ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Вебсайт інтернет-магазину товарів для продажу програмного забезпечення для бізнесу з використанням багатокритеріальних фільтрів та рекомендаційної системи»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп’ютерні науки»
Освітня програма «Комп’ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Микола АРТЕМЕНКО

Керівник: викладач кафедри інформаційного менеджменту, математики та статистики
Олег МУШИНСЬКИЙ

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп'ютерних наук
_____Сергій МІЧКІВСЬКИЙ
«____»____20___р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Артеменко Микола Володимирович

Тема роботи	Вебсайт інтернет-магазину товарів для продажу програмного забезпечення для бізнесу з використанням багатокритеріальних фільтрів та рекомендаційної системи
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Розробити вебсайт інтернет-магазину, що забезпечує продаж програмного забезпечення для бізнесу, із впровадженням багатокритеріальних фільтрів та рекомендаційної системи.
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Кривов'язюк І. В. Управлінська інноватика забезпечення досконалості бізнесу в умовах інформаційно-комунікаційної технологізації : монографія. – Луцьк : ФОП Мажула Ю. М., 2022. 172 с. 2. Hasler, D., Krumay, B., & Schallmo, D. (2022). Characteristics of digital platforms from a B2B perspective–A systematic literature review.
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має містити теоретичне, системотехнічне або експериментальне дослідження за темою роботи, яку слід розглядати як складне спеціалізоване завдання або практичну проблему в галузі комп'ютерних наук, яка характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2024 р.

Керівник Олег МУШИНСЬКИЙ

Здобувач освітнього ступеня бакалавра Микола АРТЕМЕНКО

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
Основний етап			
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
Завершальний етап			
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Олег МУШИНСЬКИЙ

Здобувач освітнього ступеня бакалавра Микола АРТЕМЕНКО

Артеменко М.В. Вебсайт інтернет-магазину товарів для продажу програмного забезпечення для бізнесу з використанням багатокритеріальних фільтрів та рекомендаційної системи.

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2025.

Ключові слова: інтернет-магазин, програмне забезпечення для бізнесу, фільтрація товарів, рекомендаційна система, персоналізовані рекомендації, автоматизація бізнес-процесів, малий та середній бізнес, пошук програмного забезпечення, порівняння продуктів, постачальники програмного забезпечення.

Рис. 15. Бібліограф.: 11 найм.

Artemenko M.V. Website for an online store selling business software using multi-criteria filters and a recommendation system.

Project explanatory note by specialty 122 – Computer science. – «KROK» University, Educational and Scientific Institute of information and communication technologies, Department of Computer Science, Kyiv, 2025.

Keywords: online store, business software, product filtering, recommendation system, personalized recommendations, business process automation, small and medium business, software search, product comparison, software providers

Fig. 15. Bibliography: 11 Items.

ЗМІСТ

ВСТУП

Актуальність теми. В умовах цифрової трансформації бізнесу, компанії все більше покладаються на програмне забезпечення для оптимізації своїх внутрішніх процесів, зниження витрат та підвищення ефективності. Діджиталізація бізнесу вже зараз набуває масового характеру, а через кілька років цифрова трансформація перетвориться на життєву необхідність для переважної більшості бізнес-проектів. Однак процес вибору і придбання відповідних програмних рішень для бізнесу є складним і часто займає багато часу. Виникає потреба у створенні зручних онлайн-платформ, які дозволяють швидко та ефективно вибирати ПЗ, яке найкраще відповідає потребам бізнесу. Розробка таких інтернет-магазинів з багатокритеріальними фільтрами та персоналізованими рекомендаціями є важливим напрямом для спрощення цього процесу.

Тема є актуальною, оскільки допомагає оптимізувати процес вибору та придбання програмного забезпечення, що важливо для розвитку малого та середнього бізнесу, а також для великого корпоративного сектору, який потребує ефективних і швидких інструментів для автоматизації та управління.

Мета дослідження полягає в розробці та аналізу інтернет-магазину програмного забезпечення для бізнесу, який включає багатокритеріальні фільтри та рекомендаційну систему для полегшення процесу вибору і покупки ПЗ. Вивчення вимог до такої системи, її функціональних і нефункціональних характеристик, а також вивчення можливих проблем, з якими можуть стикатися користувачі в процесі вибору програмних рішень.

Завдання дослідження полягають в наступному:

1) проаналізувати ринок програмного забезпечення для бізнесу та виявити основні проблеми, з якими стикаються користувачі при виборі ПЗ;

2) розробити концепцію інтернет-магазину програмного забезпечення для бізнесу, що включає багатокритеріальне фільтрування та систему рекомендацій;

3) дослідити функціональні та нефункціональні вимоги до платформи;

4) розглянути існуючі рішення на ринку та визначити їх переваги та недоліки;

5) оцінити можливості використання різних технологій для розробки такої платформи;

6) розробити програмне забезпечення та основні алгоритми рекомендацій.

Об'єктом дослідження процес вибору та покупки програмного забезпечення для бізнесу через інтернет-магазин.

Предметом дослідження є система інтернет-магазину програмного забезпечення для бізнесу.

Методи дослідження полягають в наступному:

1. Аналіз літератури – для вивчення існуючих рішень на ринку, а також методів та інструментів, що використовуються в інтернет-магазинах ПЗ для бізнесу.

2. Метод порівняння – для аналізу існуючих платформ та визначення їх переваг і недоліків.

3. Метод системного аналізу – для вивчення вимог до функціональності платформи, а також для розробки структури та інтерфейсу системи.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (11 найменувань). Пояснювальна записка містить 15 рисунків. Загальний обсяг пояснювальної записки складає 45 сторінок, основний зміст викладено на 43 сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ІНТЕРНЕТ-МАГАЗИНУ ТОВАРІВ ДЛЯ ПРОДАЖУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ БІЗНЕСУ

1.1 Предметна область

Сфера електронної торгівлі, орієнтована на продаж програмного забезпечення для бізнесу, є однією з найдинамічніших і найвибагливіших ніш сучасного e-commerce. Її особливості визначаються як високими очікуваннями клієнтів, так і складністю самих продуктів. На відміну від традиційної B2C-торгівлі, тут клієнтом є не окрема людина, а компанія або IT-відділ, який шукає рішення, що здатне покрити конкретні бізнес-завдання. Це формує зовсім інший підхід до побудови цифрового магазину.

Однією з ключових сучасних особливостей є глибока деталізація інформації про товари. Бізнес-клієнти потребують не лише загального опису, а й повного набору характеристик, технічних специфікацій, умов ліцензування, варіантів масштабування та підтримки [1]. Важливим елементом стало впровадження багатокритеріальних фільтрів, які дають змогу швидко знаходити потрібне рішення серед сотень варіантів, комбінуючи параметри, що цікавлять наприклад, ціна, модель ліцензування (підписка, одноразова покупка), відповідність галузі, інтеграція з існуючими системами тощо.

Ще одна особливість це рекомендаційні системи. Вони базуються на аналізі поведінки користувачів, їхніх пошуків, переглядів, а також на даних про схожі компанії, що купували аналогічні продукти [2]. Такі системи допомагають не лише запропонувати схожі продукти, а й будувати складніші сценарії крос-продажів, наприклад, рекомендувати додаткові модулі чи супутні сервіси, що підвищують цінність основного продукту.

Сучасні платформи продажу ПЗ також відзначаються високим рівнем інтеграції з іншими бізнес-інструментами. Це можуть бути інтеграції з CRM,

ERP, платіжними шлюзами, системами управління підписками або навіть з внутрішніми платформами великих клієнтів. Така інтеграція дозволяє автоматизувати процеси виставлення рахунків, оновлення ліцензій, доставки оновлень та технічної підтримки.

Аналітика займає важливе місце у функціонуванні таких систем. Необхідно відстежувати поведінку клієнтів, популярність продуктів, конверсійні шляхи, ефективність рекомендацій, а також будувати прогнози продажів. Для бізнесу це стає не просто онлайн-вітриною, а потужним каналом взаємодії з клієнтами, що генерує дані для стратегічного планування.

Також сучасна e-commerce платформа для продажу ПЗ враховує потребу в персоналізації. Система має адаптуватися до кожного користувача відображати йому відповідні варіанти продуктів, враховуючи масштаб бізнесу, галузеві особливості, історію попередніх покупок та інтересів. Це допомагає побудувати довгострокові відносини, а не лише одноразові угоди.

Безпека є окремим критичним аспектом. Оскільки мова йде про продаж ліцензійних продуктів, облік ключів, ліцензійних прав, а також обробку фінансової інформації, необхідно дотримуватися найвищих стандартів кібербезпеки, включаючи шифрування даних, багатофакторну аутентифікацію та захист від шахрайства [3].

Водночас ринок продажу ПЗ для бізнесу вимагає підтримки гнучких моделей монетизації: одноразові покупки, підписки, freemium-моделі, корпоративні ліцензії, оплату за користування. Це ускладнює як технічну реалізацію, так і побудову інтерфейсів, що мають бути зрозумілими для бізнес-клієнтів.

