Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Лопатіна Богдана Михайловича*
(ПІБ)

академічної групи *121-18-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка Інтернет магазину комп'ютерної техніки на базі технології Figma*

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

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-8-2* *Лопатіна Б.М.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка Інтернет магазину комп'ютерної техніки на базі технології Figma*

затверджена наказом ректора НТУ «ДП» від *18 травня 2022 р.;* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

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

Завдання прийняв до виконання _____ *Лопатін Б.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 68 с., 29 рис., 3 табл., 3 дод., 26 джерел.

Об’єкт розробки: багатофункціональний інтернет-магазин комп’ютерної техніки та комплектуючих для неї на базі Figma, TypeScript та Angular.

Мета кваліфікаційної роботи: створення програмного забезпечення для реалізації самого інтернет-магазину та панелі адміністраторів. Через магазин користувачі матимуть змогу переглядати каталог товарів, оформлювати замовлення та оплачувати їх. Панель має надавати керівниками можливість керувати каталогом магазину, відслідковувати вхідні замовлення, керувати статусом замовлень, пришвидшувати процес оформлення товарів, показувати звіт діяльності магазину.

У вступі розглядається аналіз поставленої задачі, детальніше розглядається мета кваліфікаційної роботи та обґрунтовується її актуальність.

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

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

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

Практичне значення полягає у створенні програмного додатку, що надає можливість майбутнім користувачам сервісу ознайомлюватися з запропонованим переліком товарів, додавати бажане до персонального кошика та оплачувати сформоване замовлення.

Актуальність даного програмного продукту визначається переходом сучасного світу зі звичайних магазинів до онлайн-шопінґу, що дозволяє власникам мінімізувати витрати на утримання такого магазину та підвищити ефективність та кількість замовлень через інтернет.

Список ключових слів: САЙТ, КОМП’ЮТЕР, БАЗА ДАНИХ, КАТАЛОГ, АЛГОРИТМ, ПРОЄКТУВАННЯ, МЕНЮ, ЗАСТОСУНОК, МАГАЗИН, ІНТЕРНЕТ.

ABSTRACT

explanatory note: 68 pages, 29 pictures, 3 tables, 3 additions, 26 sources.

Object of development: a multifunctional online store of computer equipment and components based on Figma, TypeScript and Angular.

The purpose of the qualification work: creation of software for the implementation of the online store and the admin panel. Through the store, users will be able to view the catalog of goods, place orders and pay for them. The panel should provide managers with the ability to manage the catalog of the store, track incoming orders, manage the status of orders, speed up the process of registration of goods, show a report on the activities of the store.

The introduction considers the analysis of the set task and the purpose of qualifying work, substantiates its urgency.

The first section analyzes the subject branch, determines the urgency of the task and the purpose of development, formulate the statement of the task, implements the requirements to the software, and specifies technologies and software.

The second section analyzes the available solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of technical means.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance is to create a software application that allows future users of the service to get acquainted with the proposed list of products, add the desired to the personal basket and pay for the order.

The relevance of this software product is determined by the transition of the modern world from regular stores to online shopping, which allows owners to minimize the cost of maintaining such a store and increase the efficiency and number of orders online

List of keywords: PROGRAM, COMPUTER, INFORMATION SYSTEM, CATALOG, ALGORITHM, DESIGN, MENU, APPLICATION, SHOP, INTERNET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – HyperText Markup Language;

CSS – Cascading Style Sheets;

SCSS – Sassy Cascading Style Sheets;

TS – TypeScript;

E-commerce – електронна комерція;

B2C – Business-to-customer;

ID – ідентифікаційний номер;

IT – інформаційні технології;

DOM – Document Object Model;

NPM – Node Package Manager.

JSON – JavaScript Object Notation;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

БД – база даних;

Гб – гігабайти;

МГц – мегагерц.

ВСТУП

Тематика даної кваліфікаційної роботи присвячена розробці макету багатофункціонального інтернет-магазину у середовищі Figma та його верстці на базі Angular та TypeScript у середовищі VSCode.

Метою даної кваліфікаційної роботи є створення програмного забезпечення для реалізації самого інтернет-магазину та панелі адміністраторів. У процесі роботи буде вивчено інструментальні засоби та принципи роботи сервісу Figma, функціонал якої забезпечує можливість повноцінно створити макет інтернет-магазину, на основі якого згодом буде проведена верстка з використанням мов HTML, SCSS, TypeScript, а також бібліотеки Angular.

На сьогоднішній день майже неможливо уявити повноцінне існування бізнесу без функціонування в інтернеті. E-commerce — або ж електронна комерція — стала важливою частиною сучасної фінансової системи, через яку щоденно проходять мільярди доларів, а роздрібна торгівля – одна з найбільших її сфер. Звичайним споживачам завжди зручніше оформити замовлення прямо зі свого комп’ютера, без потреби їздити до певного магазину та купляти потрібний товар самостійно.

Шляхів для ведення роздрібної торгівлі через інтернет існує безліч: від продажу товарів через сторінки у соцмережах, і до виходу на B2C-маркетплейси, наприклад: Amazon, eBay, Rozetka, OLX. Створення власного повноцінного онлайн-магазину — це, звісно, найзручніший та найпрестижніший варіант, який дозволяє ще гучніше заявити про свій бізнес на ринку та ефективніше керувати каталогом доступних товарів та процесом оформлення замовлень з боку споживачів.

Головними перевагами створення інтернет-магазину є: розширення доступу до каталогу ваших товарів на всю територію країни чи світу; можливість швидко оновлювати дані та публікувати новини про зміни в роботі магазину чи акційні пропозиції; зворотній зв’язок від клієнтів; проведення масштабних рекламних кампаній через інтернет.

Реалізація інтернет-магазин на сьогодні не представляє собою велику проблему, ринок повний різноманітних пропозицій, все залежить лише від бюджету замовника.

Результатом виконання даної роботи є сайт, що представляє собою інтернет-магазин з функціями перегляду каталогу, наповнення кошика, оформлення замовлення та його оплати. До додаткових завдань відноситься створення повноцінного макету сайту за допомогою сервісу Figma та налагодження зв'язку між самим сайтом та його бекендом.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Ринок E-commerce стрімко розвивається у світі, захоплюючи все більший відсоток усієї світової роздрібної торгівлі. Особливий стрибок у розвитку інтернет-торгівлі можна помітити в останні роки. Пандемія коронавіруса, локдаун та інші обмеження зіграли велику роль в масовій популяризації закупівель через інтернет. Це також стосується і ринку України.

За даними дослідження українського ринку e-commerce (рис.1.1)[1] за 2020 рік, електронна комерція зайняла 8.8% від загального об’єму ринку роздрібної торгівлі. Це в три рази більше, ніж у 2016 році, а на 2025 рік обіцяють зріст ще в два рази. Російсько-українська війна точно зробить власні коригування, але навіть незважаючи на це, e-commerce демонструє неймовірний розвиток, який не збирається зупинятися. Саме через це абсолютна більшість підприємців сьогодні або додає свої товари на великі маркетплейси, де сплачує невелику комісію, або звертається до IT-сектора для створення власної WEB-сторінки, що буде представляти їх магазин.

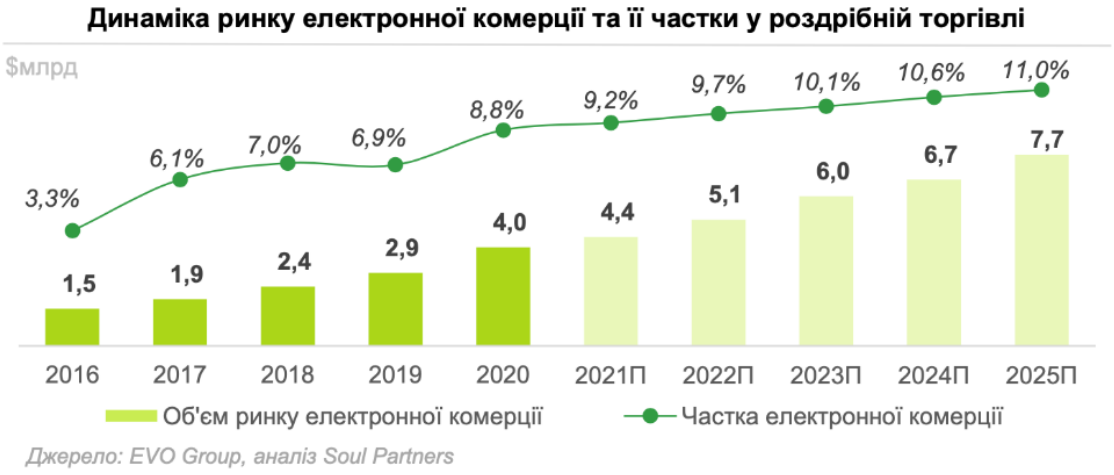


Рис. 1.1 Динаміка розвитку ринку e-commerce в Україні

Як уже було помічено раніше, створення інтернет-магазину у 2022 році – це нескладна проблема, хоча й одна з першочергових. За останні десятиліття у світі повноцінно сформувався ринок розробників e-commerce сайтів чи застосунків, як front-end, так і back-end, які пропонують різноманітні послуги та працюють на базі різних мов програмування, сервісів, бібліотек. Основою основ при розробці WEB-сторінок залишаються HTML, CSS та JavaScript, однак сьогодні складно знайти хороших програмістів, які б користувалися виключно цими мовами без різноманітних фреймворків, метамов тощо.

Фреймворк або каркас – це набір готових до використання програмних рішень, які включають дизайн, логіку та базову функціональність системи чи підсистеми. Фреймворк містить у собі спеціальні програми, скрипти, бібліотеки коди та інші елементи, що мають полегшити та покращити процес написання коду. На сьогодні найпопулярнішими фреймворками для JavaScript є Vue, React та Angular.

1) Vue.js – наймолодший фреймворк з трьох названих, був створений у 2014 році, як альтернатива проблемному на той момент Angular та все ще новому React. Основними перевагами Vue вважають:

– Маленький розмір, що дозволяє покращити продуктивність проєктів;

– Дуже детальну документацію, що дозволяє швидко його опанувати та серйозно зекономити час при написанні коду;

– Адаптивність. Перейти на Vue з React та Angular досить легко через схожість їхніх архітектур та дизайнів.

Найголовнішим недоліком Vue усе ще є невелика база розробників, бо саме вони займаються його розвитком, коли за Angular та React відповідають техногіганти Google та Facebook відповідно.

2) React – це JavaScript-бібліотека для створення користувацьких інтерфейсів, яку дуже часто навіть не називають фреймворком. Причиною для цього є структуризація React по різних бібліотеках для більшої гнучкості. Маючи лише React, користувач не зможе повноцінно працювати, оскільки для цього

доведеться встановити окремі бібліотеки (Redux, React-router тощо), які в Angular, наприклад, будуть додані одразу. З переваг у React:

– Велика кількість додаткових бібліотек, які добре розширюють можливості розробника;

– Повна сумісність між усіма версіями, через що з оновленням React не з'являється потреби перевстановлювати застарілі модулі;

– Підтримка віртуального DOM, за допомогою якого React ефективніше оновлює DOM самого браузера.

Проблеми в React з документацією, якою, зазвичай, займаються самі розробники та великою кількістю неочікуваних оновлень, які вимагають опанування.

3) Angular – це front-end фреймворк, написаний мовою TypeScript, та розроблений компанією Google на базі фреймворку AngularJS. Щодо переваг:

– Angular працює на мові програмування TypeScript, яка глобально розширює можливості JavaScript, робить код приємнішим на вигляд та спрощує процес написання;

– В Angular багатофункціональна компонентна система, яка дозволяє відокремлювати верстку сайту та інші елементи, що також дає можливість використовувати їх у інших проєктах

– Велика кількість різноманітних модулів прямо «з коробки» пришвидшують процес роботи

У проєкті використовується саме Angular, через вищеперелічені пункти, серед яких особливо варто відмітити найперший. Мова програмування TypeScript відрізняється від звичайного JavaScript підтримкою повноцінних класів, використанням уже згаданих модулів, а також статичною типізацією — можливістю явного визначення типів об'єктів.

Замість стандартного CSS у проєкті використовується SCSS – метамова на базі звичайного CSS, яка розширяє можливості базової мови: додає вкладені правила, змінні, функції, міксини та інше.

Але перед тим, як починати роботу над front-end частиною сайту, треба зробити повноцінний його макет, від якого можна буде відштовхуватися при верстці проєкту. Вибір пав на сервіс Figma, який має всі потрібні функції навіть у безкоштовній версії та працює прямо у браузері на будь-якому пристрої. Особливо на фоні конкурентів його виділяє можливість створення компонентів – елемент, властивості якого можна передавати одразу на декілька окремих елементів.

1.2 Призначення розробки та галузь застосування

Об’єктом розробки даного проекту є інтернет-магазин, реалізований як веб-застосунок «TechnoShop», у якому користувачі можуть купляти комп’ютерну техніку та комплектуючі за різними категоріями. Магазин має містити головну сторінку, каталог з можливістю сортувати за категоріями, сторінки товарів, з яких можна буде додавати бажане прямо до кошику, після чого оплачувати замовлення.

Не варто забувати про візуальну частину сервісу, яка відповідатиме стандартам сучасного UX/UI дизайну, притримуватиметься одного візуального коду, буде приємна оку та інтуїтивно зрозуміла.

Додаток буде підключено до панелі адміністраторів, через яку регулюватимуться усі замовлення та база даних товарів. Їх можна буде редагувати, видаляти, додавати нові та змінювати статус. Крім того адміністраторові буде доступна інформація про клієнтів. Завдяки збереженню даних у БД уся інформація про магазин та замовлення триматиметься структурованою.

База даних керуватиметься з backend-серверу, через який контролюватимуться властивості товарів, а також зберігатиметься інформація про користувачів та їх замовлення.

1.3 Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проєкту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

- освітня програма 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» 268-с від 18 травня 2022 р.;
- завдання на кваліфікаційну роботу на тему «Розробка інтернет-магазину комп’ютерної техніки на базі технології Figma».

1.4 Постановка завдання

Завданням кваліфікаційної роботи є створення макету та розробка frontend-частини інтернет-магазину продажу комп’ютерів та комплектуючих для комп’ютерних систем.

Програмне забезпечення призначене для відображення товарів, що зберігаються в backend-частині інтернет-магазину, та можливості оформити на основі цих товарів повноцінне замовлення. Програма повинна формувати web-сторінки на основі готового макету з використанням інформації з БД; надавати користувачеві можливість переглядати сторінки, додавати товари до кошика, спорожнювати кошик та оформлювати замовлення.

Для реалізації макету використовуватиметься онлайн-сервіс Figma, а для реалізації готового на основі макету сайту — HTML, SCSS та TypeScript.

При керуванні готовою WEB-сторінкою існуватимуть дві основні ролі: адміністратор та користувач. Адміністратор зможе контролювати наповнення інтернет-магазину через панель адміністраторів, яка зв’язана з головним сайтом

та backend-сервером через БД. Адміністратор буде здатен вносити зміни у каталог, замовлення, інформацію про користувачів тощо. Користувач, у свою чергу, буде взаємодіяти виключно з самим магазином.

