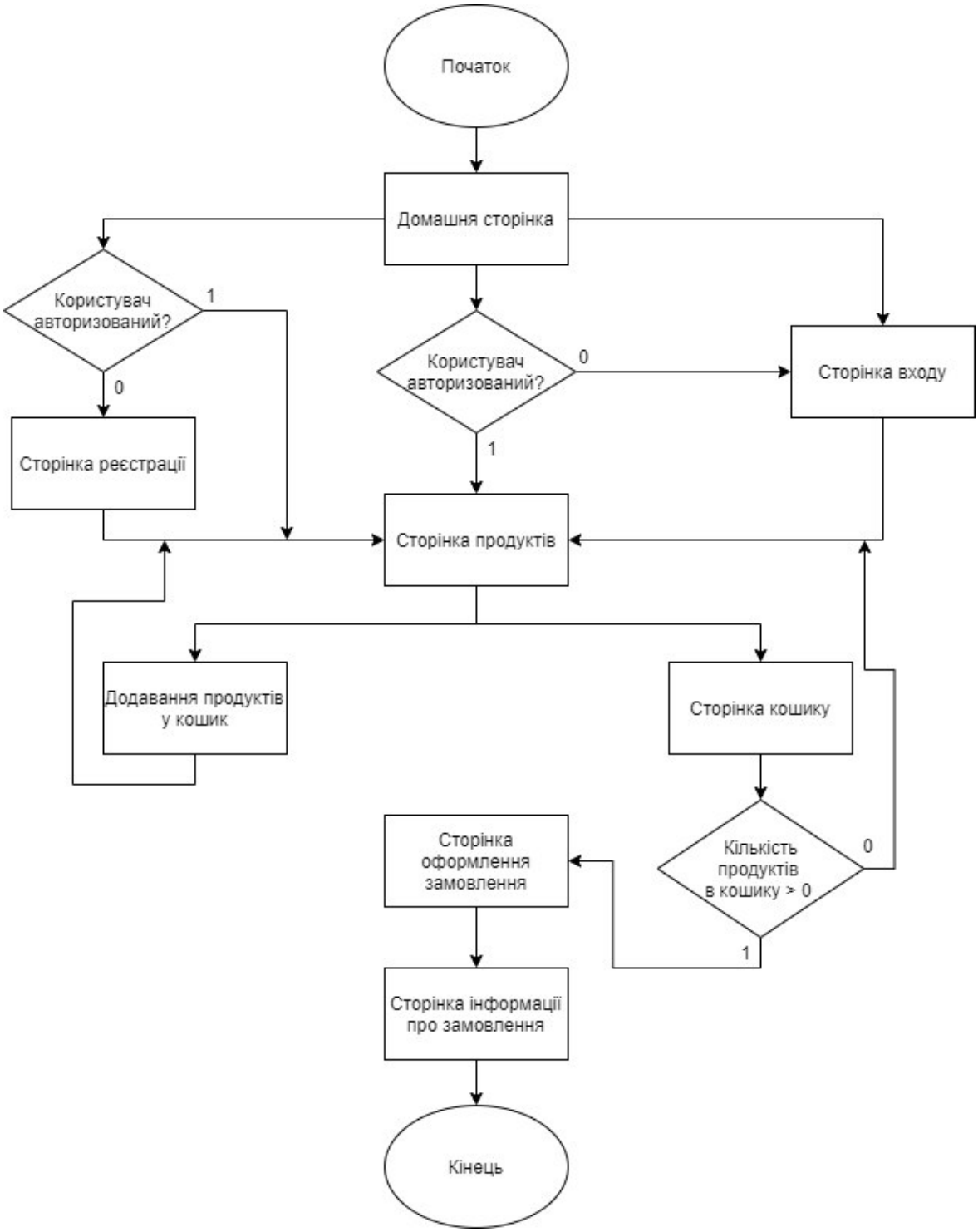
11. Брэд Дейли. Node.js, MongoDB and Angular Web Development: The definitive guide to using the MEAN stack to build web applications / Брэд Дейли, Брендан Дейли, Калеб Дейли. – 2020р. – 656с. – ISBN: 978-5-6040044-8-7
12. Шапошников И. Интернет-программирование / Шапошников И. – М.: «Книга по Требованию», 2013. – 362 с. – ISBN: 978-5-941570-24-9.
13. 15. Байдачный С. SQL Server 2012. Навчальний посібник / Байдачный. С. – М.: «Солон-Пресс», 2005 – 347 с. – ISBN: 5-98003-244-4
14. Анализ юзабилити сайта, изучение внешнего вида и эффективности работы сайта (usability) [Електронний ресурс] – Режим доступу: http://www.m-mix.com.ua/marketing/analiz-usability.html
15. Tips for Building a User-Friendly Website [Електронний ресурс] – Режим доступу: https://www.commonplaces.com/blog/10-tips-for-building-a-user-friendly-website/