Загалом сучасна e-commerce сфера для B2B-продажу програмного забезпечення це складна, високотехнологічна екосистема, що об’єднує каталогізацію, пошук, рекомендації, персоналізацію, аналітику, інтеграції та захист даних у єдине зручне рішення для корпоративного клієнта.

Головним завданням інтернет магазину в цій сфері допомогти користувачам швидко знайти оптимальне програмне забезпечення, яке

відповідає їхнім вимогам, і зменшити час, необхідний для аналізу та вибору ПЗ.

Кожен програмний продукт у такому магазині супроводжується детальним описом. У ньому вказується призначення програми, її основні можливості, сумісність із різними операційними системами, технічні вимоги, а також доступні типи ліцензій. Крім того, додаються відгуки інших користувачів, скріншоти або демонстраційні відео, що дозволяє скласти повне уявлення про продукт до моменту придбання. Програмне забезпечення зазвичай пропонується з різними моделями ліцензування – від одноразової покупки до щомісячної або щорічної підписки. Це дає змогу покупцю обрати найбільш зручний варіант оплати з урахуванням бюджету та планів розвитку компанії.

Особливу увагу також потрібно приділяти інтеграційним можливостям. Багато компаній вже використовують певні сервіси, тому важливо, щоб нове програмне забезпечення легко поєднувалося з існуючими системами. В описах товарів часто зазначено, з якими платформами та сервісами вони сумісні, чи є відкриті API, а також які інструменти для імпорту/експорту даних підтримуються.

Таким чином, інтернет-магазин програмного забезпечення для бізнесу з гнучкою системою фільтрації та персоналізованими рекомендаціями є ефективним рішенням для компаній, які прагнуть оптимізувати свою діяльність, підвищити продуктивність і зробити вибір програмних рішень простішим та більш обґрунтованим. Такий підхід поєднує в собі зручність електронної комерції та аналітичні інструменти, які дозволяють врахувати як потреби окремого бізнесу, так і загальні тенденції розвитку цифрового середовища.

Інтернет магазини в продажу програмного забезпечення мають такі товари:

- CRM-системи (для управління взаємодією з клієнтами);
- ERP-системи (для автоматизації внутрішніх бізнес-процесів);

– програмне забезпечення для обліку та звітності;

– програмні засоби для управління проектами;

– аналітичні інструменти для бізнесу;

– інструменти для кібербезпеки.

Області застосування такого типу інтернет магазинів може бути наступною:

– Малий і середній бізнес. Платформа допомагає підприємствам малого та середнього бізнесу вибирати ПЗ для управління фінансами, персоналом, проектами, комунікаціями з клієнтами тощо. Вона полегшує процес вибору недорогих і ефективних рішень для автоматизації повсякденних бізнес-операцій;

– Великий бізнес та корпорації. Для великих компаній інтернет-магазин надає доступ до більш складних рішень, таких як ERP-системи, CRM, системи для управління ланцюгами поставок, аналітики та звітності. Рішення можуть бути інтернаціональними або спеціалізованими для певних галузей (фінансовий сектор, виробництво, торгівля тощо);

– IT-компанії та постачальники програмного забезпечення. Платформа також служить місцем для розміщення та продажу ПЗ для розробників і постачальників програмних рішень, які можуть легко додавати свої продукти до каталогу, отримувати зворотний зв'язок від користувачів і аналізувати статистику продажів;

– Консалтингові та інтеграційні компанії. Організації, що надають консультаційні послуги з вибору та впровадження програмного забезпечення, можуть використовувати платформу для пошуку і рекомендованих рішень для своїх клієнтів, а також для інтеграції ПЗ у бізнес-процеси;

– Глобальний ринок програмного забезпечення. Програмний продукт має потенціал для використання на міжнародному рівні, оскільки дозволяє компаніям з різних країн знайти програмні рішення, які підходять до їхніх конкретних вимог з урахуванням специфіки їхнього ринку.

Таким чином, область застосування даного програмного продукту є дуже широкою, охоплюючи не лише бізнеси різних розмірів, а й постачальників програмного забезпечення та професіоналів, що працюють в цій сфері.

1.2 Огляд аналогів та обґрунтування очікуваних переваг порівняно з існуючими аналогами

Для визначення ключової цінності для розроблюваного нами інтернет-магазину розглянемо передових конкурентів. Одним з головних конкурентів є Microsoft AppSource. Це онлайн-магазин і платформа для пошуку, придбання та встановлення бізнес-додатків і рішень, які інтегруються з продуктами Microsoft, такими як Microsoft 365, Dynamics 365, Power Platform, а також Azure. AppSource пропонує широкий спектр додатків для різних бізнес-потреб, включаючи автоматизацію бізнес-процесів, аналітику, управління проектами, фінанси, маркетинг та інші [4].

Користувачі можуть знайти на платформі не тільки офіційні додатки від Microsoft, але й сторонні рішення від партнерів і розробників, що відповідають їхнім потребам. Microsoft AppSource (рис 1.1) допомагає компаніям швидко адаптувати та впроваджувати нові інструменти без необхідності створювати їх з нуля.

Платформа також забезпечує можливість тестування додатків перед покупкою, що дозволяє користувачам оцінити їхню ефективність та відповідність вимогам.

Продукт має наступні переваги:

– велика екосистема додатків, пов'язаних із Microsoft;

– глибока інтеграція з корпоративними рішеннями (ERP, CRM тощо);

– висока довіра серед бізнес-користувачів.

Недоліки:

– фокус на продуктах, що працюють у Microsoft-середовищі (інше ПЗ знайти складно);

– відсутність універсальної системи фільтрів та рекомендацій для підбору сторонніх програм.

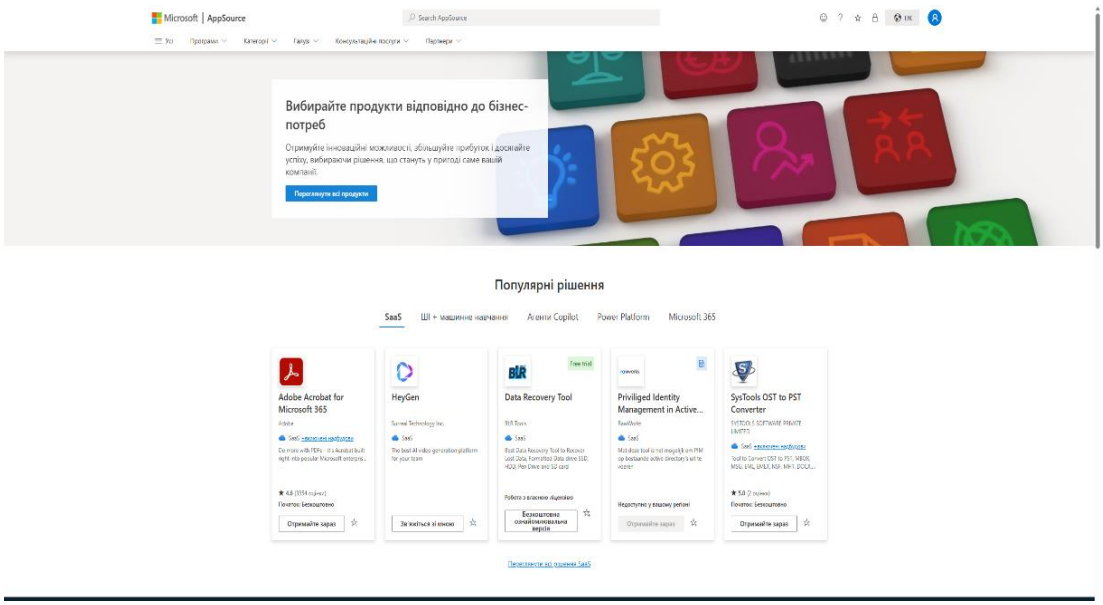

Рисунок 1.1 – Microsoft AppSource

Джерело: [5].

Наступним конкурентом для розгляду є AWS Marketplace (рис 1.2) – це онлайн-платформа, створена Amazon Web Services (AWS), яка дозволяє користувачам знаходити, купувати та запускати різноманітне програмне забезпечення та послуги, що працюють на хмарній інфраструктурі AWS. Це майданчик для сторонніх постачальників програмного забезпечення, де вони можуть пропонувати свої рішення, які інтегруються [4].

Переваги:

– орієнтація на підприємства, які використовують хмарні технології;

– широкий вибір корпоративного софту;

– автоматичне розгортання програм безпосередньо в AWS.

Недоліки:

– вузька спеціалізація (тільки хмарні сервіси, немає локальних рішень);

– високий рівень технічної складності для нефахівців;

– обмежені можливості персоналізованих рекомендацій.