Вихідну інформацію матимуть обидві головні сторінки: панель адміністраторів та сам інтернет-магазин. Перша формуватиме та відправлятиме детальну інформацію про всі товари у каталозі до інтернет-магазину, де до неї матимуть доступ користувачі. Друга, в свою чергу, навпаки до панелі адміністраторів відправлятиме інформацію про новосформоване замовлення користувача.

Вхідна інформація поступатиме також до обидвох сторінок. Адміністратор через свою панель отримуватиме масив даних, що зберігатиме у собі інформацію про замовлення від користувачів, тоді як користувач отримуватиме також масив даних, але він триматиме інформацію про товари, доступні для покупки.

Функціонування завдання припиняється при відсутності в користувача сайту стабільного інтернет-з’єднання, застарілої версії браузера, чи програмної несумісності.

1.5 Вимоги до програми або програмного виробу

1.5.1 Вимоги до функціональних характеристик

Вхідними даними для користувачів, як уже було зазначено, стануть товари та категорії, до яких вони належать. Це буде два різних об’єкти, кожен з яких складатиметься зі своїх полів, у яких зберігатиметься інформацію про ці об’єкти. До панелі адміністраторів, у свою чергу, вхідні дані потраплятимуть у форматі JSON за допомогою backend-серверу.

Вихідні дані про замовлення користувачів потраплятимуть до панелі адміністраторів через БД у вигляді того ж об’єкту, який зберігатиме у собі поля різнотипних даних про замовлення. Вихідними даними від адміністратора стануть готові об’єкти товарів, категорій тощо. Усі вищезгадані об’єкти будуть реалізовані в окремих TypeScript-інтерфейсах.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми та створення допустимих умов безпеки необхідно:

– реалізувати доступ до панелі адміністраторів виключно за допомогою логіну та паролю, які додаються до системи разом з інформацією про адміністраторів;

– використовувати метод JSON Web Token для шифрування кожного окремого користувача. Цей метод створюватиме та надаватиме доданим до системи користувачам власний токен, який зберігатиме у собі їхній ID та допомагати в ідентифікації;

– шифрувати пароль кожного користувача та адміністратора за допомогою метода BlowFish з бібліотеки bcrypt.js, який проводить блочно симетричне шифрування паролю;

– слідкувати та вчасно оновлювати всі додатки, що використовуються при реалізації інтернет-магазину для підтримки стабільної роботи та відсутності проблем з безпекою;

– реалізувати обробку виняткових ситуацій при роботі з даними для запобігання можливих помилок.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для стабільного функціонування інтернет-магазину комп’ютер на базі операційної системи Windows має відповідати наступним вимогам:

- Центральний процесор (ЦП): Intel Core i5;
- Відеоадаптер: Intel(R) Iris® Xe Graphics;
- Відеопам’ять: 4 Гб;
- Накопичувач: 520 ГБ;
- Оперативна пам’ять: 32 Гб;
- Доступ до мережі Internet.

Для стабільного функціонування інтернет-магазину комп’ютер на базі операційної системи macOS має відповідати наступним вимогам:

– ЦП: Intel Core i7;
– Відеоадаптер: Radeon Pro 555X;
– Відеопам’ять: 4 ГБ;
– Накопичувач: 520 ГБ;
– Оперативна пам’ять: 16 Гб;
– Доступ до мережі Internet.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для стабільного функціонування інтернет-магазину програмне забезпечення комп’ютера має відповідати наступним вимогам:

− операційна система: Windows (7+), Linux, MacOs Monterey;

− веб-браузер: Firefox / Google Chrome / Opera / Microsoft Edge / Safari.

Frontend-частина сервісу має бути сумісна з мовами програмування TypeScript та JavaScript з використанням фреймворку Angular.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

За завдання кваліфікаційної роботи було реалізовано frontend-частину інтернет-магазину комп'ютерних комплектуючих. Її функціональне призначення представляє собою формування макету та реалізації web-сторінок на основі інформації, отриманої з бази даних. Програма має надавати користувачеві змогу передивлятися каталог усіх товарів та сортувати їх за категоріями, ознайомлюватися з детальною інформацією про кожну окрему позиції, заповнювати потрібними товарами особистий кошик, та сплачувати готове замовлення онлайн. Отримані від користувача дані сервіс має відправляти до бази даних, звідки вони потраплять на панель адміністраторів, де перейдуть під контроль адміністраторів сайту. Панель адміністраторів, в свою чергу, повинна надавати керівникам сайту можливість керувати каталогом товарів сайту, відслідковувати всі вхідні замовлення та змінювати їхній статус. Крім того, в ній має формуватися статистика магазину: кількість проданих товарів, загальна сума прибутку тощо.

З експлуатаційної точки зори програма надає замовнику повноцінний WEB-ресурс, який дозволить поширити діяльність бізнесу на мережу Internet та, відповідно, збільшити прибуток компанії завдяки ширшому охопленню потенційних клієнтів. Замовник також отримає зручний інструмент керування інтернет-магазину у вигляді панелі адміністраторів, яка не тільки дозволить редагувати та збільшувати базу даних з товарами, а й виводитиме інформацію про загальні показники сервісу, що допоможе краще розуміти рівень популярності новоствореного магазину.

2.2 Опис застосованих математичних методів

Оскільки особливості поставленого завдання не передбачають використання математичних методів, при розробці frontend-частини інтернет-магазину математичні методи застосовані не були.

2.3 Опис використаної архітектури та шаблонів проектування

Створене програмне забезпечення за завданням кваліфікаційної роботи використовує у собі архітектуру «клієнт-сервер». Вона, виходячи з назви, складається з двох основних частин – клієнту та серверу, які зв’язані один з одним за допомогою мережевого протоколу.

Взаємодія між клієнтом та сервером проходить таким чином (рис 2.1): клієнт, тобто комп’ютер, телефон чи інший пристрій користувача, при виконанні певних дій на сайті, відправляє запит до сервера, який у відповідь має надати очікувані дані клієнту.

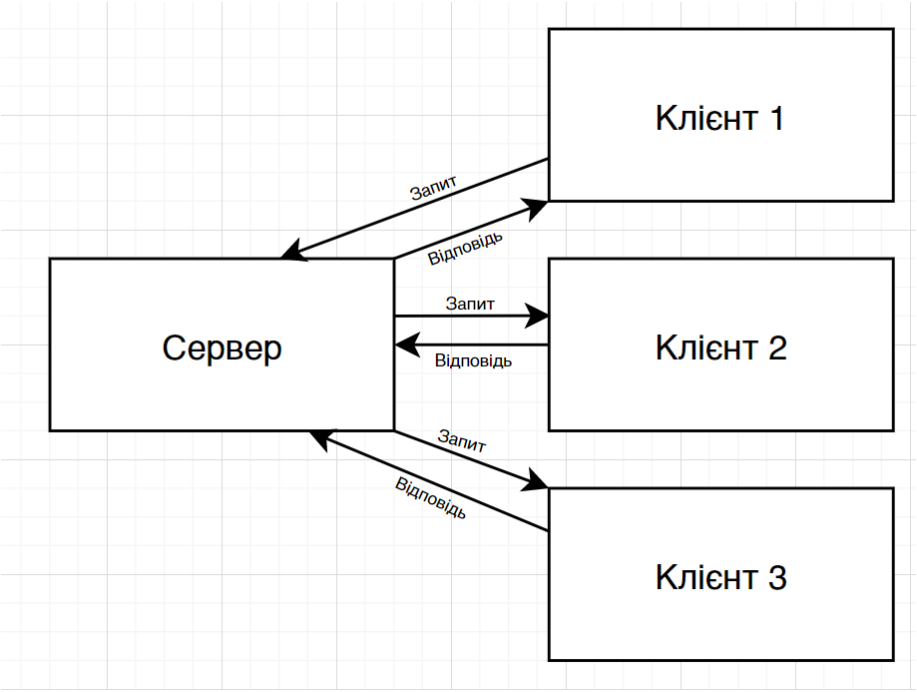


Рис 2.1 Класична модель клієнт-серверу

Як уже було зазначено раніше, для нормальної взаємодії клієнта та сервера використовується мережевий протокол. Мережевий протокол представляє собою набір правил, який регулює взаємозв'язок між комп'ютерами, що знаходяться в одній мережі. Типізація мережевих протоколів багаторівнева, тобто для стабільного функціонування мережі використовуються одразу декілька різних протоколів, кожен з яких знаходиться на власному рівні.

У створеному за завданням кваліфікаційної роботи додатку використовується клієнт-сервер архітектура, створена на базі MEAN-стеку (рис 2.2). Це комплекс програмного забезпечення серверної частини, що складається з чотирьох компонентів: MongoDB, ExpressJs, AngularJs та NodeJs.

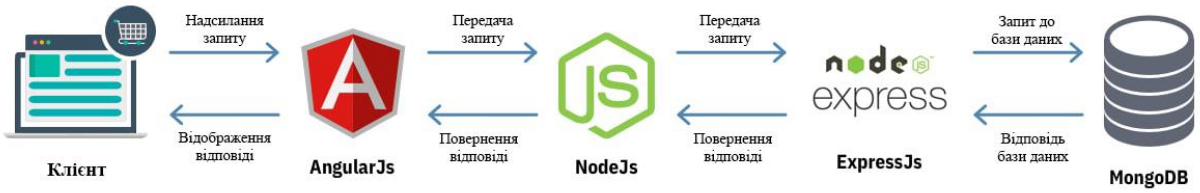


Рис 2.2 Схема MEAN-стеку

Процес обробки інформації проходить таким чином:

– Клієнт надсилає запит до Angular, який займається його обробкою та передає далі;

– З Angular він потрапляє до NodeJs, де той парситься та передається до фреймворку Express, який уже напряму поєднаний з базою даних MongoDB;

– Отримавши оброблений запит, MongoDB «дістає» зі своєї бази даних потрібну інформацію та повертає її тим же шляхом прямо до Angular;

– Angular виводить результат запиту користувачеві.

Архітектура «клієнт-сервер» зручна у використанні саме через можливість обробки запитів від декількох клієнтів, ефективний доступ до ресурсів мережі та

зменшення навантаження на мережу в цілому через те, що більшу частину обробки даних на себе бере саме серверна частина.

2.4 Опис використаних технологій та мов програмування

Розробка заданого проєкту почалася з розробки макету сайту в середовищі онлайн-сервісу Figma. Frontend-частина сервісу була реалізована за допомогою мови TypeScript з використанням фреймворку Angular.

Figma – це графічний редактор, створений у 2016 році, для розробки дизайнів інтерфейсів та прототипів web-сторінок. Метою розробників було створення безкоштовного браузерного застосунку, який полегшить розробникам та дизайнерам роботу з графічними об’єктами.

З’явившись на ринку графічних редакторів, Figma дуже швидко – всього за декілька років – вирвалася у лідери, обійшовши справжніх редакторів-«ветеранів» InVision, Sketch та Adobe XD (рис 2.3)[2]. Таки успіх Figma отримала через власну інноваційність та велику кількість зручних функцій, якими на той момент не відрізнялися конкуренти.

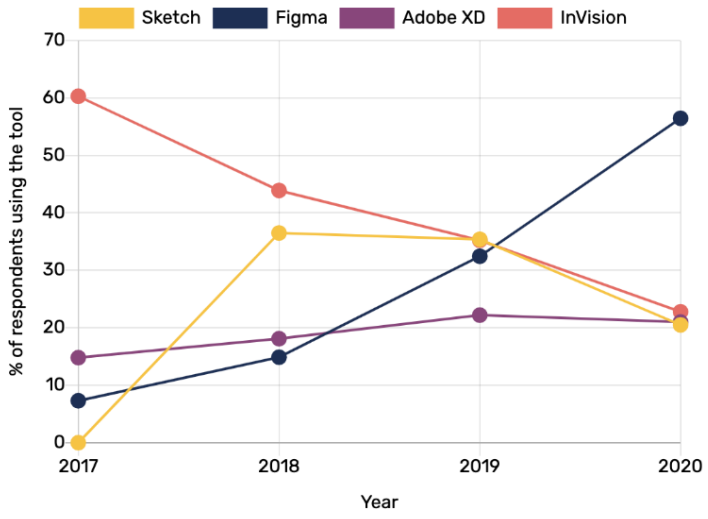


Рис 2.3 Найпопулярніші редактори серед користувачів

Перш за все, Figma повністю працездатний умовно-безкоштовний браузерний застосунок, який не вимагає встановлювати жодних додаткових модулів на комп'ютер користувача. Треба лише авторизуватися. Умовно-безкоштовний застосунок через те, що Figma має платну преміум-версію, однак майже всі потрібні функції та можливості доступні й при безоплатному користуванні.

По-друге, Figma робить великий акцент на кооперативних можливостях та командній роботі. Всі проєкти користувача знаходять у хмарному сховищі, тож над одним макетом можуть одночасно працювати одразу декілька людей, які зможуть бачити всі додані зміни у режимі реального часу. При бажанні, своїм проєктом можна поділитися через звичайне посилання, зникає потреба зберігати його окремим файлом. Крім того для групових робіт існує можливість залишати коментарі прямо в середині проєкту.

По-третє, Figma дуже легко та зрозуміла в користуванні з безліччю зручних функцій, кількість яких, за бажанням, можна збільшити за допомогою бібліотеки розширень, які створюють інші користувачі сервісу. Наприклад, створені елементи макету одразу мають CSS-сніпети, які можна згодом імплементувати при верстці сайту. Також Figma дозволяє створювати компоненти – спеціальні комплексні елементи, копії якого зберігають всі властивості батьківського компонента навіть у випадку їх змін. Це дозволяє економити час, редагуючи лише один елемент, але роблячи зміни одразу в багатьох місцях проєкту.

Для реалізації front-end частини web-додатку використовувалася мова програмування TypeScript на базі фреймворку Angular.

Angular – фреймворк з відкритим кодом, створений у 2016 році на базі AngularJS корпорацією Google. Головною його метою є спрощення розробки великих за розміром додатків та застосунків за допомогою модулей та компонентів. Основною мовою програмування для Angular є TypeScript, хоча розробники залишили можливість програмувати за допомогою класичного Javascript. Для зручної реалізації UI-елементів було додано колекцію компонентів PrimeNG.

TypeScript – мова програмування, що базується на JavaScript, але дещо видозмінює його. Найголовнішим оновленням вважається додавання визначення статичного типу. Завдяки цьому з’являється можливість описувати форми об’єктів, що покращує документацію, а також дозволяє TypeScript перевіряти код на правильність. Використання типів не є обов’язковою вимогою, однак воно може допомогти уникнути небажаних помилок.

Мовою розмітки гіпертексту та мовою стилізації обрані HTML та SCSS відповідно, оскільки вони залишаються лідерами за зручністю та практичністю у своїх сферах.

2.5 Опис структури програми та алгоритмів її функціонування

Frontend-частина додатку поділяється ще на дві частини: клієнтську панель та панель адміністраторів. Зв’язок між ними відбувається через сервер та базу даних.

