Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту

Завідувач
кафедри комп’ютерних наук,
_____к.т.н., доцент_____
(науковий ступінь, вчене звання)

_____ Ірина МАШКІНА
(підпис)
« ___ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.01 Інформатика

Тема:
Веб-застосунок для продажі продукту

Виконав
студент групи
ІНб-1-21-4.0д

Старцев Артем Олегович
(ПІБ)
_____
(підпис)

Науковий керівник:
_____к.т.н., доцент_____
(науковий ступінь, вчене звання)

_____ Вадим Абрамов
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»

Завідувач
кафедри комп'ютерних наук,
_____к.т.н., доцент_____
(науковий ступінь, вчене звання)

_____ Ірина МАШКІНА
(підпис)
« ___ » _____ 202__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІНУ РОБОТУ БАКАЛАВРА

«Веб-застосунок для продажі продукту»
Виконавець – студент групи ІНб-1-21-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Старцев Артем Олегович

1. Розроблена веб-система для полегшення процесу продажу продукту комп'ютерних комплектуючих елементів з використанням сучасного стеку технологій: React, Nest JS, PostgreSQL. Проведено аналіз предметної галузі, сформульовано функціональні вимоги, реалізовано архітектуру системи, створено діючий веб-прототип.
2. Основні завдання: *аналіз предметної галузі та обґрунтування актуальності створення веб-системи для адміністратора; огляд сучасних технологій веб-розробки й аналіз аналогічних рішень; визначення мети, об'єкта та предмета дослідження; обґрунтування вибору засобів і методів розробки; формування вимог до системи та побудова її архітектури; реалізація клієнтської і серверної частин з використанням React та Nest JS ; створення бази даних для управління інформацією; розробка та тестування прототипу системи; оцінка її ефективності; формування висновків і оформлення пояснювальної записки відповідно до вимог кафедри.*
3. Пояснювальна записка: *Обсяг – 24 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.*
4. Графічні матеріали: *не треба.*
5. Додатки: *немає.*
6. Строк подання роботи на кафедру: *«___» _____2025 р.*

Науковий керівник
к.т.н., доцент
_____ Абрамов Вадим Олексійович
дата

Виконавець:
_____ Старцев А.О.
дата

Календарний план

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Формування теми дипломної роботи бакалавра	16.11.2024	Виконано
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	25.11.2024	Виконано
3	Складання змісту та календарного плану роботи	29.11.2024	Виконано
4	Підбір літератури, складання завдання	29.11.2024	Виконано
5	Написання реферату та вступу	29.11.2024	Виконано
6	Написання першого розділу	14.12.2024	Виконано
7	Написання другого розділу	29.03.2025	Виконано
8	Написання третього розділу	29.03.2025	Виконано
9	Написання висновків	10.05.2025	Виконано
10	Виправлення зауважень	2.06.2025	Виконано
11	Створення презентації	22.05.2025	Виконано
12	Захист матеріалів дипломної роботи на засіданні кафедри	22.05.2025	Виконано
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	18.06.2025	

Здобувач_____

Керівник роботи _____

Реферат

Кваліфікаційна робота: 24 с., 2 рис., 18 літературних джерел.

Актуальність теми роботи полягає в необхідності продажі комп’ютерних комплектуючих. Сучасні торгові компанії потребують ефективних інструментів для автоматизації процесу продажу, а споживачі – зручного способу пошуку та їх покупки.

Об’єкт дослідження – процес продажі запчастин комп’ютера.

Предмет дослідження – методи, технології та інструменти розробки вебзастосунків для автоматизації продажу комп’ютера.

Метою роботи є підвищення ефективності процесу реалізації нерухомості шляхом створення веб-системи, яка забезпечує автоматизацію ключових етапів взаємодії адміністратора з клієнтом.

У ході виконання роботи було досліджено особливості предметної галузі, проведено огляд сучасних технологій веб-розробки, проаналізовано конкурентні рішення, а також спроектовано та реалізовано власну систему. Розроблений веб-застосунок надає можливість ефективно здійснювати їх публікацію, обробляти заявки від клієнтів.