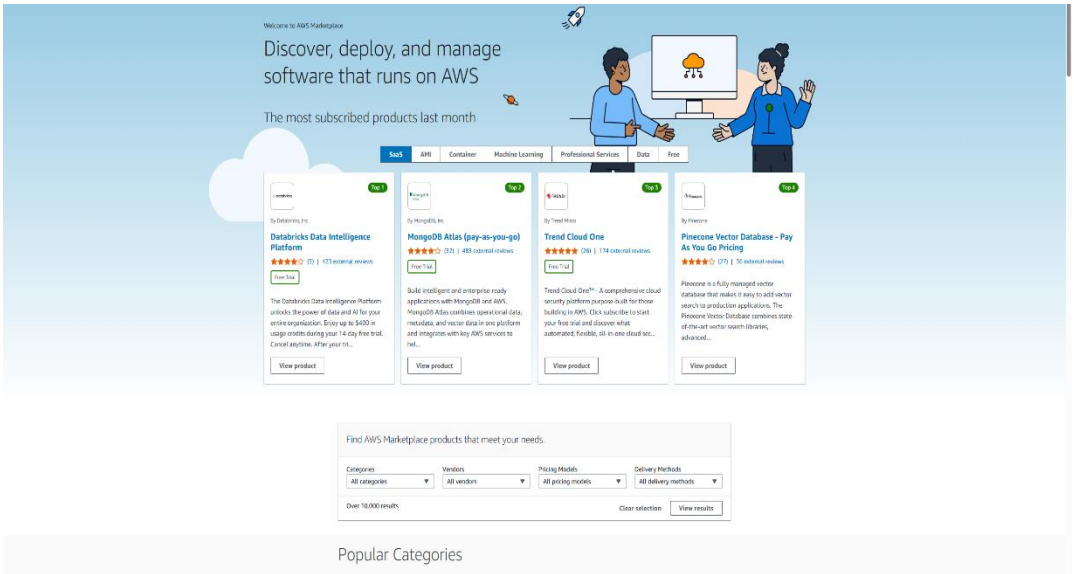

Рисунок 1.2 – AWS Marketplace

Джерело: [6].

Google Cloud Marketplace (рис 1.3) – це онлайн-платформа, створена компанією Google, яка дозволяє користувачам знаходити, купувати, та інтегрувати різноманітне програмне забезпечення та послуги з інфраструктурою Google Cloud [7]. Marketplace пропонує широкий спектр рішень від Google та сторонніх постачальників, які можуть бути використані для вирішення різних бізнес-задач, таких як управління даними, безпека, аналітика, штучний інтелект, автоматизація, і багато інших.

Переваги:

– великий вибір корпоративних рішень;

– інтеграція з Google Cloud Services;

– гнучкі варіанти ліцензування (оплата за використання, підписки тощо).

Недоліки:

– орієнтація лише на Google Cloud, що обмежує вибір клієнтів;

– відсутність ефективних інструментів для персоналізації вибору;

– більш підходить для великих компаній і розробників, ніж для малого та середнього бізнесу.

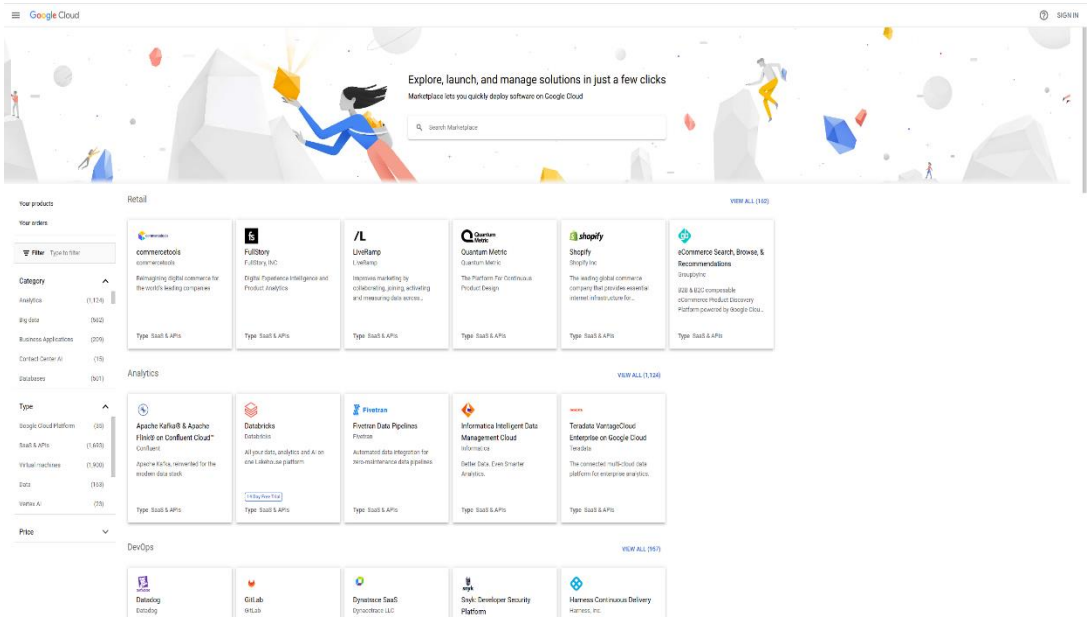

Рисунок 1.3 – Google Cloud Marketplace

Джерело: [8].

Аналіз конкурентів показав що розроблюваний інтернет-магазин програмного забезпечення для бізнесу повинен мати низку переваг шоб бути успішним на ринку. Основні конкурентні переваги полягають у гнучкості пошуку, персоналізованих рекомендаціях, інтегрованих аналітичних інструментах і покращеній взаємодії з постачальниками ПЗ. Розглянемо детальніше кожну з них:

1. Багатокритеріальна фільтрація. Існуючі платформи часто мають обмежені можливості сортування, тоді як наш продукт дозволяє фільтрувати ПЗ за широким набором параметрів: ціна, категорія, функціональність, ліцензія, рейтинг, інтеграції;

2. Рекомендаційна система на основі AI. Використання алгоритмів машинного навчання для персоналізованих рекомендацій, що враховують інтереси, історію переглядів та покупки користувача. Це підвищує релевантність пропозицій і спрощує вибір;

3. Порівняння продуктів. Можливість одночасного порівняння кількох програм за ключовими характеристиками, що дозволяє прийняти більш обґрунтоване рішення;

4. Прозорість і зворотний зв’язок. Детальна інформація про продукти, включаючи реальні відгуки користувачів, рейтинг та аналіз популярності, допомагає уникнути помилкових покупок;

5. Інтеграція з бізнес-інструментами. Підтримка API для взаємодії з CRM, ERP та іншими системами, що спрощує впровадження ПЗ у робочі процеси компаній;

6. Гнучка система ліцензування та оплати. Підтримка різних моделей оплати: одноразові покупки, підписка, пробний період. Це дозволяє користувачам обирати зручний формат використання програмного забезпечення.

1.3 Постановка задачі

Основне завдання кваліфікаційної роботи полягає в розробці інтернет-магазину програмного забезпечення для бізнесу з використанням багатокритеріальних фільтрів та рекомендаційної системи. Цей продукт повинен спрощувати процес вибору програмного забезпечення для підприємств різного розміру та галузі, дозволяючи їм знаходити оптимальні рішення для автоматизації бізнес-процесів.

В рамках роботи передбачається:

1. розробити концепцію інтернет-магазину програмного забезпечення для бізнесу;

2. визначити і реалізувати основні функціональні можливості: пошук, фільтрація, порівняння товарів, персоналізовані рекомендації;

3. визначити та реалізувати системні та нефункціональні вимоги до платформи;

4. створити саму програму та основні функції системи;

5. оцінити ефективність платформи шляхом тестування та збору відгуків.

Вхідні дані:

– дані користувача: інформація про користувача (ідентифікаційні дані, історія покупок, переваги тощо);

– каталог програмного забезпечення: дані про продукти (назва, опис, ціна, ліцензія, сумісність, рейтинг, відгуки, документація);

– запити користувача: введені дані для фільтрації або пошуку продуктів;

– технічні параметри платформи (можливості серверів, інтерфейс користувача).

Вихідні дані:

– результати пошуку та фільтрації продуктів;

– список рекомендованих програм для конкретного користувача;

– порівняння декількох продуктів за основними характеристиками;

– підсумки замовлення: підтвердження оплати, ліцензії, посилання на завантаження програмного продукту.

Апаратура та програмне забезпечення:

– сервери: платформа повинна працювати на сучасних веб-серверах із високою пропускною здатністю та можливістю масштабування;

– операційні системи: підтримка основних ОС (Linux, Windows Server);

– системи зберігання даних: база даних для збереження інформації про користувачів, товари, замовлення (MySQL);

– інтерфейси: підтримка API для інтеграції з іншими системами (платіжними, аналітичними).

Безпека та захист даних:

– шифрування всіх персональних даних користувачів (SSL, HTTPS);

– засоби захисту від атак (SQL-ін'єкції, XSS, CSRF);

– резервне копіювання даних та відновлення.

Масштабованість. Система повинна підтримувати збільшення кількості користувачів і товарів без значних змін у структурі.