Клієнтська панель складається з (рис 2.4):

– домашньої сторінки, з якої користувач починає знайомство з сайтом. На ній знаходяться посилання на каталог, кошик, окремі категорії товарів тощо;

– каталогу, який тримає у собі всі товари, додані до бази даних з можливістю їх сортування за категоріями;

– сторінку товару, що детальніше знайомить покупця з окремими позиціями, а також дозволяє додавати їх прямо до кошику з вказанням кількості;

– кошику, де можна перевірити додані товари, за потреби видалити їх, дізнатися суму замовлення та перейти до оплати;

– сторінка оплати, де вводяться усі необхідні дані про клієнта та з’єднують його з зовнішнім сервісом для сплати замовлення;

– сторінки логіну/реєстрації, які активуються при спробі сплатити замовлення.

Рис 2.4 Структура клієнтської панелі

Файлова структура умовно розподілена на три частини та знаходиться у трьох різних папках(рис 2.5):

–/techno-base містить у собі компонент, у якому зберігається візуальна форма усіх сторінок: header, footer та основна частина; компонент домашньої сторінки, а також папку з ассетами, що використовуються на сторінках;

–/libs зберігає компоненти всіх інших сторінок, окремих їхніх елементів та методів;

–/styles тримає у собі scss файли для оформлення усіх сторінок проєкту.

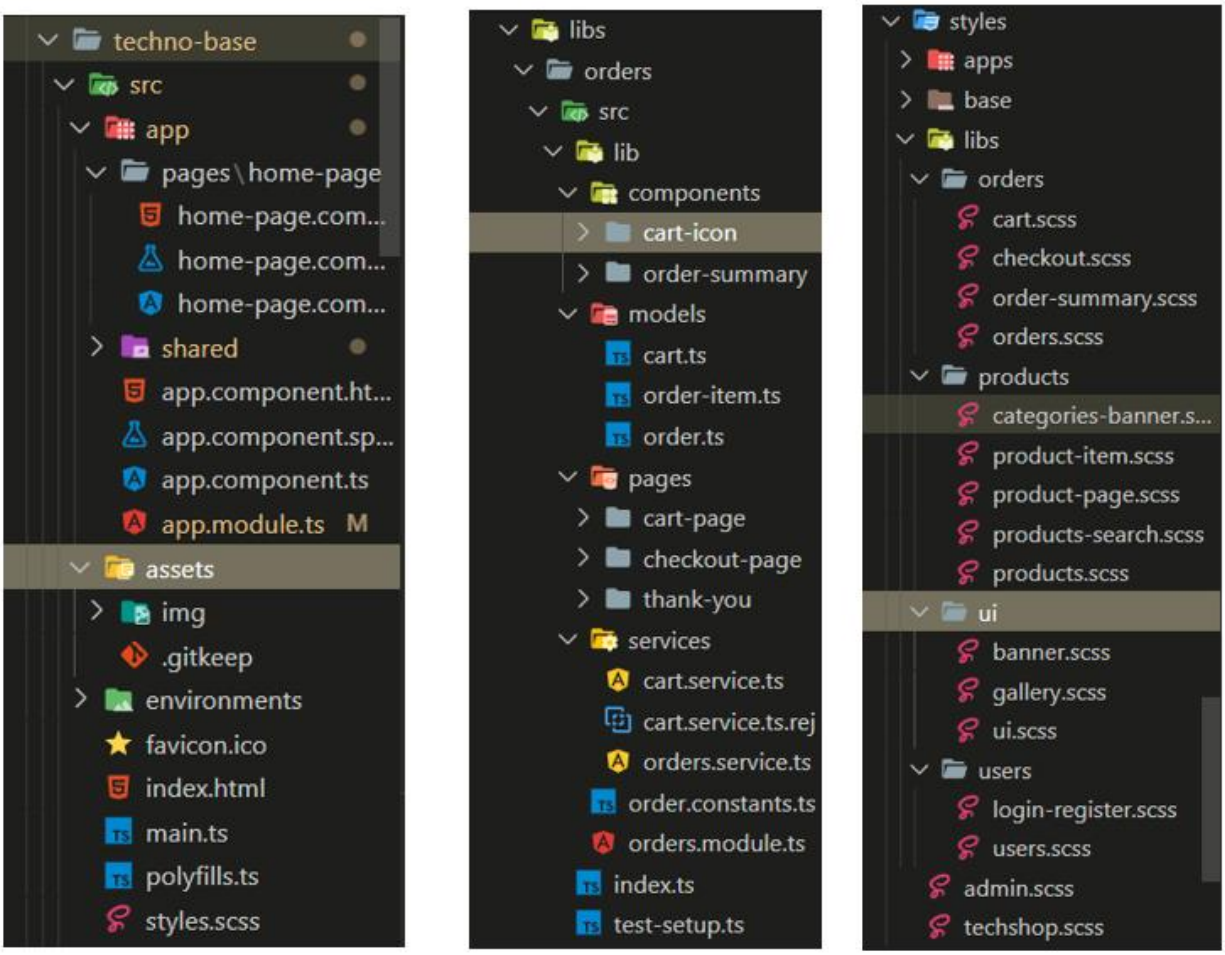


Рис 2.5 Файлова структура

На рисунку 2.6 візуалізована схема маршрутизації клієнтської панелі

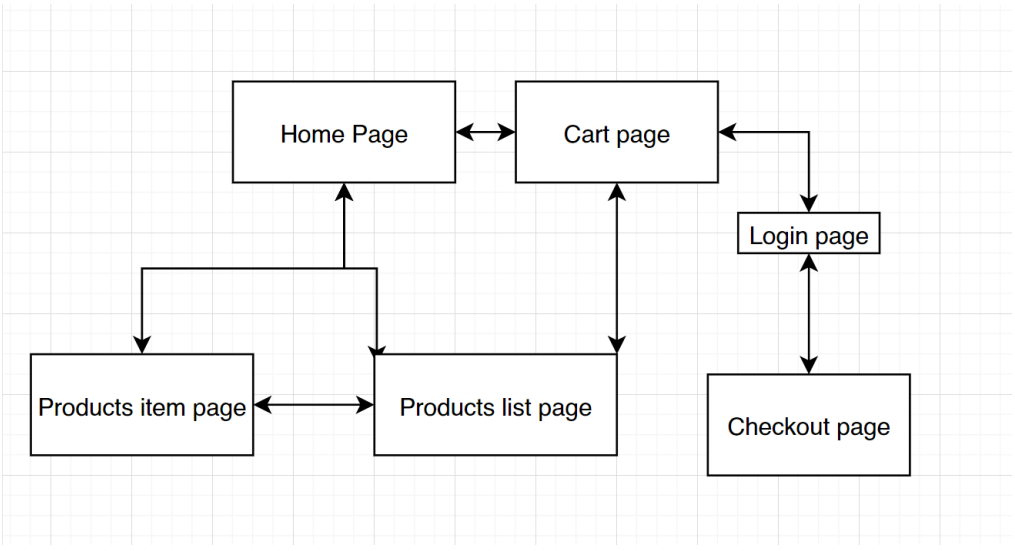


Рис 2.6 Маршрутизація клієнтської панелі

2.6 Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними клієнтської частини frontend'у є текстові дани з інформацією про товари, категорії та користувачів у форматі JSON. Ці об'єкти даних мають поля різних типів, зчитуються та виводяться на екран користувачеві. Опис об'єктів відбувається за допомогою створення інтерфейсів.

Вивід даних може відбуватися двома способами: або це звичайне звертання до полів об'єктів та їх вивід у вигляді тексту, або це звертання до конкретного об'єкту за допомогою його id та заповнення готових input-форм.

Таблиця сутностей 2.1 відображає атрибути та типи даних цих атрибутів об'єкту Product.

Поля id, name, description та image передаються звичайними типами string, price – типом number; category вимагає передачі класу об'єкту Category (таблиця 2.2), який складається з інформації про категорію товару; isFeatured – спеціальне поле для визначення, чи має товар зображуватися в окремій категорії на головній сторінці.

Таблиця 2.1

Таблиця «Product»

Назва поля	Тип даних
id	string
name	string
description	string
image	string
category	Category
price	number
isFeatured	boolean

Таблиця 2.2

Таблиця «Category»

Назва поля	Тип даних
id	string
name	string
icon	string
color	string
checked	boolean

Отримані з цього об’єкту та об’єкту Category дані виведуться у клієнтському сервісі (рис 2.7). Аналогічно працює ситуація зі звертання по id користувача (рис 2.8).

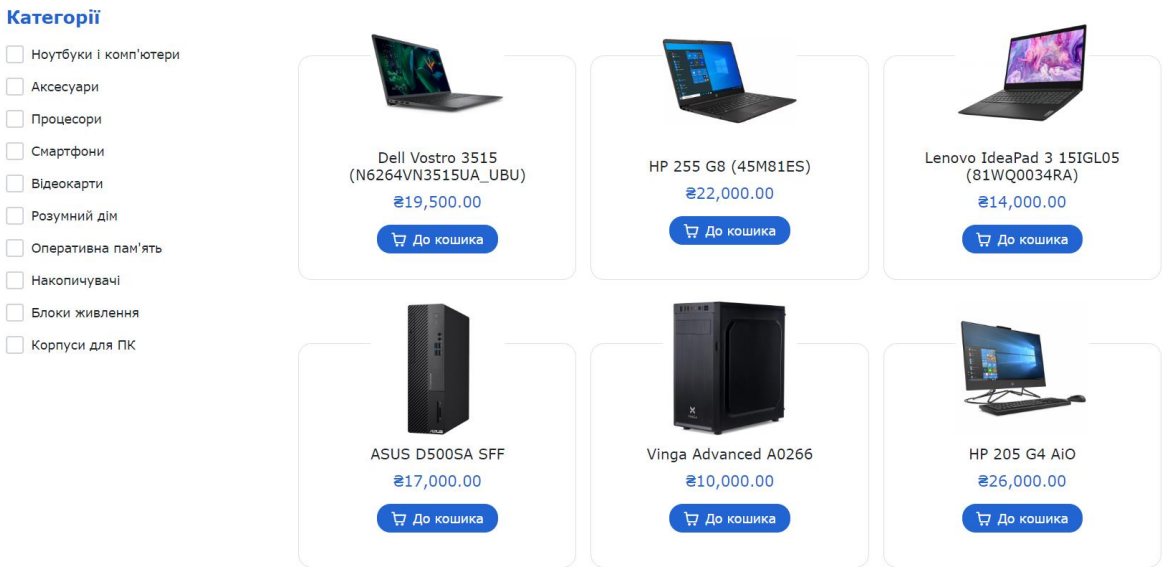


Рис 2.7 Приклад реалізації вхідних даних

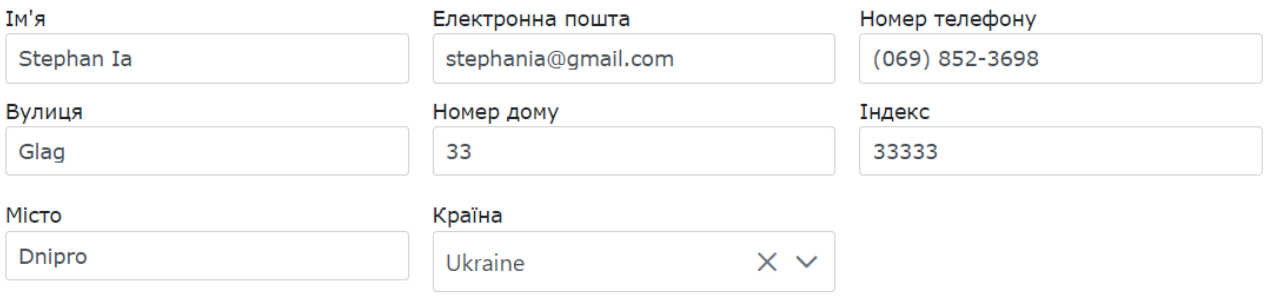


Рис 2.8 Приклад реалізації вхідних даних

Вихідні дані клієнтського сервісу представлятиме все той же текстовий файл у форматі JSON, що передаватиме всю потрібну інформацію про отримане від користувача замовлення до бази даних, звідки вона потраплятиме до панелі адміністраторів. Для передачі вихідних даних використовується клас об’єкту Order, у якому зберігається детальна інформація про замовника, а також про саме замовлення (таблиця 2.3).

Таблиця 2.3

Таблиця «Order»

Назва поля	Тип даних
id	string
orderItems	OrderItem[]
shippingAddress1	string
city	string
zip	string
country	string
phone	string
status	number
totalPrice	string
user	string
dateOrdered	string

Більшість полів зберігають у собі звичайні текстові дані, що передаються типом string; поле orderItems передається масивом даних OrderItem[], у якому тримається інформація про товари, які були замовлені; поле status має тип number для передачі конкретного номеру статусу з відповідного файлу, де зберігаються всі можливі статуси замовлення.

Приклад вихідних даних, що отримає панель адміністраторів можна побачити на рисунку 2.9.

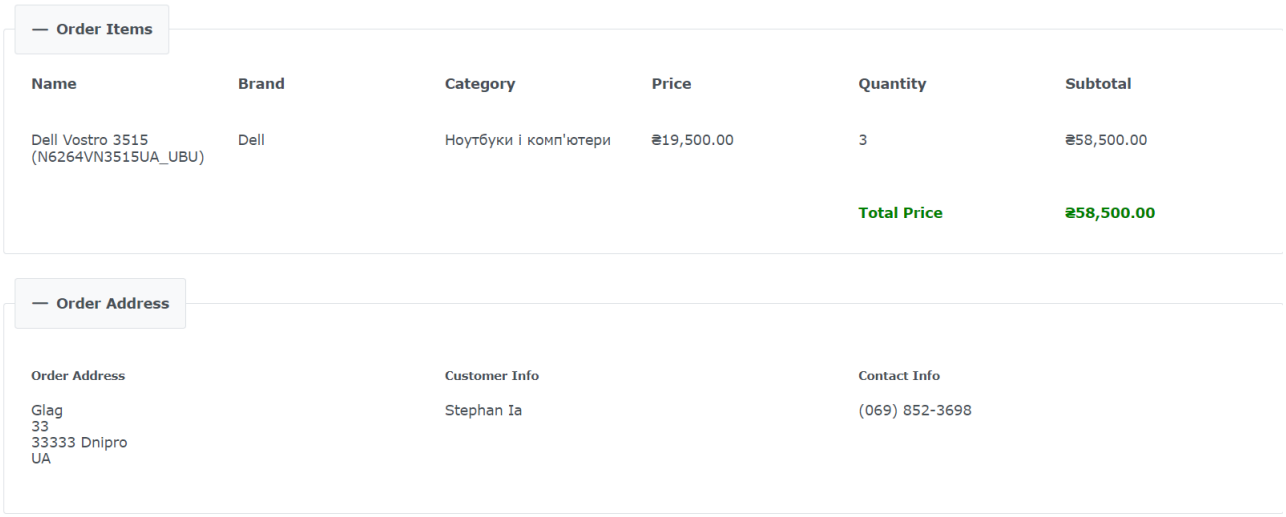


Рис 2.9 Приклад реалізації вихідних даних з клієнтського сервісу в панелі адміністраторів

2.7 Опис розробленого програмного продукту

2.7.1 Використані технічні засоби

Для повноцінного та стабільного функціонування frontend-частини інтернет-магазину рекомендованими характеристиками є:

- Центральний процесор: AMD Ryzen 7 3700X 8 core 16 threads;
- Відеоадаптер: GeForce RTX 2070 SUPER;
- Відеопам’ять: 8 Гб;

– Оперативна пам’ять: 16Гб DDR4 3200 МГц;
– Накопичувач: 1 ТВ.
– доступ до мережі Internet;
– маніпулятор "миша";
– клавіатура.