ДОДАТКИ

Додаток А (обов’язковий) Блок-схема роботи Інтернет магазину

					08-02.БДР 003.16.000.ПД			
					Розробка та наповнення сайту інтернет магазину з використанням бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кабак В.Д.*						
Перевір.		*Софина О.Ю.*				*Арк. 1*	*Аркушів 3*	
Т. Контр.		*Софина О.Ю.*			Блок-схема роботи інтернет			
Реценз.								

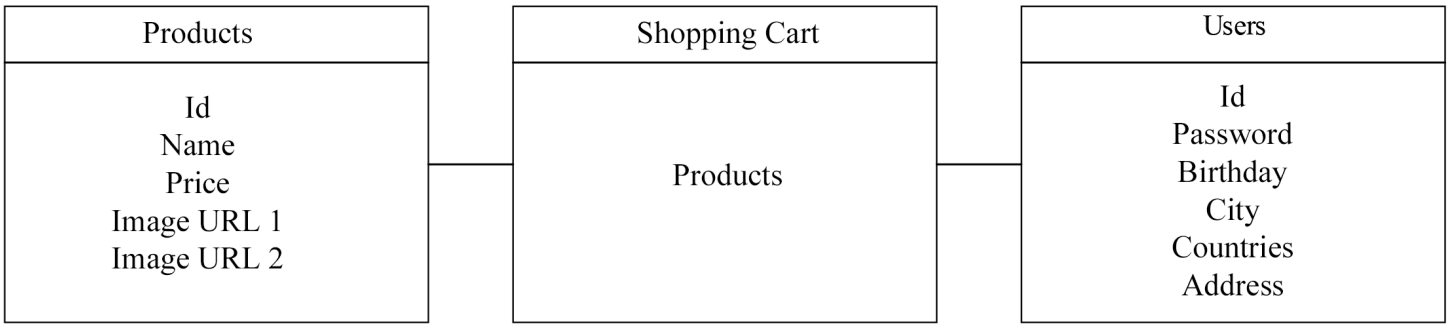
Додаток А (обов’язковий) Блок-схема роботи Інтернет магазину



Додаток Б (обов’язковий) Структура бази даних

					08-02.БДР 003.16.000.ПД			
					Розробка та наповнення сайту інтернет магазину з використанням бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кабак В.Д.*						
Перевір.		*Софина О.Ю.*				*Арк. 1*	*Аркушів 3*	
Т. Контр.		*Софина О.Ю.*						
Реценз.		*1*			Структура бази даних	1СІ-16б		

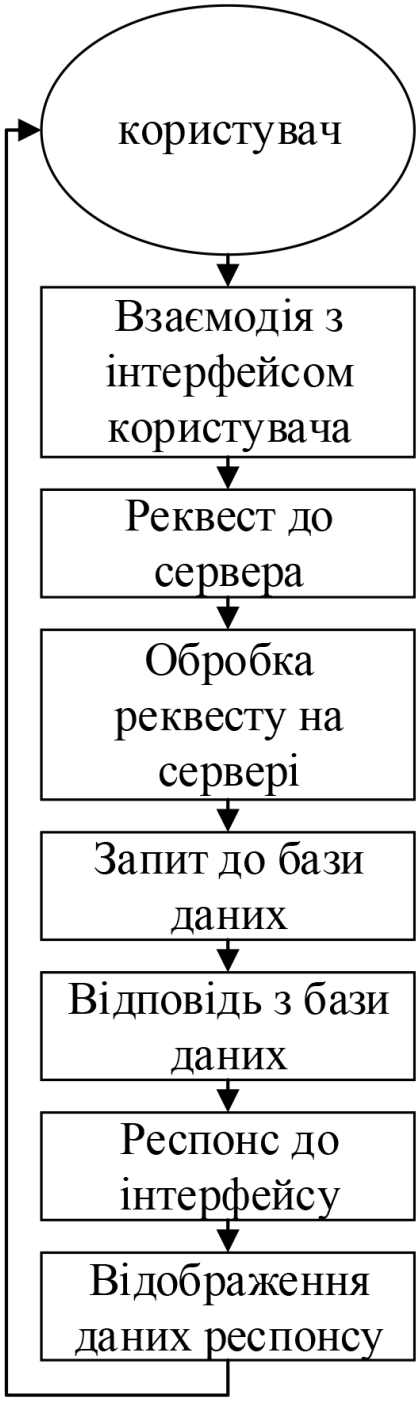
Додаток Б (обов’язковий) Структура бази даних