Функціональні вимоги

1. Реєстрація та авторизація користувачів:

– користувач повинен мати можливість зареєструватися на платформі за допомогою електронної пошти або через соціальні мережі (Google, Facebook тощо);

– доступ до особистого кабінету, де користувач може переглядати історію покупок, управляти ліцензіями та налаштуваннями профілю.

2. Каталог товарів:

– можливість перегляду детальної інформації про кожен продукт: опис, ціна, ліцензія, сумісність, відгуки.

3. Пошук і фільтрація продуктів:

– багатокритеріальне фільтрування товарів (за ціною, категорією, ліцензією, функціональними можливостями тощо);

– пошук по ключових словах та категоріях.

4. Персоналізовані рекомендації:

– алгоритми рекомендацій, що пропонують користувачеві ПЗ на основі його попередніх покупок, переглядів і відгуків інших користувачів.

5. Порівняння товарів:

– користувач має можливість порівнювати кілька товарів одночасно за основними характеристиками.

6. Оформлення замовлення та оплата:

– можливість оформлення замовлення та оплати через платіжні системи (банківські картки, PayPal, електронні гаманці);

– надання ліцензій після підтвердження оплати.

Нефункціональні вимоги

1. Продуктивність:

– відповідь системи на запити користувача не повинна перевищувати 3 секунд;

– система повинна витримувати високе навантаження, забезпечуючи роботу платформи під час пикових навантажень.

2. Доступність:

– платформа повинна бути доступною 24/7;

– підтримка резервних копій для уникнення втрати даних.

3. Інтерфейс:

– інтерфейс повинен бути зручним та інтуїтивно зрозумілим для користувачів, з адаптивним дизайном для різних пристроїв (ПК, планшет, мобільний телефон).

4. Безпека:

– захист від несанкціонованого доступу до особистих даних користувачів;

– використання сучасних методів шифрування для збереження платіжної інформації.

Кроки для розв'язання завдання

1. Аналіз вимог. Збір та уточнення вимог до системи. Аналіз потреб користувачів та ринку.

2. Проектування архітектури системи. Визначення архітектури програмного продукту (база даних, сервери, інтерфейси).

3. Розробка інтерфейсу користувача. Дизайн інтерфейсу, який є простим у використанні і зручним для кінцевих користувачів.

4. Розробка функціоналу фільтрації та рекомендацій:

– реалізація багатокритеріальної фільтрації продуктів;

– створення алгоритмів персоналізованих рекомендацій.

5. Інтеграція з платіжними системами:

– налаштування платіжних шлюзів для прийому оплат.

6. Тестування:

– тестування всіх функцій платформи на предмет їх ефективності та коректності;

– проведення навантажувального тестування для оцінки продуктивності системи.

Висновки до розділу 1

В даному розділі було чітко визначено постановку завдання і описано де саме застосовується даний продукт і також які функції має містити даний продукт. Було також проаналізовані конкуренти даного продукту що вони з себе представляють, також описано які переваги буде містити продукт що буде зроблений в даній роботі. І також було описано чітку постановку завдання, тобто визначено завдання яке має бути зроблено, які вхідні дані вихідні дані, вимоги до ПЗ, системні функціональні, нефункціональні вимоги, що саме повинен робити готовий продукт, і також які кроки необхідно зробити щоб завдання було виконаним.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ІНТЕРНЕТ МАГАЗИНУ ТОВАРІВ ДЛЯ ПРОДАЖУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ БІЗНЕСУ

2.1 Моделювання поведінки продукту

Для розуміння того що саме інтернет-магазин повинен виконувати, було побудовано сценарії використання (рис 2.1). Діаграма варіантів використання – (діаграма прецедентів, use case) – дозволяє уявити типи ролей та їх взаємодію із системою. Проте не показує порядок виконання кроків. Зображує функціональні вимоги (те, що система може зробити) з точки зору користувача [11].

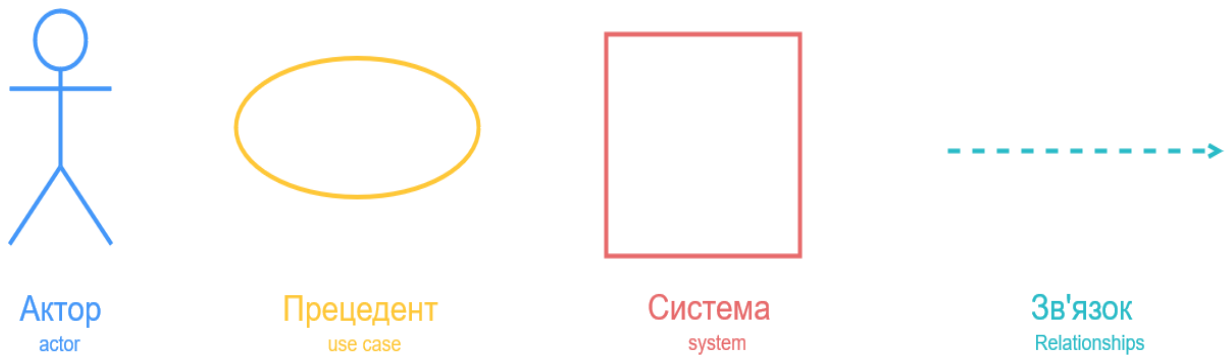

Рисунок 2.1 – Елементи з яких складається діаграма Use Case

Джерело: [9].

Сценарії використання містять ключові елементи:

1. Актор (Actor)

- це користувач або зовнішня система, яка взаємодіє із системою, що проектується;
- актор може бути людиною (наприклад, «Клієнт») або іншою системою (наприклад, «Банківський сервіс»);
- актори завжди зовнішні щодо системи.

2. Прецедент (Use Case)

- це конкретна функція або послуга, яку система надає актору;

- прецеденти описуються у вигляді дієслівних фраз: «Зареєструвати користувача», «Оформити замовлення».

3. Зв'язки між елементами:

- асоціація (лінія між актором і прецедентом): показує, що актор використовує певний варіант використання;

- include («включення»): один прецедент завжди включає інший. Наприклад, «Оформити замовлення» обов'язково включає «Підтвердити оплату»;

- extend («розширення»): один прецедент може розширювати інший за певних умов. Наприклад, «Отримати знижку» може бути додатковою можливістю при «Оформленні замовлення»;

- generalization («узагальнення»): актори або прецеденти можуть бути організовані за принципом наслідування. Наприклад, актор «Преміум-користувач» успадковує всі дії звичайного «Користувача», але має ще додаткові функції.

4. Межа системи (System Boundary):

- прямокутник, який оточує всі прецеденти;

- все, що знаходиться всередині прямокутника – це функціональність системи; все, що ззовні – актори.

На рис. 2.2, 2.3 представлені діаграми прецедентів (Use-case) де перша це для користувача, а друга для адміністратора.

На кожній діаграмі присутній відповідний актор, перший це звичайний користувач а другий це адміністратор. Кожний із них має свої завдання, і те з чим саме він взаємодіє.

Користувач в основному робить наступне, він реєструється, логіниться, він може переглядати товари, також може фільтрувати товари. Також він може додати товар/товари до кошика, оформити замовлення також може лайнкути товар/дизлайкнути товар. Також він може переглянути рекомендації, і вийти з системи.

Адміністратор в основному робить наступне, він входить під своїм ім’ям, він керує товарами, додає товари, редагує товари, видаляє товари. Також він керує замовленнями, керує користувачами, також він може вийти з системи.

Рисунок 2.2 – Use Case діаграма для користувача

Джерело: розроблено автором

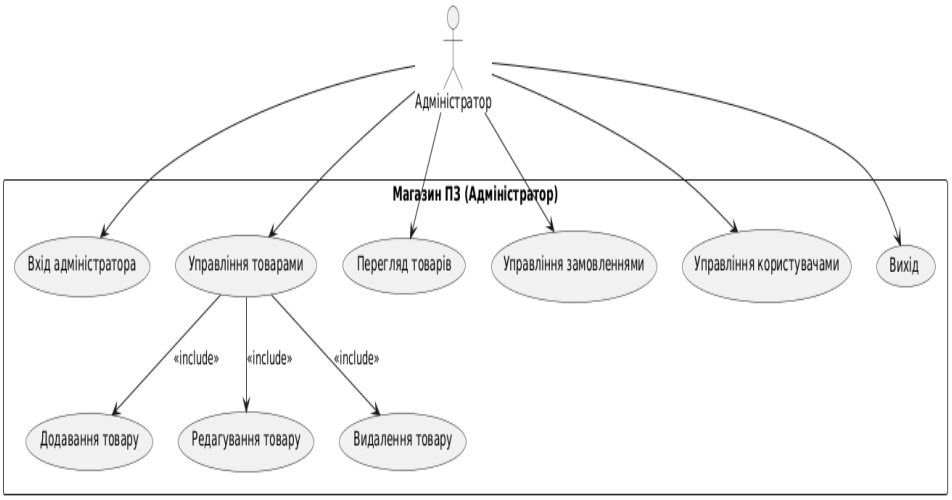

Рисунок 2.3 – Use Case діаграма для адміністратора

Джерело: розроблено автором

Для моделювання послідовності бізнес-процесів та дій, реалізованих методами класів було використано діаграму діяльності [10].