Наведені вище технічні характеристики забезпечать виконання усіх функцій програми відповідно до вимог щодо надійності, швидкості, обробки даних і безпеки, які були висунуті замовником.

2.7.2 Використані програмні засоби

Для створення макету, на базі якого було зверстано сайт, використовувався графічний редактор Figma, який не вимагає встановлювати на комп’ютер жодних додатків – увесь процес редагування макету сайту відбувається прямо у вікні браузера.

Для покращення роботи з фреймворком Angular було встановлено Angular CLI – інтерфейс командного рядка, що спрощує роботу над додатком, даючи можливість керувати ним прямо з командного рядку. Це допомогло прискорити та оптимізувати роботу.

Для реалізації командної роботи використовувався застосунок Sourcetree.

Sourcetree – це клієнт для повноцінної роботи з кодом Git, а також зручний менеджер проєктів GitHub, що спрощує процес роботи декількох розробників над одним проєктом.

Робота з кодом проводилася в середовищі Visual Studio Code.

Visual Studio Code – це безкоштовний редактор коду, що підтримує мови програмування JavaScript, TypeScript та Node.js, а також інші мови, які можна додатково завантажити через систему розширень.

Розробка неможлива також без використання менеджеру пакунків NPM, який є стандартом для середовища NodeJs. Через менеджер NPM відбувається робота з реєстром програмного забезпечення, що також несе назву npm.

Для налаштування зв’язку з backend-частиною проєкту потрібно встановити середовище NodeJS.

Node.js — це платформа з відкритим вихідним кодом для високопродуктивних веб-додатків, що були написані на JavaScript. Її особливість полягає у тому, що вона дозволяє реалізовувати та оброблювати код JavaScript на сервері застосунку, а не прямо у браузері, як це було раніше. Саме завдяки NodeJS JavaScript перейшов з рангу браузерних мов до мов загального використання.

2.7.3 Виклик та завантаження програми

Завантаження програми починається з роботи у командній строчці проєкту. Клієнтська частина не зможе повноцінно функціонувати без працюючого серверу – для цього у рядку треба ввести команду «npm run server» (рис. 2.8). Вона запустить сервер та надасть йому докальний порт 3000.

```
PS D:\University\Diploma Part Two\Techno-Base> npm run server

> my-team@0.0.0 server D:\University\Diploma Part Two\Techno-Base
> nodemon backend/app.js

[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node backend/app.js`
server is running http://localhost:3000
we are using eshop-database
Database Connection is ready...
```

Рис. 2.8 Запуск серверної частини додатку

Сервер працює, можна переходити до запуску сторінки інтернет-магазину. Робить це в тому ж терміналі. Для запуску сайту треба використати команду «npx nx serve techno-base». Вона запустить сторінку та розмістить її на локальному порті localhost:4200. Панель адміністраторів запускається аналогічним методом,

але варто враховувати, що при паралельному запуску у сторінок мають бути різні номери портів.

```
PS D:\University\Diploma Part Two\Techno-Base> npx nx serve techno-base

> nx run techno-base:serve:development

✓ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   3.22 MB |
styles.css, styles.js | styles        |   1.04 MB |
polyfills.js          | polyfills     | 294.90 kB |
main.js               | main          | 180.09 kB |
runtime.js            | runtime       |   6.52 kB |

                      | Initial Total |   4.72 MB

Build at: 2022-06-16T22:23:26.182Z - Hash: 003c6e955cd7f67d - Time: 7390ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
```

Рис 2.9 Запуск клієнтського сервісу

Відкрити готовий локальний порт можна двома методами: ввести localhost:4200 прямо у пошуковому рядку браузера, або ж з зажатою клавішею «ctrl» клацнути по посиланню у терміналі. При успішному запуску програми ви побачити головну сторінку інтернет-магазину.

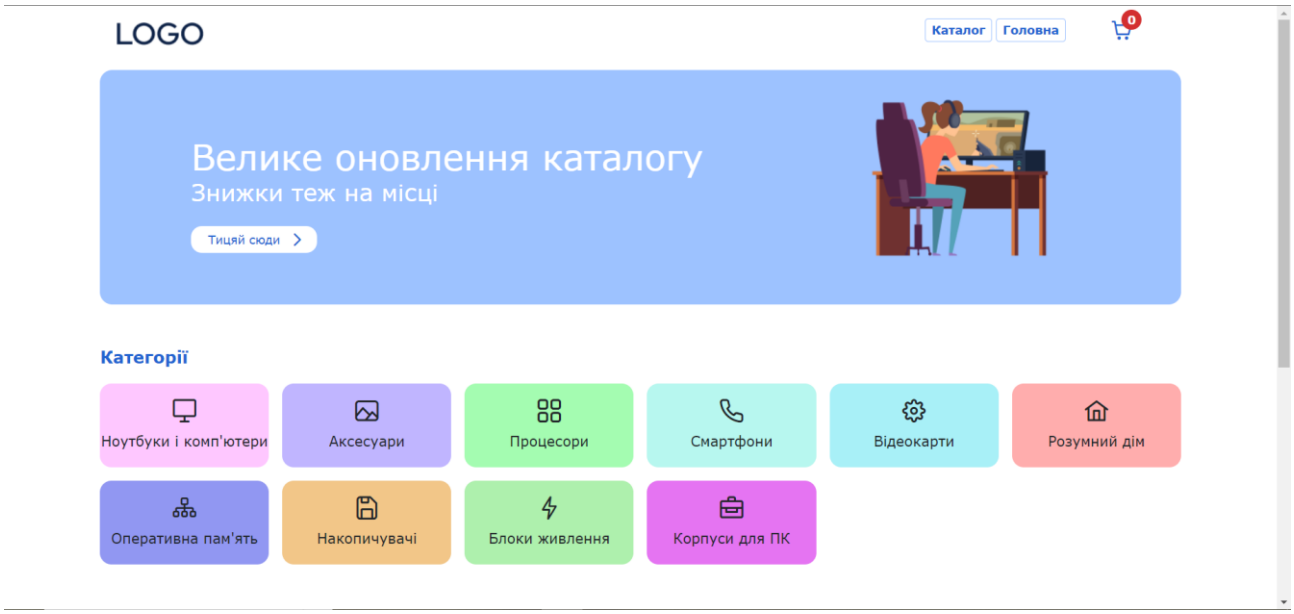


Рис. 2.10 Головна сторінка інтернет-магазину

2.7.4 Опис інтерфейсу користувача

Попередньо інтерфейс клієнтської частини був створений у середовищі онлайн-сервісу Figma (рис. 2.11) з використанням внутрішньої функції компонентів (рис. 2.12) та інтеракцій між елементами (рис. 2.13)

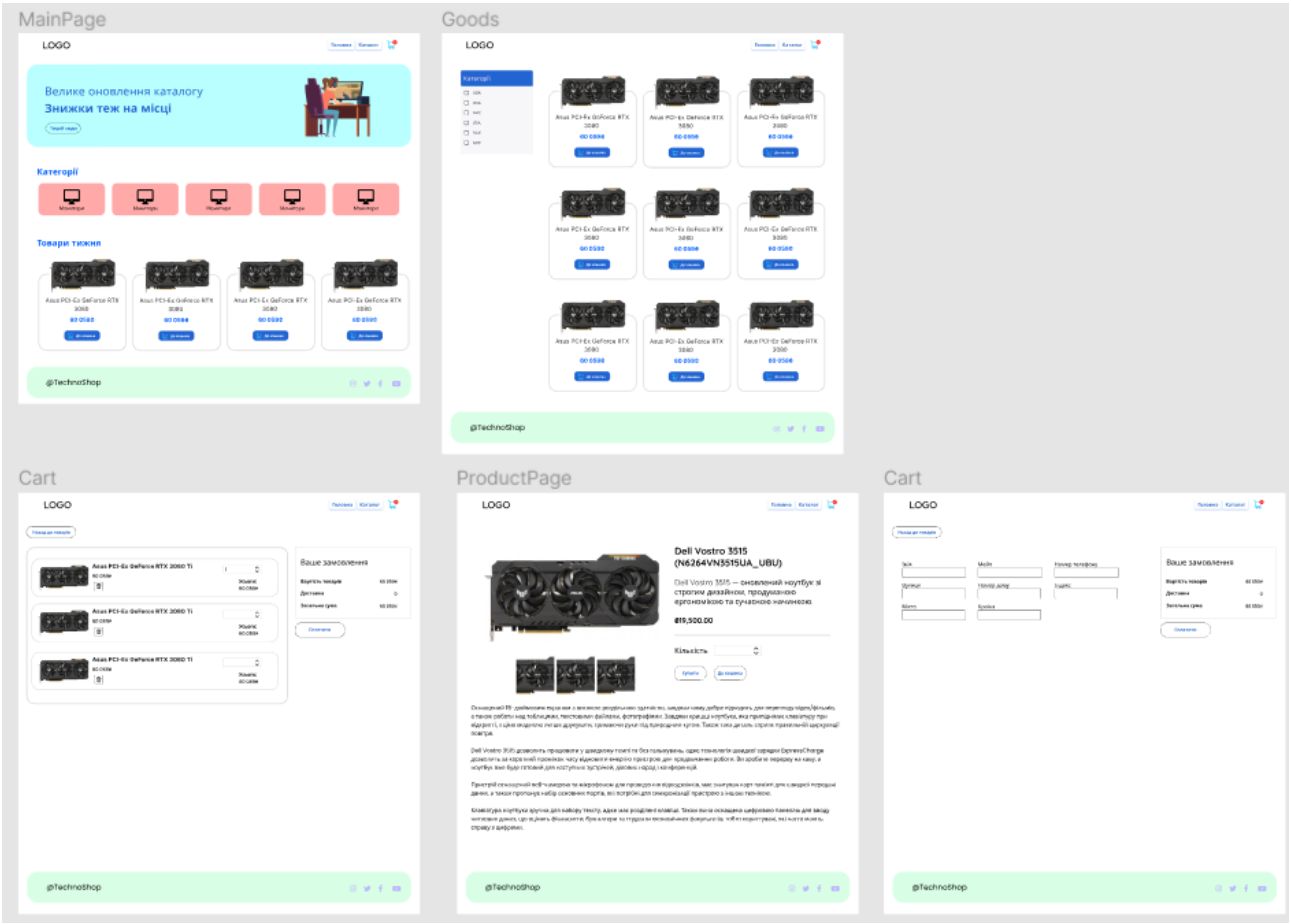


Рис. 2.11 Створений макет сайту у середовищі Figma

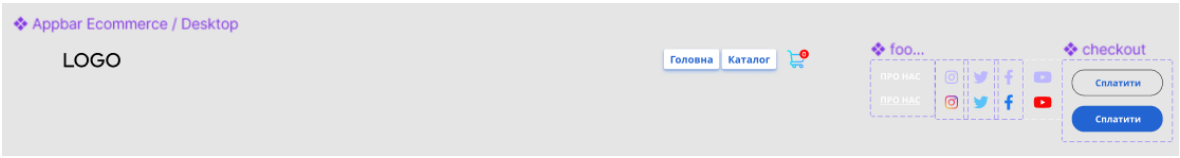


Рис. 2.12 Компоненти, що використовуються у макеті

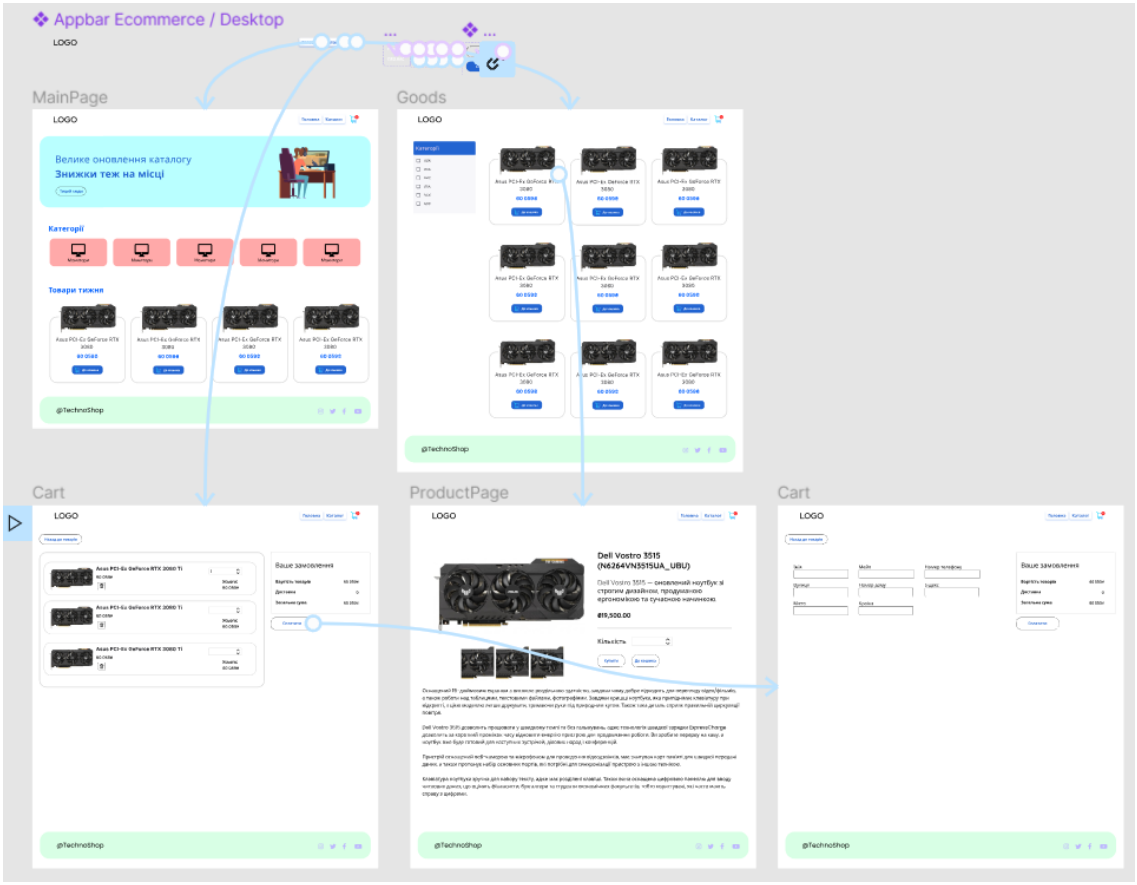


Рис. 2.13 Інтерації між сторінками

Головна сторінка, на яку ви потрапили після запуску локального серверу, складається з декількох основних частин.