Система реалізована з використанням ReactJS для фронтенду, що забезпечує компонентну структуру та швидку взаємодію з користувачем. На бекенді використано NestJS — фреймворк на базі Node.js з підтримкою TypeScript та модульної архітектури. Для зберігання даних застосовано PostgreSQL — надійну реляційну базу даних з підтримкою складних запитів.

Практичне значення полягає в можливості використання розробленої системи реальними компаніями у сфері продажу комплектуючих. Вона дозволяє зменшити часові витрати на обробку запитів, підвищити прозорість та контроль за етапами продажу, а також покращити взаємодію з потенційними покупцями.

Ключові слова: веб-застосунок, онлайн-продаж, фреймворк Reacrt, Nest js , PostgreSQL, автоматизація.

Зміст

Вступ

На сьогодні кількість користувачів, які займаються самостійним підбором комплектуючих для персональних комп'ютерів, постійно зростає. Це пов'язано зі збільшенням популярності індивідуальної збірки ПК, постійним оновленням технічних характеристик обладнання та бажанням отримати оптимальну конфігурацію за доступною ціною. Сучасні цифрові технології, зокрема спеціалізовані веб-застосунки, значно спрощують цей процес, дозволяючи швидко та зручно підібрати ідеальні компоненти для будь-яких потреб.

Сучасні інтернет-магазини комп'ютерних комплектуючих часто пропонують надто складні інтерфейси, що ускладнює покупку окремих деталей. Користувачам, які вже знають потрібний товар, доводиться долати зайві кроки - від пошуку в каталозі до оформлення замолення.

Основні проблеми існуючих рішень включають незручну навігацію, перевантажені технічними деталями сторінки та відсутність простої можливості швидко замовити конкретний товар. Особливо це заважає досвідченим користувачам, які хочуть лише швидко придбати чітко визначену комплектуючу.

Це створює потребу у спеціалізованому веб-рішенні, орієнтованому саме на швидку покупку окремих компонентів. Ключові вимоги до такого застосунку - мінімалістичний дизайн, пряме посилання на товар і максимально спрощений процес оформлення замовлення.

Актуальність теми дослідження обумовлена потребою у створенні спеціалізованого веб-застосунку з простим і швидким функціоналом для замовлення окремих комп’ютерних компонентів, що забезпечить зручність для досвідчених користувачів і підвищить ефективність процесу купівлі.

Мета дослідження — розробити веб-застосунок з мінімалістичним інтерфейсом для швидкої покупки окремих комп’ютерних комплектуючих.

Для досягнення мети необхідно виконати такі завдання:

- Провести аналіз існуючих веб-рішень у сфері онлайн-продажу комп’ютерних комплектуючих.

- Визначити оптимальний функціонал та вимоги до інтерфейсу з урахуванням принципів зручності користування.
- Розробити структуру бази даних товарів для ефективного зберігання та обробки інформації.
- Реалізувати прототип веб-застосунку з інтуїтивно зрозумілим користувацьким інтерфейсом.
- Провести тестування працездатності та зручності використання системи.

Об’єкт дослідження — процес онлайн-продажу окремих комп’ютерних комплектуючих.
Предмет дослідження — методи та інструменти розробки веб-застосунку, що забезпечує зручний інтерфейс і швидкий доступ до потрібних товарів.

Методи дослідження: аналіз науково-технічної літератури та існуючих веб-рішень; моделювання структури даних; проєктування інтерфейсів; програмування з використанням сучасних технологій веб-розробки; функціональне та UX-тестування.

Практичне значення отриманих результатів полягає у створенні універсального інструменту, який може бути використаний як самостійний сервіс або вбудований у існуючі інтернет-магазини, зокрема для малого та середнього бізнесу, з метою підвищення зручності та швидкості обслуговування користувачів.
Галузь застосування — електронна комерція, веб-програмування, роздрібна торгівля комп’ютерними комплектуючими. Результати роботи можуть бути апробовані шляхом розробки прототипу та представлення на студентських науково-практичних конференціях.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ТЕХНОЛОГІЧНИХ ПІДХОДІВ

На початковому етапі дослідження було проведено комплексний аналіз предметної галузі, що включав вивчення особливостей цільового сегменту ринку, потреб кінцевих користувачів та існуючих аналогічних рішень. Цей аналіз дозволив чітко визначити ключові вимоги до функціоналу системи та сформулювати технічне завдання для подальшої розробки.