Основні елементи які необхідні для побудови діаграми представлено в таблиці 2.1

Таблиця 2.1 – Елементи діаграми діяльності

Елемент	**Опис**
Початкова точка	чорне коло, що позначає початок процесу.
Дія (Activity)	операція чи крок (зображений овалом або прямокутником з округленими кутами).
Перехід (Control Flow)	стрілка, яка вказує напрямок переходу від однієї дії до іншої.
Рішення (Decision Node)	ромб: процес може піти різними шляхами в залежності від умови
Злиття (Merge Node)	ромб: об'єднання кількох альтернативних шляхів в один.
Форк (Fork Node)	чорна смужка: один потік розділяється на кілька паралельних
Злиття потоків (Join Node)	чорна смужка: кілька паралельних потоків зливаються в один.
Фінал процесу	коло в колі: означає завершення процесу

На рис 2.4, 2.5 зображено діаграми активностей, де перша це для користувача, а друга для адміністратора.

Для користувача все починається з того що він переходить на сайт. Далі система запитує чи користувач взагалі зареєстрований чи ні. Якщо так він просто логіниться, а якщо ні то він спочатку реєструється а після цього логіниться. Після того як попередня дії/дія виконані йому стають доступні товари і він їх може переглядати. Після цього починається те що користувач може робити, він може фільтрувати товари, переглядати деталі товарів. Він також може поставити лайк/дизлайк товарів, якщо він це хоче зробити він це робить а якщо він цього робити не хоче він цей етап пропускає і переходить в додавання товару в кошик. Після цього йому пропонується оформити

замовлення, якщо він це бажає то він оформлює замовлення і оплачує його, а якщо ні то він пропускає два останніх етапи і переходить до перегляду рекомендацій. І після всього він може взагалі вийти з системи що знаменує кінець роботі сайту з точки зору користувача.

Рисунок 2.4 – Activity діаграма для користувача

Джерело: розроблено автором

Для адміністратора все починається з того що він переходить на сайт. Далі він заходить в систему під своїм іменем. Після того як він зайшов під своїм іменем йому стає доступне перегляд товарів. Після цього йому стає доступно додати продукт, якщо він хоче додати то він його додає а якщо ні то він пропускає цей етап. Далі йому стає доступно редагувати товар, якщо він хоче редагувати то він його редагує а якщо ні то він пропускає цей етап. Далі йому стає доступно видалити продукт, якщо він хоче то він його видаляє а якщо ні то він пропускає цей етап. Після цих трьох дій він ще також може переглянути замовлення користувачів, а також переглянути самих користувачів. А після всього він виходить із системи.

Рисунок 2.5 – Activity діаграма для адміністратора

Джерело: розроблено автором

2.2 Моделювання структури продукту

Діаграма класів в уніфікованій мові моделювання – це статична структурна діаграма, що демонструє властивості системи, класи, операції та зв’язки між об’єктами для опису структури системи.

Діаграма складається з наступних елементів: перший головний елемент це сам клас, він зображається у вигляді прямокутника який розділений на три частини. Ім’я класу воно знаходиться зверху. Атрибути це по суті змінні що описують стан об’єкта знаходяться посередині. Методи це по суті функції що описують стан об’єкта знаходяться на самому низу.

Атрибути і методи завжди помічені - мінус, + плюс, #. Якщо мінус то це private, якщо плюс то це public, якщо # то це protected.

Класи також завжди мають зв’язки. Асоціація – це стандартний зв’язок між двома класами. Агрегація – це відношення «ціле-частина», але частини можуть існувати окремо. Композиція – це сильніша форма агрегації: частини не можуть існувати без цілого. Успадкування – це один клас наслідує властивості іншого. Залежність – це коли клас тимчасово використовує інший клас для виконання функцій.

Також є ще Мультиплікація, вона показує скільки об'єктів одного класу може бути пов'язано з об'єктами іншого класу, «1» – один об'єкт, «0..*» – будь-яка кількість (0 або більше), «1..*» – принаймні один, «0..1» – або жодного, або один.

Тепер переходимо до діаграми класів що відповідає роботі. Перший клас Order, він відповідає за замовлення користувача. Цей клас містить дату замовлення, користувача, список товарів, суму тощо. Він містить поля: Id: унікальний ідентифікатор замовлення, UserId: хто саме зробив замовлення, OrderDate: дата і час замовлення, OrderItems: список товарів у замовленні (зв’язок із OrderItem), TotalPrice: загальна сума замовлення.

Наступний клас Product, він описує товар в магазині. Цей клас містить поля Id: ідентифікатор продукту, Name: назва товару, Description: опис товару,

Price: ціна товару, Category: до якої категорії відноситься товар, ImageUrl: фото товару те як воно виглядає, Rating: Який рейтинг займає.

Наступний клас OrderItem він по суті знаходиться всередині класу (Order). Один Order має багато OrderItemів. Він містить поля Id: ідентифікатор рядка замовлення, OrderId: до якого замовлення належить, ProductId: який саме товар, Quantily: кількість товару, PricePerUnit: вартість товару за одну одиницю, Order: саме замовлення.

Наступний клас User він по суті описує користувача який реєструється або щось купив у системі. Він містить поля Id: ідентифікатор користувача, Username: ім’я користувача, Email: електронна пошта користувача, PasswordHash: пароль користувача (але захешований).

Наступний клас Purchase по суті надає інформацію про купівлю. Він містить поля Id: ідентифікатор користувача, UserId: хто саме зробив замовлення, ProductId: який саме товар, PurchaseDate: коли саме був куплений товар.

Останній клас Recommendation він по суті надає користувачеві рекомендацію про товар. Він містить поля Id: ідентифікатор користувача, UserId: хто саме зробив замовлення, ProductId: який саме товар, Score: по суті це оцінка чи рейтинг важливості рекомендації.

На рис. 2.6 буде представлена сама діаграма класів яка ще більш краще покаже те як виглядають класи і те як і де вони знаходяться.

2.3 Опис архітектури продукту

Програмний продукт має клієнт-серверну архітектуру:

Клієнтська частина: написана на React.js з використанням бібліотек redux, axios, react-router-dom.

Серверна частина: написана на ASP.NET Core Web API з використанням стандартних технологій: Entity Framework Core, JWT Authentication.

На клієнті використовується React. Він відповідає за інтерфейс: відображення товарів, замовлень, профілю користувача і кошика.

Використовуються бібліотеки на кшталт react-router-dom для переходів між сторінками, axios для роботи з HTTP-запитами на сервер, а також redux для глобального керування станом програми.

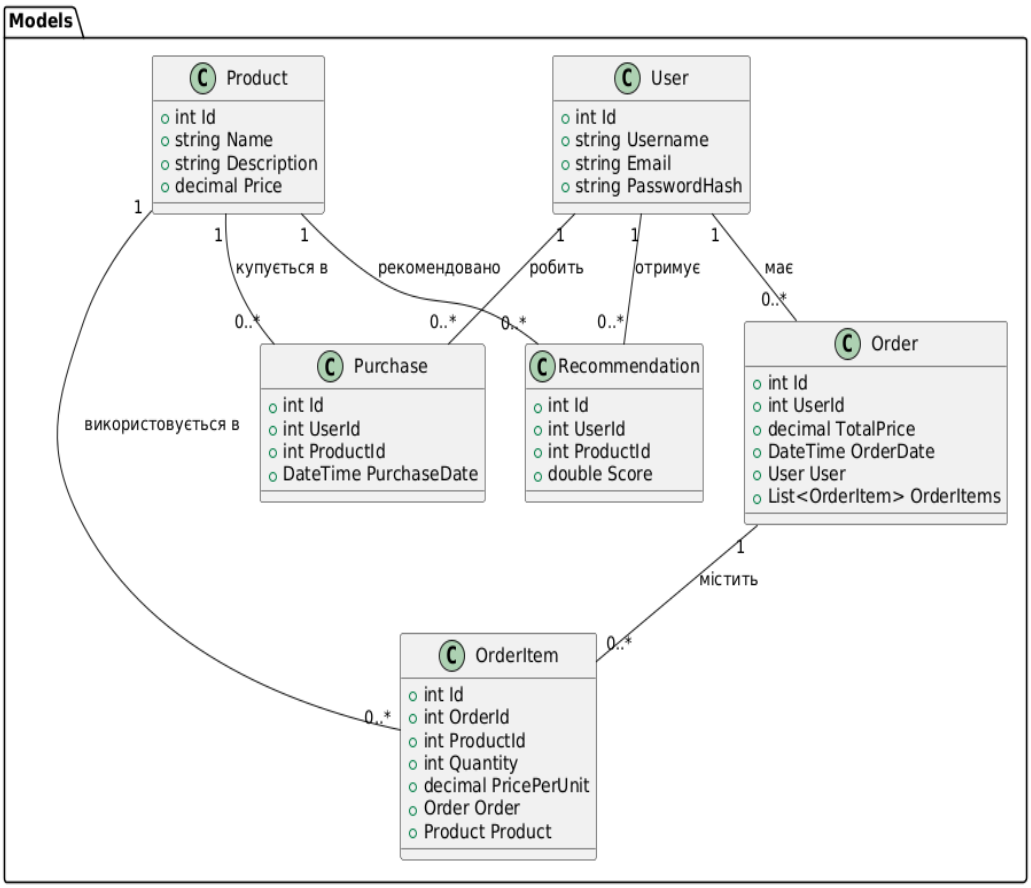

Рисунок 2.6 – Діаграма класів

Джерело: розроблено автором

На сервері використовується ASP.NET Core Web API. Він забезпечує логіку роботи додатку на бекенді: обробку запитів від клієнта, роботу з базою даних через Entity Framework Core, автентифікацію користувачів через JWT токени.