Header або «шапка» сайту (рис. 2.11) зберігає у собі посилання на каталог магазину, кнопку повернення до головної сторінки, а також кошик, який реагує на кількість доданих товарів (рис 2.12)

Рис. 2.11 Header сайту

Рис. 2.12 Інтерактивний кошик

Нижче знаходиться рекламний банер, а також список усіх категорій каталогу, які можна побачити на рисунку 2.13.

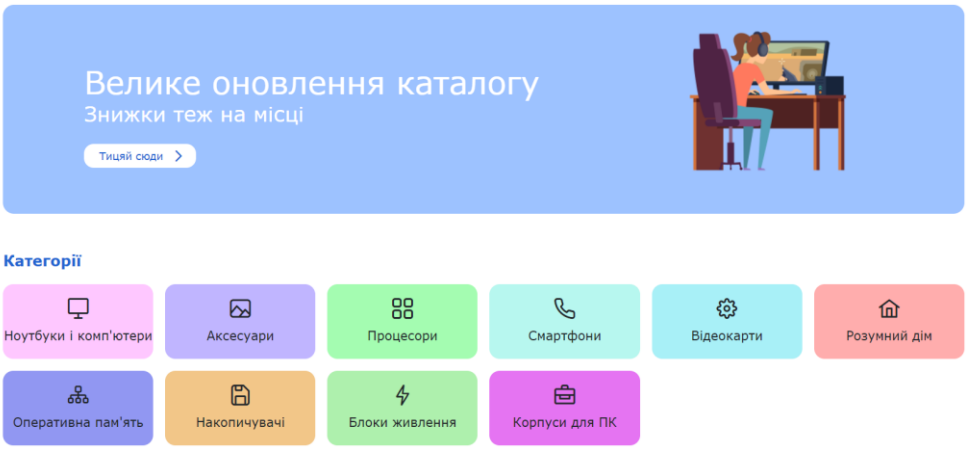


Рис. 2.13 Банер та категорії

Клік по категорії відкриває сторінки зі списком товару виключно з цієї категорії, що дає можливість зайвий раз не заходити до каталогу у пошуку бажаних товарів (рис. 2.14).

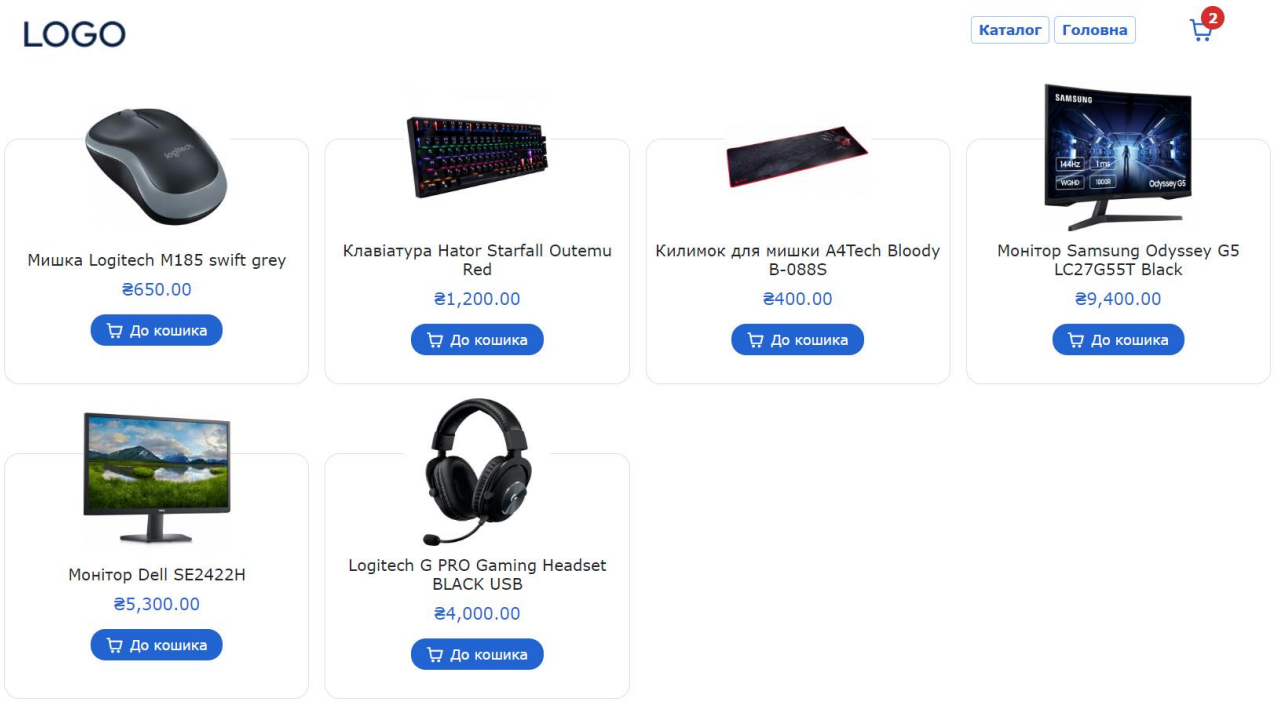


Рис. 2.14 Перехід до категорії «Аксесуари»

Ще нижче знаходяться обрані товари. Товари, що потраплять до цього компоненту, обираються через панель адміністраторів, де їм надається значення поля isFeatured (рис 2.15-16). Кнопки «До кошика» також працюють та додають товар у список бажаємого, оновлюючи цифру біля символу кошика.

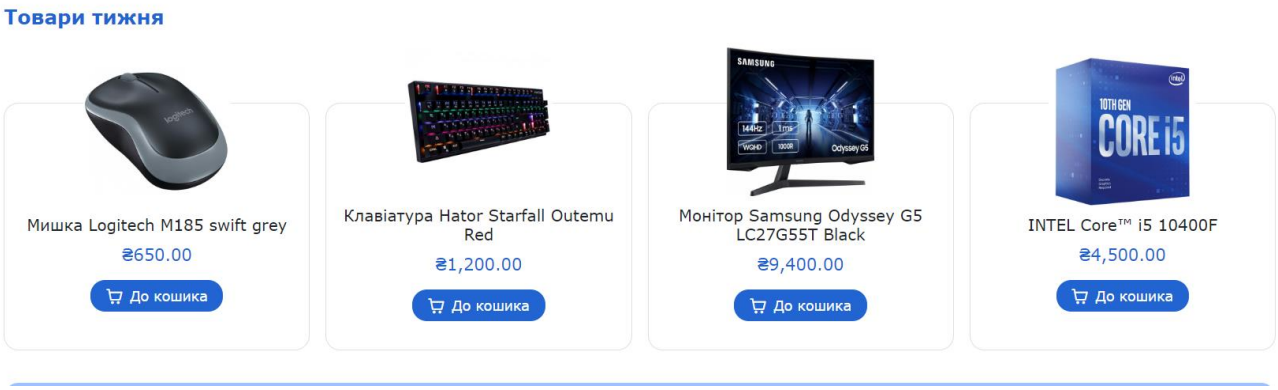
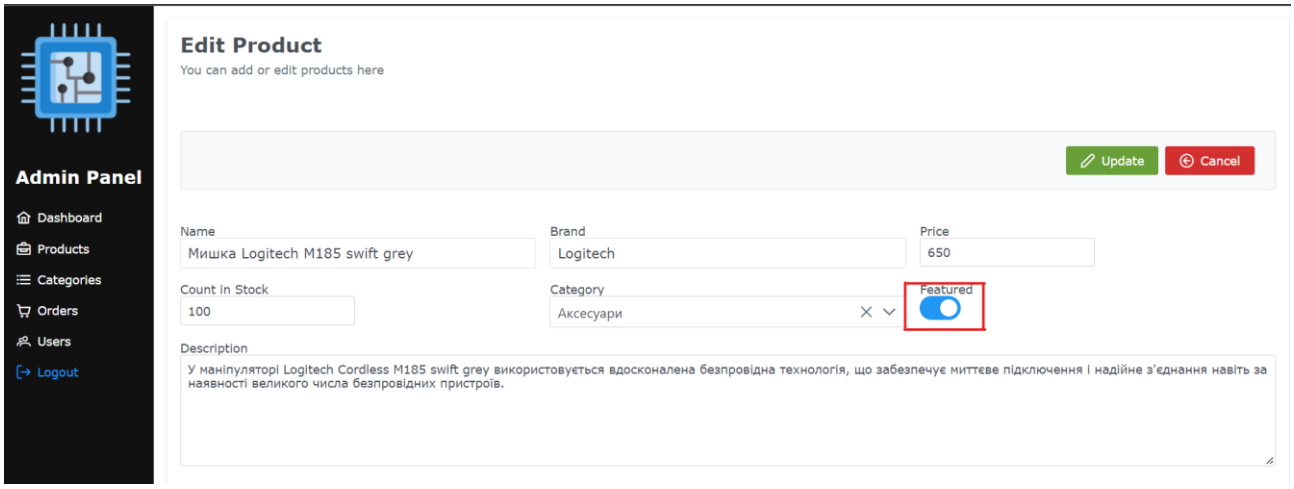


Рис. 2.15 Обрані товари



2.16 Адмін-панель та виділене поле Featured

Каталог представляє собою сітку всіх доданих до бази даних товарів з можливість сортирувати їх за категоріями. Побачити його одразу з сортуванням можна на рисунку 2.17

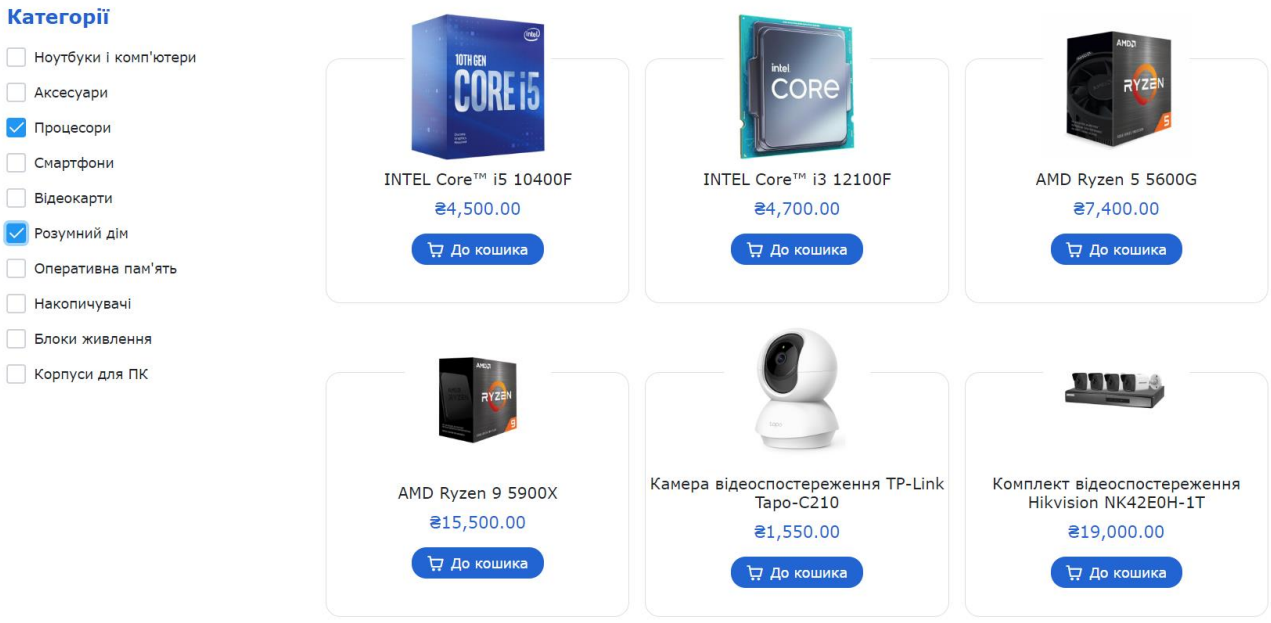


Рис. 2.17 Каталог товарів з сортуванням за категоріями

Сторінка товарів складається з детальної інформації про сам продукт – назвою, ціною, описом –, галереї його фотокарток, а також можливістю одразу додати його кошика, перед тим вказавши кількість потрібних позицій (рис. 2.18).

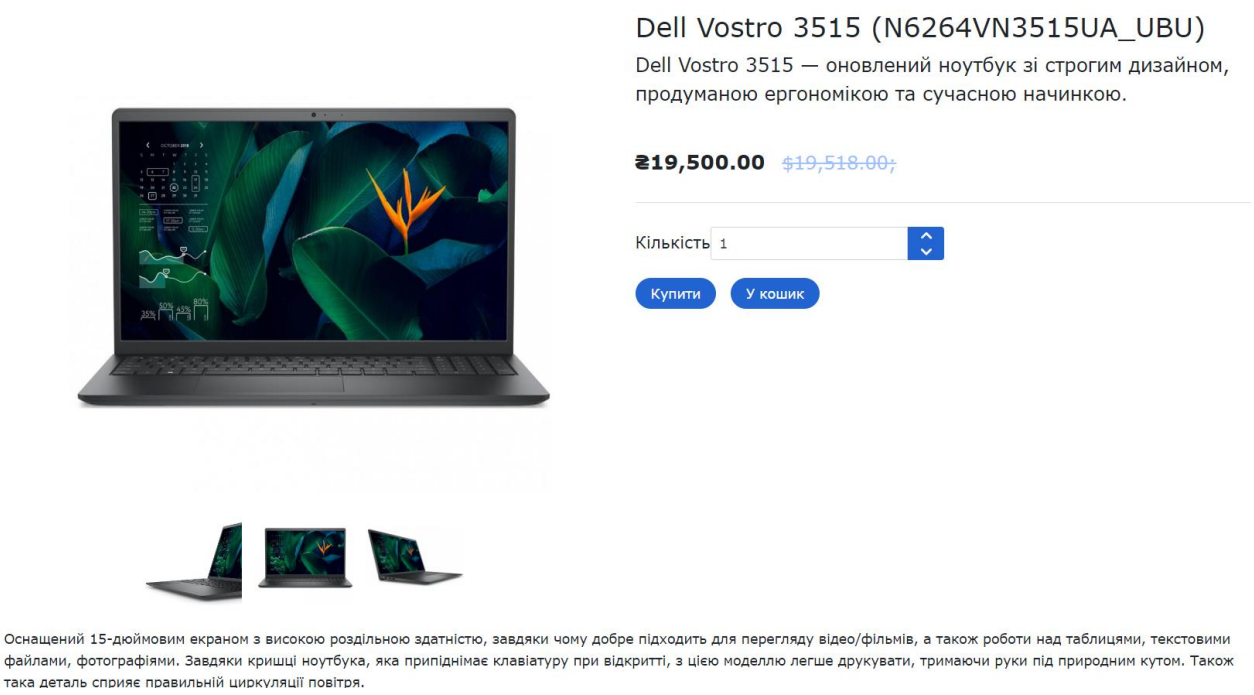


Рис. 2.18 Сторінка товару інтернет-магазину

Обравши потрібні позиції, користувач може переходити до кошика, де його чекає повний список бажаних товарів, їх кількість, загальна сума до сплати та кнопка повернення до каталогу. Доданий товар, за бажання, можна видалити з замовлення, бо навпаки збільшити його кількість – ціна буде змінюватися в залежності від будь-яких змін у самому кошикові. (Рис. 2.19).