Особливу увагу було приділено вивченню сучасних технологічних підходів до веб-розробки. Було проаналізовано переваги та недоліки різних стеків технологій, зокрема:

Для фронтенд-розробки розглядалися такі сучасні фреймворки як React.js, Angular та Vue.js, які дозволяють створювати динамічні інтерактивні інтерфейси. Особливість React.js у використанні віртуального DOM забезпечує високу продуктивність додатків, тоді як Angular пропонує повноцінне рішення для великих корпоративних проектів. Vue.js, у свою чергу, відзначається простотою інтеграції та гнучкістю.

У бекенд-розробці було досліджено можливості Node.js, Python (Django/Flask) та PHP (Laravel). Node.js дозволяє використовувати JavaScript на сервері, що спрощує розробку full-stack додатків. Python з його потужними бібліотеками ідеально підходить для реалізації складних алгоритмів обробки даних, а Laravel пропонує зручний інструментарій для швидкої розробки веб-додатків.

Для зберігання даних аналізувалися як реляційні (MySQL, PostgreSQL), так і нереляційні (MongoDB) бази даних. PostgreSQL відзначився підтримкою складних типів даних та високою продуктивністю, тоді як MongoDB показав себе як ідеальне рішення для роботи з неструктурованими даними та швидкого прототипування.

Окремий розділ дослідження був присвячений сучасним трендам веб-розробки, таким як:

Використання Progressive Web Apps (PWA) для забезпечення офлайн-роботи додатків

Застосування Server-Side Rendering (SSR) для покращення SEO

Використання мікросервісної архітектури для забезпечення масштабованості

Впровадження JAMstack архітектури для підвищення безпеки та продуктивності

Результати цього дослідження стали основою для обґрунтованого вибору технологічного стеку та архітектурних рішень при розробці власного

веб-додатку, що дозволило створити сучасне, продуктивне та масштабоване рішення, яке відповідає всім вимогам предметної галузі.

1.1. Аналіз особливостей предметної галузі

Сфера онлайн-продажів комп'ютерних комплектуючих зазнала суттєвих трансформацій у останні роки, що обумовлено стрімким зростанням популярності індивідуальної збірки ПК. Цей тренд підживлюється постійним оновленням технічних характеристик обладнання, прагненням користувачів до отримання оптимальної конфігурації за доступною ціною, а також активним розвитком геймінгу та творчих професій, які вимагають потужних апаратних рішень.

Проте сучасні інтернет-магазини часто не відповідають актуальним потребам ринку, демонструючи низку системних недоліків. Серед основних проблем варто відзначити надмірно складні інтерфейси, перевантажені технічними деталями, багатоступінчасті процеси оформлення замовлення, відсутність спеціалізованих інструментів для досвідчених користувачів та неефективну навігацію при пошуку конкретних компонентів. Ці обмеження особливо відчутні для двох ключових груп цільової аудиторії.

Першу групу становлять досвідчені ентузіасти, які чітко представляють собі необхідні компоненти і прагнуть мінімізувати час на оформлення замовлення, цінуючи прямий доступ до товару без зайвої інформації. Другу групу складають представники малого бізнесу - збірники ПК та сервісні центри, яким необхідний простий інтерфейс для регулярних закупівель з можливістю швидкого замовлення конкретних комплектуючих та отримання актуальної інформації про наявність товарів.

Враховуючи ці потреби, було сформульовано ключові вимоги до системи. Серед функціональних вимог варто виділити мінімалістичний інтерфейс з прямим доступом до товарів, миттєвий пошук за артикулом або технічними характеристиками, спрощену корзину з мінімальною кількістю кроків, можливість швидкого повторного замовлення та інтеграцію з системами обліку запасів. Нефункціональні вимоги включають час завантаження сторінок менше 1 секунди, адаптивний дизайн для мобільних пристроїв, надійний захист персональних даних користувачів та здатність обробляти понад 100 запитів на секунду.