Детальніше про моделі. В папці Models зберігаються класи, які описують структуру даних, що зберігаються в базі.

Перший клас – це User. Він описує користувача системи: має Id, Email, пароль у вигляді хешу, можливо роль (чи це звичайний користувач, чи адміністратор).

Другий клас – це Product. Він описує товар: його назву, опис, ціну, кількість на складі, картинку і інші атрибути.

Третій – Order. Це замовлення користувача. У нього є Id, дата замовлення, загальна сума, статус замовлення, а також список товарів, які були замовлені.

Ще є OrderItem. Це допоміжний клас, який описує окрему позицію в замовленні – тобто який саме товар і в якій кількості був куплений.

Потім йде Purchase, але судячи з того, що ти казав, цей клас не зовсім правильний – бо його структура може дублювати Order. Можливо, його взагалі не потрібно або він має бути частиною Order.

І нарешті Recommendation – це рекомендації товарів для користувачів. Тут вказується, кому який товар рекомендується і з яким "рейтингом" чи "балом" важливості.

Усі ці класи об’єднані в єдиний контекст бази даних через ApplicationDbContext, який визначає, які таблиці створювати і як вони пов’язані між собою.

Кожен клас зазвичай має методи-конструктори (для створення об'єкта), геттери і сеттери (для доступу до властивостей), а також іноді якісь додаткові методи, наприклад для оновлення статусу замовлення чи перевірки кількості товару.

На фронтенді кожна сторінка – це окремий компонент. Наприклад, є HomePage, де виводяться товари, є OrdersPage, де користувач бачить свої замовлення, є сторінка деталей товару – ProductDetailsPage, і профіль користувача – ProfilePage. Ці сторінки отримують дані через запити до сервера за допомогою axios.

Контексти (AuthContext, CartContext) використовуються для того, щоб зберігати інформацію про авторизованого користувача і стан кошика в будь-якому місці програми.

На сервері для кожної групи даних є свій контролер, наприклад, UserController для роботи з користувачами, ProductController для роботи з

товарами, OrderController для замовлень. В кожному контролері є методи типу GET, POST, PUT, DELETE для відповідних CRUD-операцій.

Вся система працює так: користувач взаємодіє з React-інтерфейсом ➔ React відправляє запит через axios ➔ сервер обробляє цей запит і взаємодіє з базою даних ➔ сервер відправляє відповідь назад ➔ React оновлює інтерфейс на основі отриманих даних.

Висновки до розділу 2

Було проведено моделювання поведінки продукту тобто побудовано діаграми Use Case, Activity. Було проведено моделювання структури продукту тобто побудовано діаграму класів. Також була описана архітектура продукту.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ІНТЕРНЕТ МАГАЗИНУ ТОВАРІВ ДЛЯ ПРОДАЖУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ БІЗНЕСУ

3.1 Реалізація та конструювання програмного продукту

Для реалізації інтернет-магазину, система була поділена на фронтенд частину, бекенд частину та базу даних. Для цього було використано React, .NET Core та MySQL.

Фронтенд частина розроблена за допомогою мови JavaScript із використанням бібліотеки React, яка забезпечує компонентний підхід, високу інтерактивність інтерфейсу та гнучкість для розширення. Середовище розробки – Node.js із менеджером пакетів npm, оскільки ці інструменти є стандартом де-факто для React-додатків. Використані бібліотеки включають React Router для маршрутизації, Redux для управління станом, Tailwind CSS для стилізації, а також Zod для валідації даних. Вибір цих засобів обґрунтований їхньою популярністю, підтримкою великої спільноти, наявністю готових рішень для масштабування та інтеграції, а також простотою тестування.

Бекенд веб-додатку реалізований на платформі ASP.NET Core 8, що є сучасним кросплатформеним фреймворком від Microsoft для створення високопродуктивних веб-API. Проєкт реалізовано у вигляді REST-сервісу, що забезпечує обробку запитів від клієнтської частини, виконання бізнес-логіки, взаємодію з базою даних і формування відповідей у форматі JSON.

Бекенд побудовано за принципами чистої архітектури (Clean Architecture) з розділенням на:

- контролери;
- сервіси;
- репозиторії (Repository Pattern);
- моделі та DTO;
- Entity Framework Core.

Контролери відповідають за обробку HTTP-запитів від фронтенду. Вони розміщені в папці Controllers і мають атрибути [ApiController] та [Route("api/[controller]")]. Наприклад, контролер RecommendationsController обробляє запити на отримання рекомендацій для користувача через методи HttpGet, які викликають відповідні сервіси.

Сервісний шар (розташований у Services) реалізує бізнес-логіку: наприклад, алгоритми підбору рекомендацій, обробку замовлень, фільтрацію товарів. Використання сервісного шару дає змогу розділити логіку та контролери, підвищуючи підтримуваність і тестованість.

Репозиторії відповідають за доступ до даних: виконання запитів до бази, отримання й збереження об’єктів. Наприклад, IProductRepository може мати методи GetAllProducts(), GetProductById(id), GetProductsByFilter(criteria).

Моделі це класи, що відповідають структурі таблиць бази даних (наприклад, Product, Category). DTO це об’єкти, якими дані передаються між шарами, часто без полів, які не потрібні клієнту (наприклад, без технічних ID або службових полів).

Для роботи з MySQL використовується Pomelo.EntityFrameworkCore.MySql. В коді є контекст бази даних (DbContext), що конфігурується в appsettings.json для підключення, а також міграції, які дозволяють оновлювати схему бази при змінах моделей.

Було використано наступні технології:

- ASP.NET Core Web API – для побудови HTTP-контролерів;
- Entity Framework Core – ORM для роботи з MySQL;
- Pomelo.EntityFrameworkCore.MySql – офіційний провайдер для взаємодії з MySQL;
- Dependency Injection – для підключення сервісів у контролери;
- appsettings.json – для зберігання параметрів підключення та конфігурацій.

На рис. 3.1 представлено фрагмент реалізації рекомендаційного компонента на стороні клієнта, написаного з використанням бібліотеки React.

Компонент Recommendations відповідає за динамічне отримання даних рекомендацій із серверної частини через HTTP-запит до відповідного API-ендпоінта /api/recommendations/{userId}. Для збереження отриманих даних використовується React-хук useState, а хук useEffect забезпечує виконання запиту під час першого рендерингу компонента або при зміні ідентифікатора користувача.

Після отримання списку рекомендованих товарів, компонент відображає їх у вигляді HTML-списку. Такий підхід дозволяє реалізувати динамічний інтерфейс, що адаптується до поведінки користувача та даних, які надходять з бекенду. Компонент є частиною реалізації функціоналу персоналізації інтерфейсу, що є однією з сучасних тенденцій у розробці e-commerce систем і суттєво підвищує користувацький досвід.

Рисунок 3.1 – Реалізація рекомендаційного компонента

Джерело: розроблено автором

На рис. 3.2. наведено фрагмент реалізації серверного сервісу RecommendationService, який відповідає за бізнес-логіку формування рекомендацій. Сервіс реалізує інтерфейс IRecommendationService і взаємодіє з

базою даних через контекст AppDbContext. У наведеному прикладі використовується спрощений підхід: як рекомендації повертаються продукти з найбільшою кількістю продажів, що реалізовано через сортування за полем SalesCount і вибір топових позицій за допомогою методу Take(5).