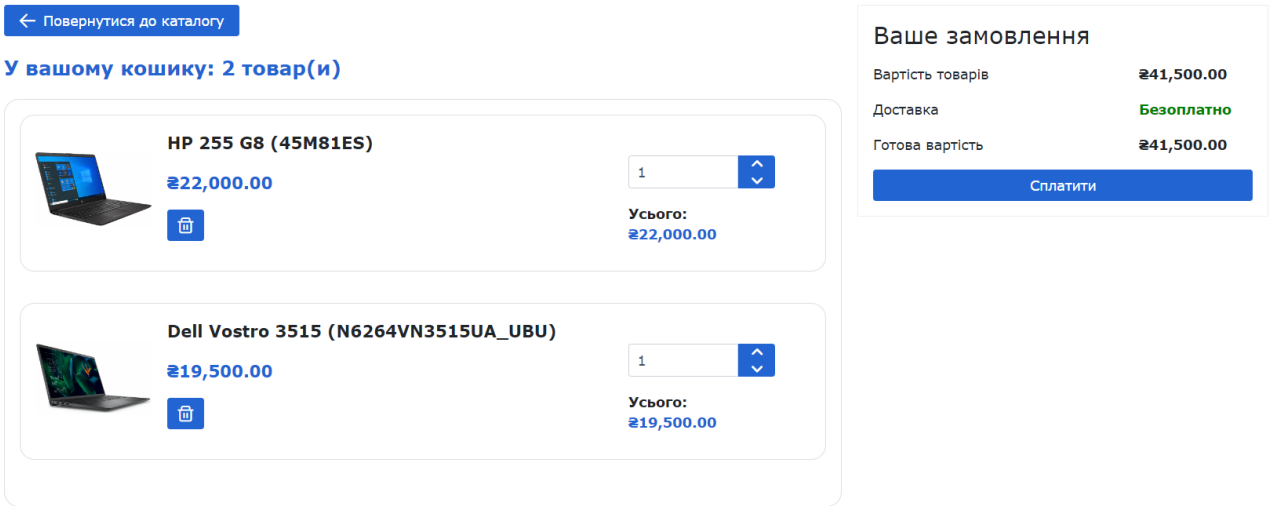


Рис. 2.19 Кошик з доданими до нього товарами

При спробі оформити замовлення інтернет-магазин видасть форму для входу в систему, аналогічну тій, що використовується у панелі адміністраторів (рис. 2.20).

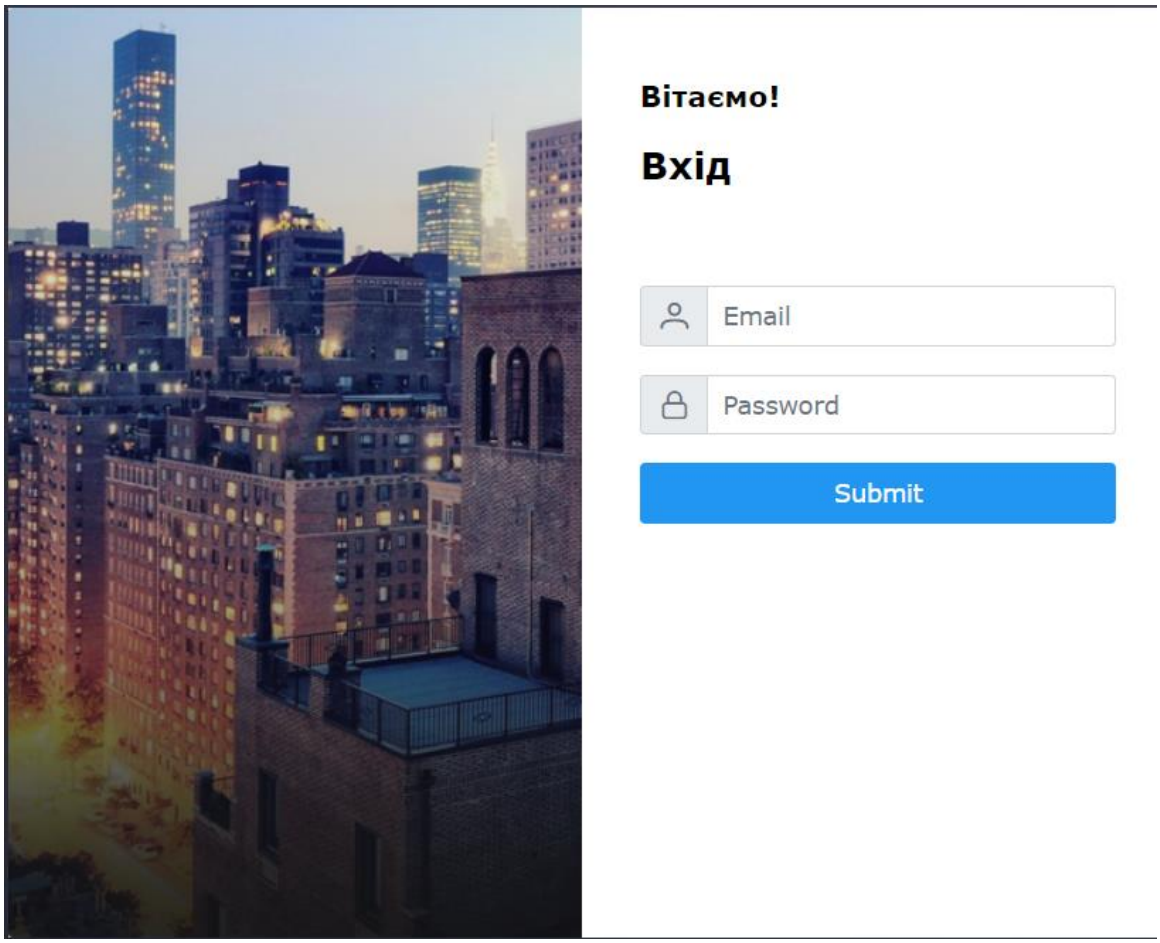


Рис. 2.20 Сторінка авторизації

При успішній авторизації відкриється вікно детального оформлення замовлення. Воно складатиметься з інформаційних полів, які має заповнити користувач, суми замовлення та кнопки переходу до сплати. Якщо в користувача вже є акаунт з заповненою про нього інформацією, вона автоматично підтягнеться та заповнить потрібні поля (рис. 2.21).

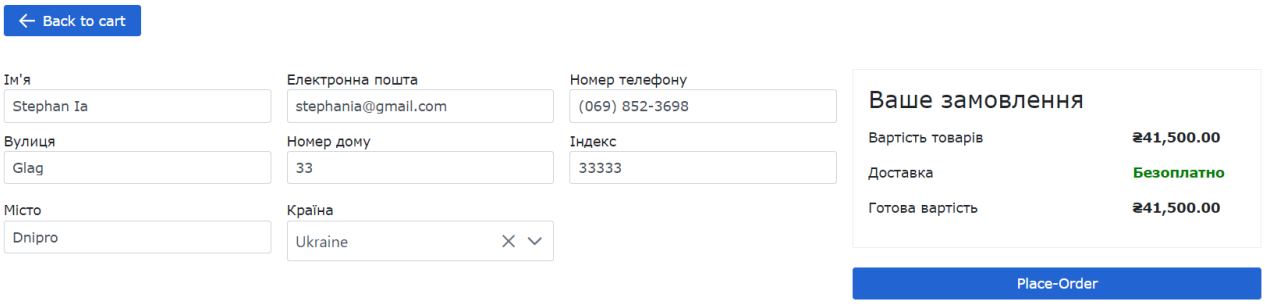


Рис. 2.21 Сторінка оформлення замовлення

Після оплати замовлення відкриється сторінка з подякою від керівників сайту (рис. 2.22), а саме замовлення відправиться до бази даних, звідки його можна регулювати через панель адміністраторів (рис. 2.23). Там доступна повна інформація: який користувач, о котрій годині, що саме та в якій кількості замовив. Крім того адміністратор у праві контролювати статус виконання замовлення та, за бажанням, взагалі його скасовувати.

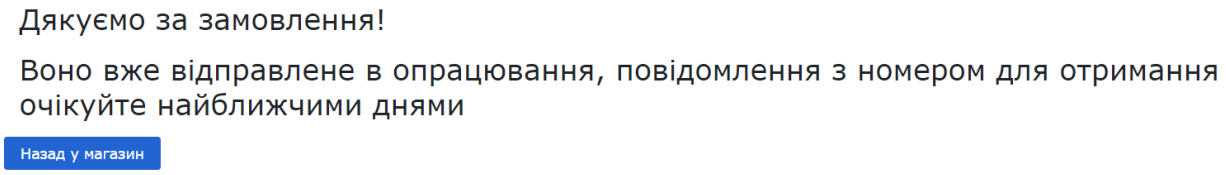


Рис. 2.22 Подяка за замовлення

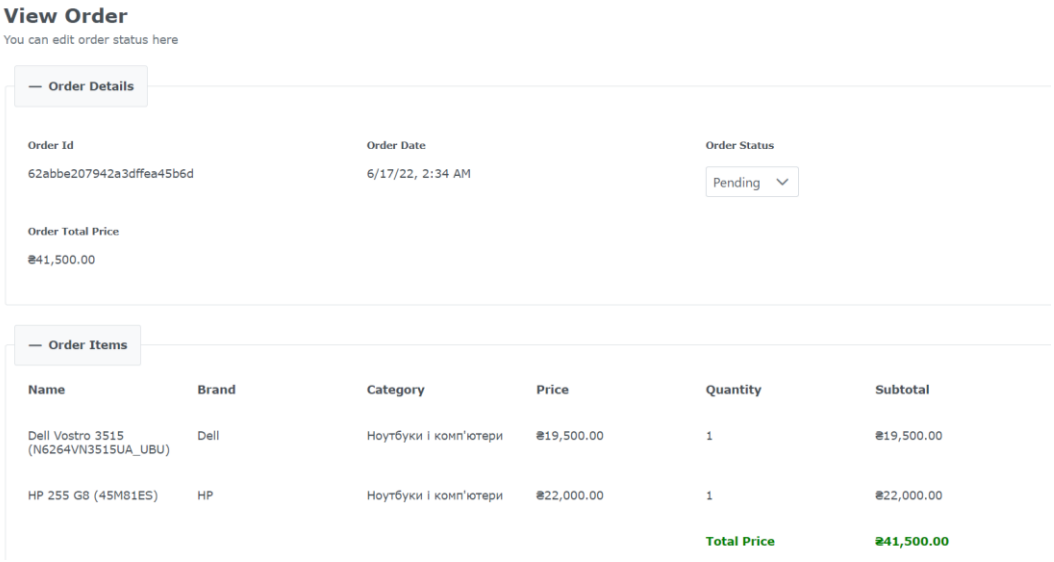


Рис. 2.23 Новостворене замовлення у панелі адміністраторів

РОЗДІЛ 3
ЕКОНОМІЧНІЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 2200;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт корекції програми в ході її розробки – 0,06;
4. годинна заробітна плата програміста – 147 грн/год;

Середня годинна зарплатня розробника була вирахувана використовуючи статистичні данні з сайту dou.ua[10]. Середня заробітна плата Junior Software Engineer розробника становить 700 доларів у місяць. Наразі курс долара за даними НБУ становить 29.25 гривень. Таким чином зарплата у гривнях для розробника у місяць становить 20475. Якщо для працівників встановлений 40-годинний робочий тиждень, як правило, це означає, що вони працюють 5 днів на тиждень по 8 годин на день. Відповідно, на місяць випадає від 21 до 23 трудових днів, якщо на будні дні не припадають свята. Тож у середньому візьмемо 22 робочих дні по 8 годин. З цього виходить, що розробник працює 176 годин на місяць у середньому. Тому це приблизно 116 грн/год.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,5;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1;
7. вартість машино-години ЕОМ – 15 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

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

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

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

де q – передбачуване число операторів (2200);

C – коефіцієнт складності програми (1,3);

p – коефіцієнт кореляції програми в ході її розробки (0,06).

$$Q = 2500 \cdot 1{,}4 \cdot (1 + 0{,}06) = 3031;$$

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

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

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,5);

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1);

$$t_u = \frac{3031 \cdot 1{,}5}{85 \cdot 1} = 53.6, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

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

$$t_a = \frac{3031}{20 \cdot 1} = 151.6, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

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

$$t_n = \frac{3031}{25 \cdot 1} = 121{,}2, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

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

$$t_{\text{отл}} = \frac{3031}{5 \cdot 1} = 606{,}2, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{\text{к}} = 1{,}5 \cdot t_{\text{отл}}; \quad (3.7)$$

$t_{\text{отл}}^{\text{к}} = 1{,}5 \cdot 606{,}2 = 909.3$, людино-годин,

Витрати праці на підготовку документації:

$$t_{д} = t_{дp} + t_{до}; \quad (3.8)$$

де $t_{дp}$ – трудомісткість підготовки матеріалів і рукопису;

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

$t_{д} = \frac{3031}{20 \cdot 1} = 151{,}6$, людино-годин,

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0{,}75 \cdot t_{др}; \quad (3.10)$$

$t_{до} = 0{,}75 \cdot 151{,}6 = 113{,}7$, людино-годин.

$t_{д} = 151.6 + 113{,}7 = 265{,}3$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 53.6+ 151.6 + 121,2 + 909.3 + 265,3 = 1551, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1551 людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

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

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

де t – загальна трудомісткість, людино-годин;

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

З урахуванням того, що середня годинна зарплата програміста становить 116 грн/год, то отримаємо:

$$З_{ЗП} = 1551 \cdot 116 = 179916, \text{ грн.}$$

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

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

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

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

$$З_{МВ} = 606{,}2 \cdot 15 = 9093 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 179916 + 9093 = 189009 \text{ грн.}$$

Очікуваний період створення ПЗ:

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

де B_k – число виконавців;

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

Витрати на створення програмного продукту:

$$T = \frac{1551}{2 \cdot 176} = 4.4 \text{ міс.}$$

Висновки. Front-end частина інтернет-магазину має вартість 189009 грн. Ймовірний очікуваний час розробки – 4.4 місяці при стандартному 40-годинному робочому тижні, 178-годинному робочому місяці і роботі двох працівників. Цей термін пов’язаний з кількістю операторів і включає в себе час для дослідження та розробку алгоритму розв’язання задачі, розробку дизайну і створення документації. Усього на розробку буде витрачено 1551 людино-годин.