Предметна галузь також має ряд специфічних технологічних викликів, таких як необхідність частого оновлення асортименту в базі даних, обробка великої кількості технічних параметрів, коливання цін на компоненти, а також проблема перевірки їх сумісності. Усі ці фактори були враховані при проектуванні архітектури майбутнього рішення, що дозволить створити по-справжньому ефективний інструмент для роботи на сучасному ринку комп'ютерних комплектуючих.

1.2. Огляд сучасних технологій веб-розробки

Сучасна веб-розробка характеризується стрімким розвитком технологій, що дозволяють створювати більш продуктивні, інтерактивні та масштабовані рішення. У даному розділі розглядаються ключові інструменти та підходи, які сьогодні визначають тренди у створенні веб-додатків.
Сучасний фронтенд базується на трьох основних технологіях: HTML5 (структура), CSS3 (стилізація) та JavaScript (логіка). Однак реальна потужність сучасних інтерфейсів розкривається завдяки фреймворкам та бібліотекам:

- React.js (Meta) – найпопулярніша бібліотека для створення динамічних інтерфейсів з використанням компонентного підходу. Основні переваги: віртуальний DOM для високої продуктивності, багата екосистема (Redux, Next.js) та підтримка серверного рендерингу.
- Vue.js – прогресивний фреймворк, що поєднує простоту інтеграції з потужними можливостями. Особливості: реактивність, однофайлові компоненти, плавна крива навчання.
- Angular (Google) – повноцінний MVC-фреймворк для складних корпоративних рішень. Переваги: TypeScript-підтримка, вбудовані інструменти тестування, ін'єкція залежностей.

У сучасній веб-розробці для стилізації інтерфейсів активно застосовуються різноманітні підходи та інструменти. CSS-препроцесори, такі як Sass та Less, значно розширюють можливості традиційного CSS, додаючи змінні, вкладені правила та міксини. Альтернативою є CSS-in-JS рішення на кшталт Styled Components, які дозволяють інкапсулювати стилі прямо в JavaScript-коді. Для прискорення розробки широко використовуються CSS-фреймворки - Tailwind CSS з його утилітарним підходом та Bootstrap з готовими компонентами, що особливо корисні для швидкого прототипування.

Бекенд-розробка сучасних додатків характеризується переходом до мікросервісної архітектури та serverless-підходів. Серед популярних технологій

виділяється Node.js, який дозволяє використовувати JavaScript на сервері та ідеально підходить для real-time додатків, з такими фреймворками як Express.js, NestJS та Koa. Python пропонує потужні рішення для бекенду через Django (з його "батарейками в комплекті") та мінімалістичний Flask, особливо ефективні для інтеграції з AI/ML. Традиційний PHP представлений сучасними фреймворками Laravel та Symfony, а для високонавантажених систем часто обирають Java/Kotlin з Spring Boot.

Вибір системи управління базами даних залежить від специфіки проекту. Реляційні СУБД, такі як PostgreSQL з підтримкою розширених типів даних та MySQL з його швидкодією, підходять для структурованих даних. Для роботи з неструктурованими даними використовують документоорієнтовані рішення на кшталт MongoDB, а для кешування та швидкісних операцій - Redis.

Сучасний DevOps-стек включає інструменти контейнеризації (Docker), оркестрації (Kubernetes), автоматизації CI/CD (GitLab CI/CD, GitHub Actions) та хмарні платформи (AWS, GCP, Azure). Архітектурні підходи до веб-розробки охоплюють SPA/PWA для клієнтської інтерактивності, JAMstack для безсерверної архітектури, GraphQL для ефективного API та WebAssembly для високопродуктивних обчислень.

Сучасний технологічний стек дозволяє створювати високопродуктивні, масштабовані, крос-платформенні та безпечні системи. Вибір конкретних технологій має ґрунтуватися на вимогах проекту, досвіді команди, можливостях довгострокової підтримки та розвиненості екосистеми. Головною тенденцією останніх років стало поєднання продуктивності, безпеки та розширюваності в єдиній архітектурі, що дозволяє розробникам створювати складні, але ефективні рішення для сучасного вебу.