Додаток В (обов’язковий) Діаграма послідовності роботи програми

					08-02.БДР 003.16.000.ПД		
					Розробка та наповнення сайту інтернет магазину з використанням бібліотеки JQuery та фреймворку Bootstrap на платформі NodeJS	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кабак В.Д.*				*Арк. 1*	*Аркушів 3*	
Перевір.		*Софина О.Ю.*						
Т. Контр.		*Софина О.Ю.*			Діаграма послідовності роботи програми	1СІ-16б		
Реценз.								

Додаток В (обов’язковий) Діаграма послідовності роботи програми



Додаток Г (обов’язковий) Лістинг програми

app.js

```
const express = require("express");
const MongoClient = require("mongodb").MongoClient;
const ObjectId = require("mongodb").ObjectID;
const session = require('express-session');
const bodyParser = require('body-parser');

const app = express();
const jsonParser = express.json();
const cookieParser = require('cookie-parser');

const mongoClient = new MongoClient("mongodb://localhost:27017/", {
useNewUrlParser: true });

let dbClient;

mongoClient.connect(function(err, client){
  if(err) return console.log(err);
  dbClient = client;
  app.locals.db = client.db("webdis");
  app.listen(3000, function(){
      console.log("Сервер ожидает подключения...");
  });
});

app.set("view engine", "pug");
```

```
app.use(session({secret: 'ssshhhhh',saveUninitialized: true,resave: true}));
app.use(cookieParser('secret key'));

app.use('/assets', [
  express.static(__dirname + '/node_modules/jquery/dist'),
  express.static(__dirname + '/FE')
]);


const urlencodedParser = bodyParser.urlencoded({extended: false});

app.get("/", function (request, response) {
  response.render("index", {});
});

app.get("/plp", function(request, response){
  if (!request.session.user) {
    response.redirect('/login');
  } else {
    const productsCollection = request.app.locals.db.collection('products');
    productsCollection.find({}).toArray(function(err, prods){
      if(err) return console.log(err);
      response.render("plp", {
        pArr: prods
      });
    });
  }
});
```

```
app.get("/basket", function(request, response){
  if (!request.session.user) {
    response.redirect('/login');
  } else {
    const productsCollection = request.app.locals.db.collection('products');
    var listOfIds = request.cookies.products ? request.cookies.products.split(',') : [];
    var documentIds = listOfIds.map(function(myId) { return ObjectId(myId); });
    productsCollection.find({_id: {$in: documentIds }}).toArray(function(err,
prods){
        if(err) return console.log(err);
        response.render("basket", {
          pArr: prods
        });
    });
  }
});

app.get("/register", function (request, response) {
    response.render("register", {});
});


app.get("/logout", function (request, response) {
  request.session.user = '';
  response.redirect('/login');
});

app.post("/register", urlencodedParser, function (request, response) {
```

```
    if(!request.body) return response.sendStatus(400);
    const usersCollection = request.app.locals.db.collection('users');
    usersCollection.insertOne( {
      name: request.body.userName,
      password: request.body.password}, function(err, result){
        if(err) {response.send('Користувач з таким логіном існує')}
        else {
          request.session.user = result.ops[0].name;
          response.redirect('/plp');
        }

    });
});

app.get("/login", function (request, response) {
  if (request.session.user) {
    response.redirect('/plp');
  }
  response.render("login", {});
});

app.post("/login", urlencodedParser, function (request, response) {
  if(!request.body) return response.sendStatus(400);
  const usersCollection = request.app.locals.db.collection('users');
  usersCollection.findOne({name: request.body.userName, password:
request.body.password}, function(err, result){
    if(err) response.send(err);
    if (!result) {
      response.send('Користувача не знайдено');
```

```
    } else {
    request.session.user = result.name;
    response.redirect('/plp');
    }
  });
});

process.on("SIGINT", () => {
  dbClient.close();
  process.exit();
});
```