ВИСНОВКИ

Задачею даної кваліфікаційної роботи була розробка frontend-частини багатофункціонального інтернет-магазину комп'ютерної техніки на базі сервісу Figma, технології Angular та мови програмування TypeScript.

Це програмне забезпечення створювалося з метою забезпечити оформлення та можливість керування повноцінним інтернет-магазином через адміністративну панель. Практичне значення програмного забезпечення полягає у створенні програмного додатку, що надає можливість майбутнім користувачам сервісу переглядати запропонований керівниками каталог товарів, додавати бажане до персонального кошика та оплачувати сформоване замовлення. Реалізоване це за допомогою зручного та зрозумілого інтерфейсу, який спрощує процес експлуатації сервісу.

Клієнт-серверна архітектура проєкту представлена у вигляді MEAN-стеку. Клієнтська частина додатку створювалася за допомогою мови програмування TypeScript на базі фреймворку Angular. База даних NodeJS функціонує за допомогою JSON-запитів. З сервером, створеним за допомогою NodeJS, комунікація відбувається через технологію Node Express.

Під час розробки кваліфікаційної роботи було:

– вивчено предметну галузь поставленого завдання;

– створено алгоритм реалізації;

– розроблено базу даних, сервер та клієнтську програму, які пов'язані за допомогою вищеназваного програмного забезпечення.

Створений сервіс виконує наступні дії:

– дозволяє вносити зміни у базу даних, додавати та видаляти інформацію;

– керувати каталогом, замовленнями та існуючими користувачами через панель адміністраторів;

– формувати web-сторінки на основі створеного шаблону, на яких має виводити інформацію, отриману з бази даних;

– оформлювати замовлення та відправляти інформацію про них до бази даних.

Актуальність представленої роботи обумовлюється високою потребою для будь-якого сучасного бізнесу у створенні повноцінної інтернет-сторінки, через яку можна буде вести продажі та підбивати статистичні підсумки роботи.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (1551 чол-год), підраховані витрати на створення програмного забезпечення (189009 грн.) і гаданий період розробки (4,4 міс.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дослідження ринку електронної комерції України URL: https://soulpartners.com.ua/news/tpost/x2dve03v71-rinok-elektronno-komerts-v-ukran-dosyagn (дата звернення: 02.05.2022).
2. The precipitous rise of Figma and fall of InVision URL: https://uxdesign.cc/the-precipitous-rise-of-figma-and-fall-of-invision-435f07e8d1b6 (дата звернення: 02.05.2022).
3. Удовик І.М., Коротенко Л.М., Шевцова О.С. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Програмне забезпечення”. – М : НТУ «Дніпровська політехніка», 2013. 16 с.
4. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
5. Вакансія Junior Software Engineer URL: https://www.work.ua/ru/jobs/2679106/ (дата звернення: 28.05.2022).
6. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).
7. Figma – що це? Огляд онлайн-сервісу URL: https://muraha.eu/2020/04/07/figma-chto-eto-obzor-onlajn-servisa-lajfhaki/ (дата звернення: 28.05.2022).
8. Документація фреймворку Angular URL: https://angular.io/docs (дата звернення: 10.05.2022).
9. Бібліотека коммпонентів для фреймворку Angular PrimeNG URL: https://www.primefaces.org/primeng/setup
10. Сучасний підручник JavaScript URL: https://learn.javascript.ru/ (дата звернення: 02.05.2022).
11. TypeScript – популярна мова програмування URL: https://www.typescriptlang.org (дата звернення: 02.05.2022).

12. Програмування по-українськи URL: programming.in.ua (дата звернення: 25.04.2022).

13. Прохоренок Н.А. HTML, JavaScript, PHP і MySQL. Джентльменський набір Webмайстра - СПб. : БХВ-Петербург, 2010. 912 с.

14. Чіртік А.В. Популярний самовчитель HTML. - СПб. : Пітер, 2012. 56 с.

15. Зандстра М. PHP: Объекты, шаблоны и методики программирования. -М.: Вильямс, 2015. 577 с.

16. Ташков П.А. Веб-мастеринг. HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка. -М. : Питер, 2010. 512 с.

17. Михайлов С., Кулі Л. Основи дизайну. - К .: Нове знання, 1999. 106 с.

18. Печников В.Н. Создание Web-сайтов без посторонней помощи. – М.: Триумф, 2006, 463 с.

19. Дронов В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Webсайтов / В.А.Дронов - СПб.: БХВ Петербург, 2011. - 416 с.: ил. - (Профессиональное программирование).

20. Десять лучших IDE и редакторов кода для веб-разработчиков [Електронний ресурс]/ URL: https://www.reg.ru/blog/10-luchshih-ide-iredaktorov-koda-dlya-veb-razrabotchikov/ (дата звернення: 06.05.2022).

21. Загрози інформаційної безпеки [Електронний ресурс] / URL: https://uk.wikipedia.org/wiki/Загрози_інформаційної_безпеки (дата звернення: 06.05.2021).

22. Статичні та динамічні web-сайти/ URL: https://armedsoft.com/ua/blog/statychni-ta-dynamichni-web-sayty (дата звернення: 11.05.2022).

23. Botha, J., Bothma, C., Geldenhuys, P. Managing E-commerce in Business. Cape Town: Juta and Company Ltd. – 2005. – p. 3.

24. Todd Fredrich. REST API Tutorial. – 2012. – p. 13-15.

25. JavaScript. Изучение веб-разработки. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/ (дата звернення: 25.05.2022).

26. Никсон Робин. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е изд. : Издательство 25 лет Питер, СанктПетербург, 2016. С. 331 – 668.

Додаток А

ЛІСТИНГ ПРОГРАМИ

app.component.html //розмітка усіх сторінок сайту

```
<div class="container">
    <techno-header></techno-header>
    <router-outlet></router-outlet>
    <techno-footer></techno-footer>
</div>
```

app.component.ts //логіка компоненту app

```
import { Component, OnInit } from '@angular/core';
import { UsersService } from '@my-team/users';

@Component({
    selector: 'my-team-root',
    templateUrl: './app.component.html'
})
export class AppComponent implements OnInit {
    constructor(private usersService: UsersService) {}

    ngOnInit() {
        this.usersService.initAppSession();
    }
    title = 'techno-base';
}
```

app.module.ts //імпорт модулів

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { RouterModule, Routes } from '@angular/router';

import { AppComponent } from './app.component';
import { HomePageComponent } from './pages/home-page/home-page.component';
import { HeaderComponent } from './shared/header/header.component';
import { FooterComponent } from './shared/footer/footer.component';
import { ProductsModule } from './../../../../libs/products/src/lib/products.module';

import { UiModule } from './../../../../libs/ui/src/lib/ui.module';
import { AccordionModule } from 'primeng/accordion';
import { NavComponent } from './shared/nav/nav.component';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { OrdersModule } from '@my-team/orders';
import { CommonModule } from '@angular/common';
import { JwtInterceptor, UsersModule } from '@my-team/users';
import { StoreModule } from '@ngrx/store';
import { EffectsModule } from '@ngrx/effects';
import { MessageService } from 'primeng/api';

const routes: Routes = [
    {
        path: 'main',
```

```
        component: HomePageComponent
    }
];

@NgModule({
    declarations: [AppComponent, HomePageComponent, HeaderComponent, FooterComponent,
NavComponent],
    imports: [
        BrowserModule,
        BrowserAnimationsModule,
        RouterModule.forRoot(routes),
        HttpClientModule,
        StoreModule.forRoot({}),
        EffectsModule.forRoot([]),
        ProductsModule,
        AccordionModule,
        UiModule,
        OrdersModule,
        CommonModule,
        UsersModule
    ],
    providers: [MessageService, { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor,
multi: true }],
    bootstrap: [AppComponent]
})
export class AppModule {}
```

home-page.component.html //розмітка головної сторінки

```
<ui-banner></ui-banner>
<products-categories-banner></products-categories-banner>
<products-featured-products></products-featured-products>
```

categories-banner.component.html //розмітка банеру категорій

```
<div class="categories-banner">
    <h5>Категорії</h5>
    <div class="grid">
        <div class="col-2" *ngFor="let category of categories">
            <div class="category" [routerLink]="'/category/' + category.id" [ngStyle]="{
backgroundColor: category.color }">
                <div class="category-icon">
                    <i class="pi pi-{{ category.icon }}"></i>
                </div>
                <div class="category-name">{{ category.name }}</div>
            </div>
        </div>
    </div>
</div>
```

categories-banner.component.ts //логіка компоненту categories-banner

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Subject, takeUntil } from 'rxjs';
import { Category } from '../../models/category';
import { CategoriesService } from '../../services/categories.service';

@Component({
    selector: 'products-categories-banner',
    templateUrl: './categories-banner.component.html',
    styles: []
})
```

```
export class CategoriesBannerComponent implements OnInit, OnDestroy {
    categories: Category[] = [];
    endSubs$: Subject<any> = new Subject();

    constructor(private categoriesService: CategoriesService) {}

    ngOnInit(): void {
        this.categoriesService
            .getCategories()
            .pipe(takeUntil(this.endSubs$))
            .subscribe((categories) => {
                this.categories = categories;
            });
    }

    ngOnDestroy() {
        this.endSubs$.next(void 0);
        this.endSubs$.complete();
    }
}
```

categories-banner.scss //стилізація компоненту categories-banner

```
.categories-banner {
    margin-top: 50px;

    h5 {
        margin-bottom: 20px;
        font-weight: bold;
        color: $primary-color;
    }

    .category {
        min-height: 100px;
        border-radius: 14px;
        text-align: center;
        display: flex;
        background-color: aqua;
        justify-content: center;
        align-items: center;
        flex-direction: column;
        cursor: pointer;

        &-icon {
            .pi {
                font-size: 2em;
                margin-bottom: 10px;
            }
        }

        &-name {
            font-size: 1.2em;
        }
    }
}
```

featured-products.component.html //розмітка вибраних товарів

```
<div class="featured-products">
    <h5>Товари тижня</h5>
    <div class="grid">
        <div class="col-3" *ngFor="let product of featuredProducts">
```

```
            <products-product-item [product]="product"></products-product-item>
        </div>
    </div>
</div>
```

featured-products.component.ts // логіка компоненту featured-products

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Subject, takeUntil } from 'rxjs';
import { Product } from '../../models/product';
import { ProductsService } from '../../services/products.service';

@Component({
    selector: 'products-featured-products',
    templateUrl: './featured-products.component.html',
    styles: []
})
export class FeaturedProductsComponent implements OnInit, OnDestroy {
    featuredProducts: Product[] = [];
    endSubs$: Subject<any> = new Subject();
    constructor(private prodService: ProductsService) {}

    ngOnInit(): void {
        this._getFeaturedProducts();
    }

    ngOnDestroy(): void {
        this.endSubs$.next(void 0);
        this.endSubs$.complete();
    }

    private _getFeaturedProducts() {
        this.prodService
            .getFeaturedProducts(4)
            .pipe(takeUntil(this.endSubs$))
            .subscribe((products) => {
                this.featuredProducts = products;
            });
    }
}
```

Product-list.component.html //розмітка каталогу товарів

```
<div class="products-page">
    <div class="grid">
        <div class="col-3" *ngIf="!isCategoryPage">
            <h5>Категорії</h5>
            <div class="field-checkbox" *ngFor="let category of categories">
                <p-checkbox        [(ngModel)]="category.checked"        [binary]="true"
[inputId]="category.id!" (onChange)="categoryFilter()"></p-checkbox>
                <label for="{{ category.id }}">{{ category.name }}</label>
            </div>
        </div>
        <div [ngClass]="{ 'col-9': !isCategoryPage, 'col-12': isCategoryPage }">
            <div class="grid" *ngIf="products">
                <div [ngClass]="{ 'col-4': !isCategoryPage, 'col-3': isCategoryPage }"
*ngFor="let product of products">
```

```
                    <products-product-item [product]="product"></products-product-item>
                </div>
            </div>
        </div>
    </div>
</div>
```

Product-list.component.ts // логіка компоненту Product-list

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Category } from '../../models/category';
import { Product } from '../../models/product';
import { CategoriesService } from '../../services/categories.service';
import { ProductsService } from '../../services/products.service';

@Component({
    selector: 'products-list',
    templateUrl: './products-list.component.html',
    styles: []
})
export class ProductsListComponent implements OnInit {
    products: Product[] = [];
    categories: Category[] = [];
    isCategoryPage: boolean | undefined;

    constructor(private prodService: ProductsService, private catService:
CategoriesService, private route: ActivatedRoute) {}

    ngOnInit(): void {
        this.route.params.subscribe((params) => {
            params['categoryid'] ? this._getProducts([params['categoryid']]) :
this._getProducts();
            params['categoryid'] ? (this.isCategoryPage = true) : (this.isCategoryPage =
false);
        });
        this._getCategories();
    }

    private _getProducts(categoriesFilter?: string[]) {
        this.prodService.getProducts(categoriesFilter).subscribe((resProducts) => {
            this.products = resProducts;
        });
    }

    private _getCategories() {
        this.catService.getCategories().subscribe((resCats) => {
            this.categories = resCats;
        });
    }

    categoryFilter() {
        const selectedCategories = this.categories
            .filter(
                (
                    c
                ): c is typeof c & {
                    checked: true;
                    id: string;
                } => Boolean(c.checked && c.id)
            )
```

```
            .map(({ id }) => id);
        this._getProducts(selectedCategories);
    }
}
```

Product-item.component.html //розмітка форми товару

```
<div class="product-item" *ngIf="product">
    <div class="product-item-wrapper">
        <img    [routerLink]="'/products/'    +    product.id"    [src]="product.image"
class="image" alt="" />
        <h6 class="name">{{ product.name }}</h6>
        <h6 class="price">{{ product.price | currency: 'UAH':'symbol-narrow' }}</h6>
        <p-button styleClass="mt-3 add-to-cart" label="До кошика" icon="pi pi-shopping-
cart" (onClick)="addProductToCart()"></p-button>
    </div>
</div>
```

Product-item.component.ts // логіка компоненту product-item

```
import { Component, Input, OnInit } from '@angular/core';
import { CartService, CartItem } from '@my-team/orders';
import { Product } from '../../models/product';

@Component({
    selector: 'products-product-item',
    templateUrl: './product-item.component.html',
    styles: []
})
export class ProductItemComponent implements OnInit {
    @Input()
    product!: Product;

    constructor(private cartService: CartService) {}

    ngOnInit(): void {}

    addProductToCart() {
        const cartItem: CartItem = {
            productId: this.product.id,
            quantity: 1
        };
        this.cartService.setCartItem(cartItem);
    }
}
```