1.3. Аналіз існуючих аналогічних рішень у сфері онлайн-продажу комп'ютерних комплектуючих

Сучасний ринок онлайн-продажу комп'ютерних комплектуючих пропонує різноманітні платформи, які можна класифікувати за такими типами:

- Універсальні маркетплейси (Rozetka, Prom.ua) – мають широкий асортимент, але складну навігацію для цільових покупок комплектуючих.
- Спеціалізовані магазини (Telemart, Citrus, Brain) – орієнтовані на IT-продукти, але часто мають перевантажений інтерфейс.

- Конфігуратори ПК (DNS Synthesizer, Moyo PC Builder) – пропонують інструменти для збірки, але не завжди зручні для швидкої покупки окремих деталей.

Проведений аналіз аналогічних веб-рішень виявив низку типових проблем, які негативно впливають на зручність користування та ефективність взаємодії з системами.

Складність навігації є однією з ключових проблем. У багатьох випадках користувачам доводиться проходити через багаторівневі категорії та численні фільтри, щоб знайти потрібний товар. Часто сторінки перевантажені великою кількістю технічних параметрів, що ускладнює вибір. До того ж, бракує швидкого доступу до найпопулярніших або рекомендованих компонентів, що змушує користувача витрачати зайвий час.

Неоптимальний процес покупки також є поширеним недоліком. Для оформлення замовлення часто потрібно виконати надто багато кроків, що знижує ефективність та створює бар'єри. Сторінки перенасичені нав'язливими рекомендаціями та супутніми пропозиціями. Крім того, обов'язкова реєстрація для здійснення покупки відлякує частину потенційних клієнтів.

Інтерфейсні проблеми включають в себе перевантаженість сторінок візуальними елементами, неочевидне розташування ключових функціональних блоків (ціна, наявність, кнопка «Купити») та відсутність адаптації під різні рівні підготовки користувачів, що ускладнює роботу як новачкам, так і досвідченим покупцям.

На основі виявлених недоліків було сформульовано основні вимоги до нового веб-застосунку, орієнтованого на спрощення користувацького досвіду та підвищення ефективності.

Мінімалістичний інтерфейс має забезпечити чітку, інтуїтивно зрозумілу структуру з прямим доступом до категорій і товарів. Уникаючи зайвих візуальних елементів, інтерфейс має бути легким для сприйняття. Обов'язковим є адаптивний дизайн, що забезпечить коректну роботу на різних типах пристроїв — від смартфонів до настільних комп'ютерів.

Оптимізований процес покупки передбачає можливість оформлення замовлення в один-два кліки. Основна увага має бути зосереджена на ключових характеристиках товару. Кількість обов'язкових полів при оформленні має бути зведена до мінімуму, а реєстрація — необов'язковою.

Інтелектуальний пошук має включати швидке фільтрування товарів за ключовими параметрами, автоматичні підказки при введенні запитів та зручні інструменти для порівняння характеристик різних позицій.

Технічний аналіз наявних конкурентних рішень показав, що в більшості випадків використовується сучасний технологічний стек. Для клієнтської частини застосовуються React.js або Vue.js, що забезпечують високу продуктивність та гнучкість інтерфейсу. На серверній стороні домінують Node.js та PHP.

У сфері баз даних спостерігається змішане використання реляційних систем (MySQL, PostgreSQL) і документоорієнтованих (MongoDB), залежно від потреб у структурі даних. Крім того, більшість рішень реалізовують RESTful або GraphQL API, що дозволяє інтегруватися з мобільними додатками, сторонніми сервісами та платіжними системами.

У результаті аналізу сформовано бачення майбутнього веб-застосунку, що має суттєві переваги над існуючими аналогами.

Спрощений UX включає інтуїтивний інтерфейс без перевантаження, прямий доступ до популярних товарів та можливість швидкого замовлення без обов’язкової реєстрації. Це значно покращує загальну зручність використання.

Технічні переваги передбачають використання сучасних фреймворків, оптимізовану архітектуру для ефективної роботи з каталогом, а також забезпечення швидкого завантаження сторінок навіть при великій кількості даних.