Даний підхід демонструє реалізацію базової логіки системи персоналізованих рекомендацій у форматі REST-сервісу. Отримані дані передаються контролеру, який формує HTTP-відповідь для клієнтської частини. В майбутньому ця логіка може бути замінена або доповнена складнішими алгоритмами – наприклад, на основі машинного навчання, аналізу поведінки користувачів чи колаборативної фільтрації. Така гнучкість архітектури дозволяє масштабувати та вдосконалювати рекомендаційну систему без зміни зовнішнього інтерфейсу API.

```
public class RecommendationService : IRecommendationService
{
    Ссылок: 3
    private readonly AppDbContext _context;

    Ссылок: 0
    public RecommendationService(AppDbContext context)
    {
        _context = context;
    }

    Ссылок: 0
    public async Task<IEnumerable<Product>> GetRecommendationsForUserAsync(int userId)
    {
        var recommendations = await _context.Recommendations
            .Where(r => r.UserId == userId)
            .OrderByDescending(r => r.Score)
            .Take(5)
            .Select(r => r.ProductId)
            .ToListAsync();

        return await _context.Products
            .Where(p => recommendations.Contains(p.Id))
            .ToListAsync();
    }
```

Рисунок 3.2 – Реалізація серверного сервісу рекомендацій (RecommendationService.cs)

Джерело: розроблено автором

На рис. 3.3 зображено фрагмент Redux-слайсу cartSlice, який відповідає за логіку управління кошиком у фронтенд-додатку. Слайс містить три

редуктори: addToCart, removeFromCart і clearCart, які керують станом cartItems в залежності від дій користувача. Під час додавання товару до кошика перевіряється, чи вже існує такий товар у списку – якщо так, змінюється його кількість, інакше – товар додається до масиву. Після кожної операції стан кошика зберігається в localStorage, що дозволяє зберігати дані між сеансами.

Таке рішення є типовим прикладом використання Redux Toolkit для керування локальним станом додатка в React. Застосування localStorage гарантує, що користувацькі дані кошика зберігаються навіть після перезавантаження сторінки, що є важливим елементом для UX у системах електронної комерції. Цей код демонструє як ефективно поєднувати сучасні підходи до керування станом із локальним кешуванням даних на клієнті.

```
const cartSlice = createSlice({
    name: 'cart',
    initialState,
    reducers: {
        addToCart(state, action) {
            const item = action.payload;
            const existItem = state.cartItems.find(x => x.productId === item.productId);
            if (existItem) {
                existItem.quantity += item.quantity;
            } else {
                state.cartItems.push(item);
            }
            localStorage.setItem('cartItems', JSON.stringify(state.cartItems));
        },
        removeFromCart(state, action) {
            state.cartItems = state.cartItems.filter(x => x.productId !== action.payload);
            localStorage.setItem('cartItems', JSON.stringify(state.cartItems));
        },
        clearCart(state) {
            state.cartItems = [];
            localStorage.removeItem('cartItems');
        },
    },
});
```

Рисунок 3.3 – Реалізація логіки управління кошиком за допомогою Redux Toolkit

Джерело: розроблено автором

3.2 Тестування програмного продукту

Після завершення основної розробки програмного продукту було проведено всебічне тестування для перевірки його функціональності,

стабільності, зручності використання та сумісності з різними платформами. Основною метою тестування було забезпечити коректну роботу багатокритеріальних фільтрів, рекомендаційної системи, кошика замовлень, а також взаємодії між фронтендом і бекендом через REST API.

Тестування проводилося як вручну, так і з використанням автоматизованих підходів. Ручне функціональне тестування здійснювалося для всіх основних користувацьких сценаріїв: перегляд каталогу, фільтрація товарів за кількома параметрами, додавання товарів до кошика, отримання персоналізованих рекомендацій, оформлення замовлення (табл 3.1). Було протестовано граничні умови, наприклад: додавання великої кількості товарів до кошика, фільтрація за неіснуючими комбінаціями параметрів, порожні результати пошуку тощо.

Окрему увагу приділено модульному тестуванню бекенд-логіки зокрема сервісу рекомендацій. Для цього були створені юніт-тести з використанням бібліотеки xUnit, які перевіряли коректність логіки сортування й вибору рекомендованих товарів залежно від кількості продажів або категорії. З метою перевірки зв’язку з базою даних та її стабільності було проведено інтеграційне тестування за допомогою in-memory бази Sqlite, що дозволило перевірити роботу Entity Framework Core без впливу на основну базу.

На фронтенді виконано перевірку рендерингу компонентів за допомогою React Testing Library. Було протестовано компонент рекомендацій, роботу фільтрів, відображення товарів на головній сторінці та обробку подій натискання кнопок. Наприклад, перевірялося, що після натискання кнопки «Додати до кошика» відповідний товар відображається у списку, а стан зберігається у localStorage.

Крім функціонального, було проведено кросбраузерне тестування (Chrome, Firefox, Edge) та перевірка адаптивності на різних розмірах екранів. Результати тестування показали, що всі ключові функції працюють стабільно й відповідають вимогам, сформульованим на етапі проєктування. Виявлені незначні помилки в інтерфейсі були усунуті до фінального релізу системи.

Таким чином, реалізоване тестування підтверджує готовність програмного продукту до практичного використання. Забезпечено високий рівень якості, надійності та користувацької зручності, що є критичними чинниками для веб-платформи у сфері електронної комерції.

Таблиця 3.1 – Тест кейси для перевірки функціональності вебсайту

ID	**Назва тесту**	**Вхідні дані**	**Очікуваний результат**	**Статус**
TC-01	Фільтрація товарів за категорією	Категорія = 'CRM-системи'	Виведено лише товари з цієї категорії	Пройдено
TC-02	Додавання товару до кошика	Натискання кнопки 'Додати до кошика'	Товар додається до списку кошика	Пройдено
TC-03	Отримання рекомендацій	Користувач з ID = 5	Повертаються 5 рекомендованих товарів	Пройдено
TC-04	Очищення кошика	Натискання кнопки 'Очистити кошик'	Кошик стає порожнім	Пройдено
TC-05	Фільтрація за кількома критеріями	Категорія = 'ERP', Ціна < 500	Виведено лише ERP-програми з ціною < 500	Пройдено
TC-06	Збереження стану кошика	Перезавантаження сторінки після додавання товару	Кошик не порожній, товари збережені	Пройдено

3.3 Використання програмного продукту

Розроблений програмний продукт є веб-додатком типу «інтернет-магазин», спеціалізованим на підборі та рекомендації програмного забезпечення для бізнесу. Основна функціональність орієнтована на спрощення процесу вибору програмних рішень відповідно до галузі, розміру підприємства та типу необхідного ПЗ. Завдяки зручному інтерфейсу користувач має змогу швидко знайти релевантні рішення, порівняти їх та перейти до ознайомлення з деталями.

Головна сторінка інтерфейсу (див. рис 3.4) містить заголовок із закликом знайти ідеальне програмне забезпечення та інтерактивну панель із трьома

випадаючими списками. У першому користувач обирає галузь діяльності, у другому – розмір компанії, а в третьому – тип програмного забезпечення (наприклад, CRM, ERP, бухгалтерські системи тощо). Натискання на кнопку «Знайти рекомендації» викликає запит до серверної частини, який передає вибрані параметри. Після обробки запиту користувач бачить список рекомендованих продуктів, які відповідають заданим критеріям.

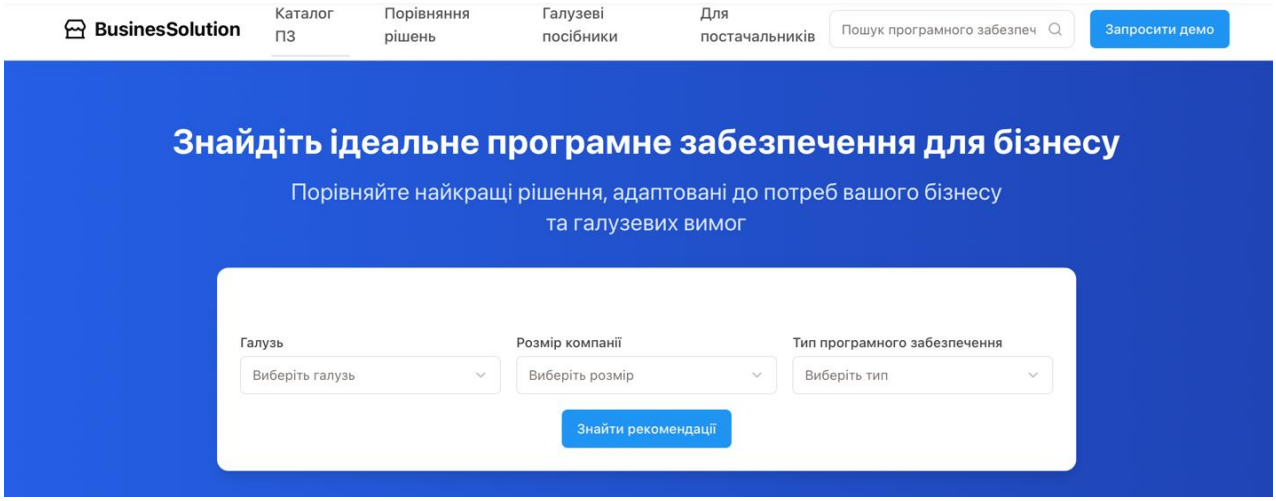

Рисунок 3.4 – Головна сторінка

Джерело: розроблено автором

На сторінці результатів представлені картки продуктів (рис 3.5), кожна з яких містить назву, рейтинг, короткий опис, рекомендовані типи компаній, ціновий діапазон та кнопки «View Details» і «Compare». Завдяки впровадженню багатокритеріального фільтра користувач може додатково уточнювати пошук за допомогою чекбоксів у лівій частині інтерфейсу. Доступні фільтри за галузями, розмірами компаній та діапазоном вартості. Зміни фільтрів динамічно оновлюють результати без необхідності перезавантаження сторінки.