Product-page.component.html //розмітка сторінки товару

```
<div class="product-page" *ngIf="product">
    <div class="grid">
        <div class="col-6">
            <ui-gallery [images]="product.images!"></ui-gallery>
        </div>
        <div class="col-6 product">
            <h4 class="product-name">{{ product.name }}</h4>
            <p class="product-desc">
                {{ product.description }}
            </p>
            <div class="product-price">
                {{ product.price | currency: 'UAH':'symbol-narrow' }}
                <span class="price-before"> {{ product.price! + 18 | currency }}; </span>
```

```
            </div>

            <div class="product-quantity">
                <div class="field col-12 p-md-3 p-0">
                    <label for="quantity">Кількість</label>
                    <p-inputNumber         [(ngModel)]="quantity"         mode="decimal"
[showButtons]="true" inputId="quantity" [min]="1" [max]="100"> </p-inputNumber>
                </div>
            </div>

            <div class="product-action">
                <button pButton pRipple type="button" label="Купити" class="p-button-
rounded p-mr-2"></button>
                <button pButton pRipple type="button" label="У кошик" class="p-button-
rounded" (click)="addProductToCart()"></button>
            </div>
        </div>
    </div>
    <div class="grid product-rich-desc">
        <div class="col-12">
            <div [innerHTML]="product.richDescription"></div>
        </div>
    </div>
</div>
```

Product-page.component.ts // логіка компоненту product-page

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { CartItem, CartService } from '@my-team/orders';
import { Subject, takeUntil } from 'rxjs';
import { Product } from '../../models/product';
import { ProductsService } from '../../services/products.service';

@Component({
    selector: 'products-product-page',
    templateUrl: './product-page.component.html',
    styles: []
})
export class ProductPageComponent implements OnInit, OnDestroy {
    product!: Product;
    endSubs$: Subject<any> = new Subject();
    quantity: number = 1;

    constructor(private prodService: ProductsService, private route: ActivatedRoute,
private cartService: CartService) {}

    ngOnInit(): void {
        this.route.params.subscribe((params) => {
            if (params['productid']) {
                this._getProduct(params['productid']);
            }
        });
    }

    ngOnDestroy(): void {
        this.endSubs$.next(void 0);
        this.endSubs$.complete();
    }

    addProductToCart() {
        const cartItem: CartItem = {
            productId: this.product.id,
```

```
            quantity: this.quantity
        };

        this.cartService.setCartItem(cartItem);
    }

    private _getProduct(id: string) {
        this.prodService
            .getProduct(id)
            .pipe(takeUntil(this.endSubs$))
            .subscribe((resProduct) => {
                this.product = resProduct;
            });
    }
}
```

products.module.ts //імпорт модулів та налаштування шляхів для товарів

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ProductsSearchComponent } from './components/products-search/products-search.component';
import { CategoriesBannerComponent } from './components/categories-banner/categories-banner.component';
import { RouterModule, Routes } from '@angular/router';
import { ProductItemComponent } from './components/product-item/product-item.component';
import { FeaturedProductsComponent } from './components/featured-products/featured-products.component';
import { ButtonModule } from 'primeng/button';
import { CheckboxModule } from 'primeng/checkbox';
import { RatingModule } from 'primeng/rating';
import { ProductsListComponent } from './pages/products-list/products-list.component';
import { FormsModule } from '@angular/forms';
import { ProductPageComponent } from './pages/product-page/product-page.component';
import { InputNumberModule } from 'primeng/inputnumber';
import { UiModule } from '@my-team/ui';

const routes: Routes = [
    {
        path: 'products',
        component: ProductsListComponent
    },
    {
        path: 'category/:categoryid',
        component: ProductsListComponent
    },
    {
        path: 'products/:productid',
        component: ProductPageComponent
    }
];

@NgModule({
    imports: [CommonModule, RouterModule.forChild(routes), ButtonModule, CheckboxModule, RatingModule, FormsModule, InputNumberModule, UiModule],
    declarations: [
        ProductsSearchComponent,
        CategoriesBannerComponent,
        ProductItemComponent,
        FeaturedProductsComponent,
        ProductsListComponent,
        ProductPageComponent
```

```
    ],
    exports: [ProductsSearchComponent, CategoriesBannerComponent, ProductItemComponent,
FeaturedProductsComponent, ProductsListComponent, ProductPageComponent]
})
export class ProductsModule {}
```

Cart-page.component.html //розмітка кошику

```
<div class="cart-page">
    <div class="grid">
        <div class="col-8">
            <div>
                <p-button label="Повернутися до каталогу" icon="pi pi-arrow-left"
(onClick)="backToShop()"></p-button>
            </div>
            <div class="cart-prods">
                <h5>У вашому кошику: {{ cartCount }} товар(и)</h5>
            </div>
            <div class="cart-item">
                <div class="cart-item mb-5" *ngFor="let cartItem of cartItemsDetailed">
                    <div class="grid p-fluid">
                        <div class="col-2 cart-item-image">
                            <img                          [src]="cartItem.product.image"
[attr.alt]="cartItem.product.image" />
                        </div>
                        <div class="col-7">
                            <div     class="cart-item-name">{{     cartItem.product.name
}}</div>
                            <div class="cart-item-price">
                                {{  cartItem.product.price  |  currency:  'UAH':'symbol-
narrow' }}
                            </div>
                            <div class="card-item-remove">
                                <p-button               icon="pi               pi-trash"
(onClick)="deleteCartItem(cartItem)"></p-button>
                            </div>
                        </div>
                        <div class="col-3">
                            <div class="product-quantity">
                                <div class="p-field cart-item-quantity">
                                    <p-inputNumber
                                        mode="decimal"
                                        [showButtons]="true"
                                        inputId="cartItem.product.id"
                                        [min]="1"
                                        [max]="100"
                                        [(ngModel)]="cartItem.quantity"
                                        (onInput)="updateCartItemQuantity($event,
cartItem)"
                                    >
                                    </p-inputNumber>
                                </div>
                                <div class="cart-item-subtotal">
                                    Усього:
                                    <span class="cart-item-subtotal-value">{{
                                        cartItem.product.price!  *  cartItem.quantity!  |
currency: 'UAH':'symbol-narrow'
                                    }}</span>
                                </div>
                            </div>
                        </div>
                    </div>
```

```
                </div>
            </div>
        </div>
        <div class="col-4">
            <orders-order-summary></orders-order-summary>
        </div>
    </div>
</div>
```

Cart-page.component.ts // логіка компоненту cart-page

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Subject } from 'rxjs';
import { takeUntil } from 'rxjs/operators';
import CartItemDetailed from '../../models/cart';
import { CartService } from '../../services/cart.service';
import { OrdersService } from '../../services/orders.service';

@Component({
    selector: 'orders-cart-page',
    templateUrl: './cart-page.component.html',
    styles: []
})
export class CartPageComponent implements OnInit, OnDestroy {
    cartItemsDetailed: CartItemDetailed[] = [];
    cartCount = 0;
    endSubs$: Subject<any> = new Subject();

    constructor(private router: Router, private cartService: CartService, private
ordersService: OrdersService) {}

    ngOnInit(): void {
        this._getCartDetails();
    }

    ngOnDestroy() {
        this.endSubs$.next(void 0);
        this.endSubs$.complete();
    }

    private _getCartDetails() {
        this.cartService.cart$.pipe(takeUntil(this.endSubs$)).subscribe((respCart) => {
            this.cartItemsDetailed = [];
            this.cartCount = respCart?.items?.length ?? 0;
            respCart.items?.forEach((cartItem) => {

this.ordersService.getProduct(cartItem.productId!).subscribe((respProduct) => {
                    this.cartItemsDetailed.push({
                        product: respProduct,
                        quantity: cartItem.quantity
                    });
                });
            });
        });
    }

    backToShop() {
        this.router.navigate(['/products']);
    }

    deleteCartItem(cartItem: CartItemDetailed) {
```

```
        this.cartService.deleteCartItem(cartItem.product.id);
    }

    updateCartItemQuantity(event: any, cartItem: CartItemDetailed) {
        this.cartService.setCartItem(
            {
                productId: cartItem.product.id,
                quantity: event.value
            },
            true
        );
    }
}
```

Checkout-page.component.html //розмітка оформлення замовлення

```
<div class="checkout-page">
    <div>
        <p-button      label="Back      to      cart"      icon="pi      pi-arrow-left"
(onClick)="backToCart()"></p-button>
    </div>
    <div class="grid checkout-form">
        <div class="col-8">
            <form [formGroup]="checkoutFormGroup">
                <div class="p-fluid p-formgrid grid">
                    <div class="p-field col-4">
                        <label for="name">Ім'я</label>
                        <input  formControlName="name"  id="name"  type="text"  pInputText
/>
                        <small    *ngIf="checkoutForm.name.invalid    &&    isSubmitted"
class="p-error">Ім'я є обов'язкове</small>
                    </div>
                    <div class="p-field col-4">
                        <label for="email">Електронна пошта</label>
                        <input formControlName="email" id="email" type="text" pInputText
/>
                        <small    *ngIf="checkoutForm.email.invalid    &&    isSubmitted"
class="p-error"
                            ><span  *ngIf="checkoutForm.email.errors!.required">Пошта  є
обов'язкова</span>
                            <span   *ngIf="checkoutForm.email.errors!.email">Неправильна
пошта</span></small
                        >
                    </div>
                    <div class="p-field col-4">
                        <label for="color">Номер телефону</label><br />
                        <p-inputMask   mask="(999)   999-9999"   formControlName="phone"
placeholder="(999) 999-9999"></p-inputMask>
                        <small    *ngIf="checkoutForm.phone.invalid    &&    isSubmitted"
class="p-error">Телефон є обов'язковим</small>
                    </div>
                </div>
                <div class="p-fluid p-formgrid grid">
                    <div class="p-field col-4">
                        <label for="street">Вулиця</label>
                        <input    formControlName="street"    id="street"    type="text"
pInputText />
                        <small   *ngIf="checkoutForm.street.invalid   &&   isSubmitted"
class="p-error">Вулиця є обов'язковою</small>
                    </div>
                    <div class="p-field col-4">
                        <label for="street">Номер дому</label>
```

```
                        <input  formControlName="apartment"  id="apartment"  type="text"
pInputText />
                        <small  *ngIf="checkoutForm.apartment.invalid  &&  isSubmitted"
class="p-error">Номер дому є обов'язковим</small>
                    </div>
                    <div class="p-field col-4">
                        <label for="street">Індекс</label>
                        <input formControlName="zip" id="zip" type="text" pInputText />
                        <small *ngIf="checkoutForm.zip.invalid && isSubmitted" class="p-
error">Індекс є обов'язковим</small>
                    </div>
                    <div class="p-field col-4">
                        <label for="city">Місто</label>
                        <input  formControlName="city"  id="city"  type="text"  pInputText
/>
                        <small    *ngIf="checkoutForm.city.invalid    &&    isSubmitted"
class="p-error">Місто є обов'язковим</small>
                    </div>
                    <div class="p-field col-4">
                        <label for="country">Країна</label><br />
                        <p-dropdown
                            [options]="countries"
                            formControlName="country"
                            optionLabel="name"
                            optionValue="id"
                            [filter]="true"
                            filterBy="name"
                            [showClear]="true"
                            placeholder="Select a Country"
                        ></p-dropdown>
                        <small   *ngIf="checkoutForm.country.invalid   &&   isSubmitted"
class="p-error">Країна є обов'язковою</small>
                    </div>
                </div>
            </form>
        </div>
        <div class="col-4">
            <orders-order-summary></orders-order-summary>
            <div class="checkout-button">
                <p-button label="Place-Order" (onClick)="placeOrder()"></p-button>
            </div>
        </div>
    </div>
</div>
```

Checkout-page.component.ts // логіка компоненту Checkout-page

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { UsersService } from '@my-team/users';
import { Cart } from '../../models/cart';
import { Order } from '../../models/order';
import { OrderItem } from '../../models/order-item';
import { CartService } from '../../services/cart.service';
import { OrdersService } from '../../services/orders.service';
import { ORDER_STATUS } from '../../order.constants';
import { Subject, take, takeUntil } from 'rxjs';

@Component({
    selector: 'orders-checkout-page',
```

```
    templateUrl: './checkout-page.component.html'
})
export class CheckoutPageComponent implements OnInit, OnDestroy {
    constructor(
        private router: Router,
        private usersService: UsersService,
        private formBuilder: FormBuilder,
        private cartService: CartService,
        private ordersService: OrdersService
    ) {}
    checkoutFormGroup: FormGroup;
    isSubmitted = false;
    orderItems: OrderItem[] = [];
    userId: any;
    countries: any[];
    unsubscribe$: Subject<any> = new Subject();

    ngOnInit(): void {
        this._initCheckoutForm();
        this._autoFillUserData();
        this._getCartItems();
        this._getCountries();
    }

    ngOnDestroy() {
        this.unsubscribe$.next(void 0);
        this.unsubscribe$.complete();
    }

    private _initCheckoutForm() {
        this.checkoutFormGroup = this.formBuilder.group({
            name: ['', Validators.required],
            email: ['', [Validators.email, Validators.required]],
            phone: ['', Validators.required],
            city: ['', Validators.required],
            country: ['', Validators.required],
            zip: ['', Validators.required],
            apartment: ['', Validators.required],
            street: ['', Validators.required]
        });
    }

    private _autoFillUserData() {
        this.usersService
            .observeCurrentUser()
            .pipe(takeUntil(this.unsubscribe$))
            .subscribe((user) => {
                if (user) {
                    this.userId = user.id;
                    this.checkoutForm.name.setValue(user.name);
                    this.checkoutForm.email.setValue(user.email);
                    this.checkoutForm.phone.setValue(user.phone);
                    this.checkoutForm.city.setValue(user.city);
                    this.checkoutForm.street.setValue(user.street);
                    this.checkoutForm.country.setValue(user.country);
                    this.checkoutForm.zip.setValue(user.zip);
                    this.checkoutForm.apartment.setValue(user.apartment);
                }
            });
    }

    private _getCartItems() {
```

```
        const cart: Cart = this.cartService.getCart();
        this.orderItems = cart.items!.map((item) => {
            return {
                product: item.productId,
                quantity: item.quantity
            };
        });
    }

    private _getCountries() {
        this.countries = this.usersService.getCountries();
    }

    backToCart() {
        this.router.navigate(['/cart']);
    }

    placeOrder() {
        this.isSubmitted = true;
        if (this.checkoutFormGroup.invalid) {
            return;
        }

        const order: Order = {
            orderItems: this.orderItems,
            shippingAddress1: this.checkoutForm.street.value,
            shippingAddress2: this.checkoutForm.apartment.value,
            city: this.checkoutForm.city.value,
            zip: this.checkoutForm.zip.value,
            country: this.checkoutForm.country.value,
            phone: this.checkoutForm.phone.value,
            status: 0,
            user: this.userId,
            dateOrdered: `${Date.now()}`
        };

        this.ordersService.createOrder(order).subscribe(
            () => {
                //redirect to thank you page // payment
                this.cartService.emptyCart();
                this.router.navigate(['/success']);
            },
            () => {
                //display some message to user
            }
        );
    }

    get checkoutForm() {
        return this.checkoutFormGroup.controls;
    }
}
```

Додаток Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему:

«Розробка Інтернет магазину комп’ютерної техніки на базі технології Figma»

студента групи 121-18-2 Лопатін Богдан Михайлович

Додаток В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім'я файлу	**Опис**
Пояснювальні документи	
Диплом_Лопатін.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Лопатін.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Лопатін.ppt	Презентація кваліфікаційної роботи