Бізнес-переваги полягають у зменшенні часу на оформлення замовлення, зниженні рівня відмов та підвищенні конверсії за рахунок кращої взаємодії з користувачем.

Таким чином, нове рішення має не лише покращити користувацький досвід, але й забезпечити конкурентоспроможність на ринку завдяки технологічній та функціональній перевазі.

2. ФОРМУВАННЯ ВИМОГ ТА АРХІТЕКТУРНІ РІШЕННЯ ПРОЕКТУ

2.1. Визначення та специфікація вимог до системи

Процес формування вимог до системи є фундаментальним етапом у розробці будь-якого програмного продукту. Він починається з аналізу потреб бізнесу та кінцевих користувачів, що був проведений на попередніх етапах дослідження. На основі цього аналізу формується чіткий перелік функціональних та нефункціональних вимог, які мають бути реалізовані в системі.

Функціональні вимоги детально описують конкретні можливості системи, які вона повинна надавати користувачам. Кожна функція має бути чітко специфікована з описом вхідних даних, процесу обробки та очікуваних результатів. Важливо враховувати всі можливі сценарії використання, включаючи стандартні та граничні випадки.

Нефункціональні вимоги визначають якісні характеристики системи, такі як продуктивність, безпека, масштабованість, зручність використання та сумісність. Ці вимоги встановлюють критерії, за якими буде оцінюватись якість готового продукту. Особливу увагу приділяють вимогам до продуктивності, які мають враховувати очікуване навантаження на систему та час реакції на різні типи запитів.

Специфікація вимог також включає визначення обмежень, таких як бюджет проекту, терміни реалізації, вимоги до апаратного забезпечення та інші фактори, які можуть вплинути на процес розробки. Усі вимоги документуються у вигляді технічного завдання, яке слугує основним керівним документом для команди розробників.

2.2. Обґрунтування вибору інструментарію розробки

Вибір технологічного стеку для реалізації проекту є стратегічним рішенням, яке впливає на всі етапи життєвого циклу розробки. В основі цього вибору лежить такі ключові фактори, як специфіка проекту, досвід команди, масштабованість та довгострокова підтримка.

Для фронтенд-розробки обрано ReactJS – сучасний JavaScript-фреймворк, який забезпечує компонентний підхід для побудови масштабованих інтерфейсів, швидку взаємодію з користувачем завдяки віртуальному DOM (без перезавантаження сторінки), а також має велику екосистему та активну спільноту, що спрощує розробку та підтримку.

Бекенд-частина реалізована на NestJS – прогресивному фреймворку на базі Node.js, який пропонує модульність, підтримку TypeScript для підвищення

якості коду, архітектуру MVC для зручної організації проекту, а також вбудовані інструменти для створення ефективних API та інтеграції з різними сервісами.
Як основну базу даних використовується PostgreSQL – надійна реляційна СУБД з підтримкою складних SQL-запитів і транзакцій, високим рівнем безпеки та стабільності, а також гнучкістю завдяки розширюваності і можливості роботи з JSON.
Для забезпечення ефективного процесу розробки застосовуються:

- система контролю версій Git;
- CI/CD-інструменти (GitHub Actions, GitLab CI) для автоматизації тестування та деплою;
- хмарні платформи (AWS, Google Cloud, Azure) для масштабованого розгортання.

Остаточний вибір технологій базується на аналізі продуктивності, вартості підтримки, доступності фахівців та можливості майбутнього масштабування. Це дозволяє досягти оптимального балансу між потужністю рішення, швидкістю розробки та стабільністю системи в довгостроковій перспективі.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЛУАТАЦІЙНІ АСПЕКТИ

3.1. Реалізація структури та компонентів веб-сайту

Процес практичної реалізації починається з розробки базової структури веб-сайту, яка включає модулі та компоненти, що забезпечують його функціональність. На цьому етапі створюється каркас додатку з чітко визначеними залежностями між різними частинами системи. Для фронтенд-частини використовується компонентний підхід, де кожен елемент інтерфейсу (навігаційне меню, форми, картки товарів тощо) розробляється як окремий модуль з власною логікою та стилями.
Серверна частина організовується за принципом MVC (Model-View-Controller) або сучаснішими архітектурними підходами, що дозволяють чітко розділити бізнес-логіку, роботу з даними та представлення інформації. Особливу увагу приділяють системі маршрутизації, яка визначає, як клієнтські запити спрямовуються до відповідних обробників на сервері.