basket.pug

```
<html>

    <head>

        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q
9Ifjh" crossorigin="anonymous">

        <script src="assets/jquery.min.js"></script>

        <script src="assets/cart.js"></script>
```

```
    </head>

    <body>

        <a href="/" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Index page</a>

        <a href="/plp" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Back to products</a>

        <div class="row justify-content-md-center">

            <div class="js-container col-md-auto">

                each p in pArr

                    <div class="card" style="width: 18rem;">

                        <img src="#{p.img}" class="card-img-top" alt="...">

                        <div class="card-body">

                            <h5 class="card-title">#{p.title}</h5>

                            <p class="card-text">#{p.description}</p>

                            <button type="button" data-id="#{p._id}" class="btn btn-danger
js-delete-product">DELETE FROM BASKET</button>
```

```
                </div>

            </div>

        </div>

    </div>

    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao" crossorigin="anonymous"></script>

    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifw
B6" crossorigin="anonymous"></script>

  </body>

</html>
```

index.pug

```
<!DOCTYPE html>
```

```
<html>

<head>

    <title>Вхід</title>

    <meta charset="utf-8" />

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q
9Ifjh" crossorigin="anonymous">

    <script src="assets/jquery.min.js"></script>

    <script src="assets/index.js"></script>

</head>

<body>

    <button class="btn btn-success btn-lg active js-ajax">AJAX</button>

    <a href="/register" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Register</a>

    <a href="/login" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Login</a>
```

```
    <a href="/plp" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Products</a>

    <a href="/logout" class="btn btn-danger btn-lg active" role="button" aria-
pressed="true">Вийти</a>

    <div class="js-ajax-result">

    </div>

    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao" crossorigin="anonymous"></script>

    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifw
B6" crossorigin="anonymous"></script>

</body>

</html>
```

login.pug

```
<!DOCTYPE html>

<html>

<head>

    <title>Вхід</title>

    <meta charset="utf-8" />

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q
9Ifjh" crossorigin="anonymous">

    <script src="assets/jquery.min.js"></script>

</head>

<body>

    <a href="/" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Index page</a>

    <h1>ВХІД</h1>
```

```
    <form action="/login" method="post">
      <div class="form-group">
        <label for="userName">Ім'я</label>
        <input type="text" class="form-control" name="userName" id="userName"/>
      </div>
      <div class="form-group">
        <label for="password">Пароль</label>
        <input type="password" class="form-control" name="password"
id="password"/>
      </div>
        <button type="submit" class="btn btn-primary">Відправити</button>
    </form>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao" crossorigin="anonymous"></script>
```

```
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifw
B6" crossorigin="anonymous"></script>

</body>

</html>
```

plp.pug

```
<html>

  <head>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q
9Ifjh" crossorigin="anonymous">

    <script src="assets/jquery.min.js"></script>

    <script src="assets/plp.js"></script>

  </head>
```

```
<body>

    <a href="/" class="btn btn-primary btn-lg active" role="button" aria-pressed="true">Index page</a>

    <a href="/basket" class="btn btn-primary btn-lg active" role="button" aria-pressed="true">Basket</a>

    <div class="row justify-content-md-center">

      <div class="js-container col-md-auto">

        each p in pArr

          <div class="card" style="width: 18rem;">

            <img src="#{p.img}" class="card-img-top" alt="...">

            <div class="card-body">

              <h5 class="card-title">#{p.title}</h5>

              <p class="card-text">#{p.description}</p>

              <button type="button" data-id="#{p._id}" class="btn btn-primary js-add-product">ADD</button>

            </div>
```

```
            </div>

        </div>

    </div>

    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao" crossorigin="anonymous"></script>

    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifw
B6" crossorigin="anonymous"></script>

  </body>

</html>
```