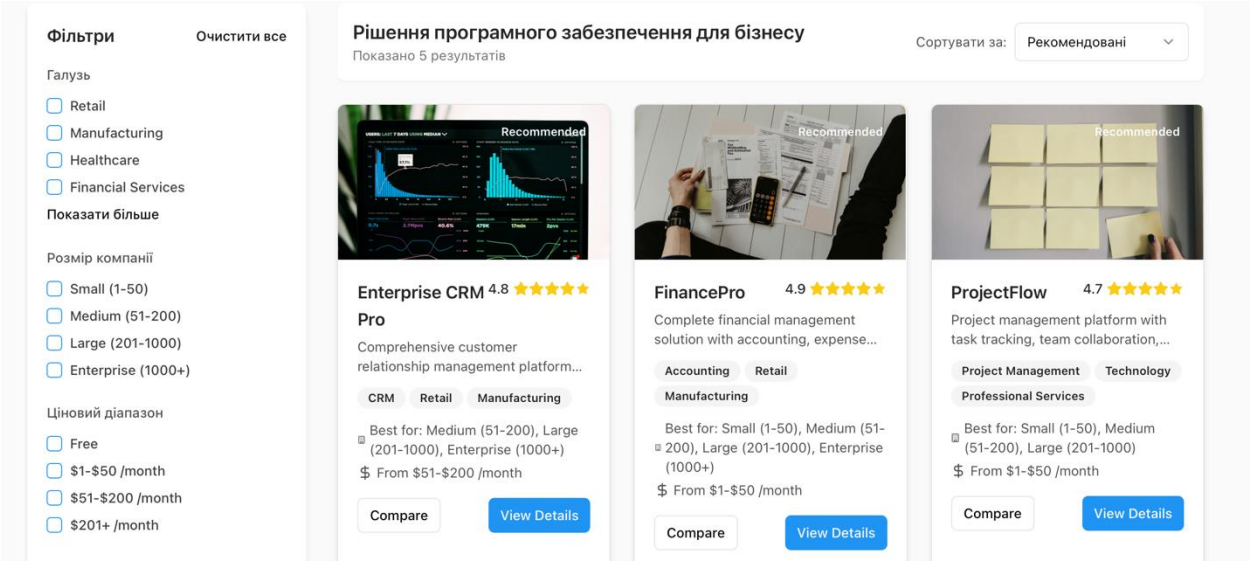

Рисунок 3.5 – Картки продуктів

Джерело: розроблено автором

Програмний продукт також реалізує модуль кошика. При натисканні на кнопку «Додати до кошика» товар зберігається у локальному сховищі браузера (localStorage), що дозволяє зберігати вибрані елементи навіть після оновлення сторінки. Управління кошиком відбувається через Redux-слайс cartSlice, який забезпечує функції додавання, видалення та очищення кошика. Ця функціональність ілюструє підхід до побудови зручної клієнтської логіки в рамках архітектури односторінкового додатку (SPA).

Інтерфейс адаптований для різних типів пристроїв, включаючи десктопи, планшети та мобільні телефони. У нижній частині сайту (рис3.6) розміщено футер, який містить додаткові навігаційні посилання на категорії ПЗ, ресурси для бізнесу, інформацію про компанію та її місію. Це сприяє кращій орієнтації користувача та полегшує доступ до супровідних матеріалів.

Таким чином, реалізований програмний продукт надає інтуїтивно зрозумілий, функціонально повний і візуально зручний інтерфейс, який дозволяє користувачам ефективно знаходити, аналізувати та обирати програмні рішення відповідно до потреб свого бізнесу. Застосування сучасних технологій фронтенду (React, Redux, Tailwind) у поєднанні з REST-сервісами

на ASP.NET Core забезпечує високу продуктивність, гнучкість та масштабованість системи.

Рисунок 3.6 – Футер інтернет-магазину

Джерело: розроблено автором

Висновки до розділу 3

У третьому розділі було докладно розглянуто реалізацію, функціонування та тестування розробленого веб-додатку – інтернет-магазину з багатокритеріальними фільтрами та системою рекомендацій для вибору програмного забезпечення бізнес-напрямку. Було обґрунтовано вибір інструментів реалізації як для клієнтської (React, Redux, Tailwind CSS), так і для серверної частини (ASP.NET Core, Entity Framework Core, MySQL), що забезпечило гнучкість архітектури, ефективну інтеграцію та зручність масштабування.

На основі аналізу функціональних вимог реалізовано ключові компоненти системи: рекомендаційний механізм, модуль фільтрації, інтерфейс кошика з локальним збереженням даних та систему управління користувацьким станом. Було розроблено сучасний та інтуїтивно зрозумілий інтерфейс, орієнтований на кінцевого користувача – представника малого або середнього бізнесу, що потребує адаптованих рішень у сфері ПЗ.

Важливим етапом стало тестування системи, яке включало функціональне, модульне, інтеграційне та UX-тестування. Результати підтвердили коректну роботу всіх основних функцій, стійкість до граничних ситуацій і відповідність очікуванням. Ретельне проєктування та реалізація

дозволили створити програмний продукт, який не лише відповідає вимогам технічного завдання, а й має потенціал для подальшого розвитку й комерційного застосування.

ВИСНОВКИ

У кваліфікаційній роботі було успішно вирішено завдання створення веб-додатку інтернет-магазину для продажу програмного забезпечення, орієнтованого на потреби малого, середнього та великого бізнесу. Основною метою було розробити систему, яка забезпечує ефективний пошук, фільтрацію та персоналізовану рекомендацію програмних рішень, що дозволяє користувачам з різним досвідом швидко знаходити оптимальні варіанти ПЗ для автоматизації своїх бізнес-процесів.

У процесі роботи було здійснено аналіз предметної області та конкурентного середовища, визначено функціональні й нефункціональні вимоги до системи, спроєктовано архітектуру та реалізовано основні компоненти – інтерфейс користувача, серверну логіку та сховище даних. Важливою особливістю реалізації є впровадження багатокритеріального фільтру, який дозволяє користувачеві гнучко налаштовувати параметри пошуку, а також рекомендаційної системи, яка враховує поведінкові фактори та продажі. Веб-додаток розроблений із використанням сучасних технологій: React, Redux, ASP.NET Core, Entity Framework та MySQL, що забезпечило високу масштабованість, зручність інтерфейсу та якісну обробку запитів.

Тестування показало стабільність роботи системи, відповідність очікуваним сценаріям використання, високу продуктивність та адаптивність до різних пристроїв. Таким чином, розроблений інтернет-магазин програмного забезпечення відповідає сучасним вимогам до B2B e-commerce платформ і може бути використаний як основа для подальшого розвитку та комерціалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kumar, V., Aksoy, L., Donkers, B., Venkatesan, R., Wiesel, T., & Tillmanns, S. (2010). Undervalued or overvalued customers: Capturing total customer engagement value. *Journal of Service Research*.

2. Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender Systems: An Introduction.

3. Chen, J., Zhang, C., & Xu, Y. (2009). The role of mutual trust in building members' loyalty to a C2C platform provider. *International Journal of Electronic Commerce*.

4. Amazon Web Services. (2025). Що таке AWS Marketplace? AWS Marketplace. URL: https://docs.aws.amazon.com/marketplace/latest/userguide/what-is-marketplace.html (дата звернення:07.04.2025)

5. Microsoft. (2025). AppSource. Microsoft. URL: https://appsource.microsoft.com/uk-ua/ (дата звернення: 07.04.2025)

6. Amazon Web Services. (2025). AWS Marketplace. Amazon Web Services. URL: https://aws.amazon.com/marketplace (дата звернення: 07.04.2025)

7. Cloudfresh. (2025). Що таке GCP та як ви можете використовувати його для свого бізнесу. Cloudfresh. URL: https://cloudfresh.com/ua/cloud-blog/shho-take-gcp-ta-yak-vy-mozhete-vykorystovuvaty-jogo-dlya-svogo-biznesu/ (дата звернення: 07.04.2025)

8. Google Cloud. (2025). Marketplace. Google Cloud. URL: https://console.cloud.google.com/marketplace?hl=uk&invt=AbuIMw (дата звернення: 07.04.2025)

9. Каграманова Ю. (2022). Як будувати UML-діаграми. DOU. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 02.05.2025).

10. Таврійський державний агротехнологічний університет імені Дмитра Моторного. (2025). Лабораторна робота №7: Діаграма діяльності.

URL: http://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/laboratorna-robota-78-diahrama-dijalnosti.pdf (дата звернення: 02.05.2025).

11. MindOnMap. (2023). Що таке діаграма класів UML і найкращий творець діаграм класів UML. MindOnMap. URL: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (дата звернення: 02.05.2025).