register.pug

```
<!DOCTYPE html>

<html>
```

```
<head>
    <title>Регистрация</title>
    <meta charset="utf-8" />
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q
9Ifjh" crossorigin="anonymous">
    <script src="assets/jquery.min.js"></script>
</head>
<body>
    <a href="/" class="btn btn-primary btn-lg active" role="button" aria-
pressed="true">Index page</a>
    <h1>Введите данные</h1>
    <form action="/register" method="post">
      <div class="form-group">
        <label for="userName">Ім'я</label>
```

```
        <input type="text" class="form-control" name="userName" id="userName"/>
      </div>
      <div class="form-group">
        <label for="password">Пароль</label>
        <input type="password" class="form-control" name="password"
id="password"/>
      </div>
        <button type="submit" class="btn btn-primary">Відправити</button>
    </form>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao" crossorigin="anonymous"></script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifw
B6" crossorigin="anonymous"></script>
```

```
</body>

</html>
```

cart.js

```
$(document).ready(function () {

  Array.prototype.remove = function(value) {
    var idx = this.indexOf(value);
    if (idx != -1) {
        return this.splice(idx, 1);
    }
    return false;
  }

  function deleteCookie(name) {
      setCookie(name, "", {
        'max-age': -1
      })
    }

  function getCookie(name) {
      let matches = document.cookie.match(new RegExp(
        "(?:^|; )" + name.replace(/([\.$?*|{}\(\)\[\]\\\/\+^])/g, '\\$1') + "=([^;]*)"
      ));
      return matches ? decodeURIComponent(matches[1]) : undefined;
    }
```

```
 function setCookie(name, value, options = {}) {

  options = {
   path: '/',
   // при необходимости добавьте другие значения по умолчанию
   ...options
  };

  if (options.expires instanceof Date) {
   options.expires = options.expires.toUTCString();
  }

  let updatedCookie = encodeURIComponent(name) + "=" + value;

  for (let optionKey in options) {
   updatedCookie += "; " + optionKey;
   let optionValue = options[optionKey];
   if (optionValue !== true) {
    updatedCookie += "=" + optionValue;
   }
  }

  document.cookie = updatedCookie;
 }

$('.js-delete-product').on('click', function (e) {
  e.preventDefault();
  var cookieValue = getCookie('products');
```

```
    var cookieArr = cookieValue.split(',');
    cookieArr.remove($(this).data('id'));
    setCookie('products', cookieArr.join(','));
    $(this).closest('.card').remove();
  });
})
```

index.js

```
$(document).ready(function () {
  $('.js-ajax').on('click', function () {
    $.ajax({
      url: 'https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json',
      success: function(data){
        console.log("Результат запиту:");
        console.log(data);
        console.log("Результат запиту (stringify):");
        console.log(JSON.stringify(data));
```

```
    },
    error: function(err) {
      console.log(err);
    }
  });
 })
});
```

plp.js

```
$(document).ready(function () {
  function deleteCookie(name) {
    setCookie(name, "", {
      'max-age': -1
    })
  }
  function getCookie(name) {
    let matches = document.cookie.match(new RegExp(
```

```
    "(?:^|; )" + name.replace(/([\.$?*|{}\(\)\[\]\\\/\+^])/g, '\\$1') + "=([^;]*)"
  ));
  return matches ? decodeURIComponent(matches[1]) : undefined;
}

function setCookie(name, value, options = {}) {

  options = {
    path: '/',
    // при необходимости добавьте другие значения по умолчанию
    ...options
  };

  if (options.expires instanceof Date) {
    options.expires = options.expires.toUTCString();
  }

  let updatedCookie = encodeURIComponent(name) + "=" + value;

  for (let optionKey in options) {
    updatedCookie += "; " + optionKey;
    let optionValue = options[optionKey];
    if (optionValue !== true) {
      updatedCookie += "=" + optionValue;
    }
  }

  document.cookie = updatedCookie;
}
```

```
  $('.js-add-product').on('click', function (e) {
      e.preventDefault();
      if (getCookie('products') && getCookie('products').indexOf($(this).data('id'))
!== -1) {
        return
      }
      var cookieValue = getCookie('products') ? getCookie('products') + ',' +
$(this).data('id') : $(this).data('id');
      setCookie('products', cookieValue);
      $(this).addClass('disabled btn-success').removeClass('btn-
primary').empty().text('ADDED');
  });
})
```