3.2. Проектування структури даних

Проектування структури даних є критично важливим етапом, який визначає ефективність роботи всього застосунку. На основі аналізу вимог створюється детальна ER-діаграма (Entity-Relationship), яка відображає основні сутності системи та зв'язки між ними. Для реляційних баз даних проводиться нормалізація таблиць до відповідних нормальних форм для уникнення надмірності даних та забезпечення цілісності.
У випадку використання NoSQL рішень проектування орієнтоване на оптимальну організацію документів або інших структур даних з урахуванням характерних для них підходів до зберігання інформації. Важливим аспектом є визначення індексів для прискорення пошуку та оптимізації запитів, а також розробка стратегій кешування для часто використовуваних даних.

3.3. Розробка архітектури та логіки веб-застосунку

Архітектура застосунку будується з урахуванням принципів масштабованості та підтримки. Для сучасних веб-додатків часто використовується мікросервісна архітектура, де функціональність розподілена між незалежними сервісами, що спілкуються через API. Альтернативою є монолітний підхід, який підходить для менших проектів з меншою складністю.

Розробка бізнес-логіки включає створення сервісного шару, який інкапсулює основні правила роботи системи, та репозиторіїв для роботи з даними. Важливо дотримуватися принципів SOLID та застосовувати патерни проектування, що спрощують подальшу підтримку та розширення функціоналу. Для обробки запитів реалізується система middleware, яка дозволяє централізовано керувати аутентифікацією, логуванням та обробкою помилок.

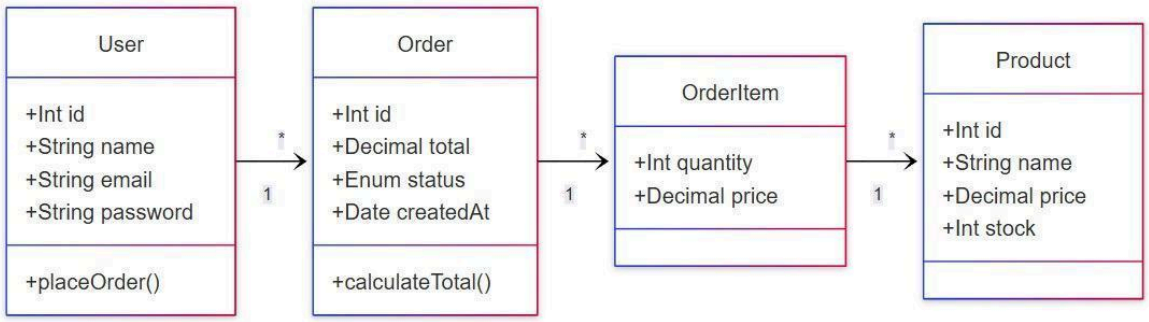

Рисунок 1.1 Діаграма класів

Опис та пояснення діаграми класів

Ця діаграма представляє основні сутності (класи) системи інтернет-магазину та їх взаємозв'язки. Вона використовує нотацію, схожу на UML, де:

- + означає public доступ (атрибути/методи)
- Назви класів написані зверху
- Атрибути та методи перераховані під назвою класу

1. Основні класи системи

User (Користувач)

Атрибути:

- id: Int – унікальний ідентифікатор
- name: String – ім'я користувача
- email: String – електронна пошта (використовується для входу)
- password: String – зашифрований пароль

Призначення: Відповідає за облікові дані покупця, авторизацію та управління профілем.

Product (Товар)

Атрибути:

- id: Int – унікальний ідентифікатор товару
- name: String – назва товару
- price: Decimal – ціна
- stock: Int – кількість на складі

Призначення: Описує товари, доступні для замовлення. Атрибут stock дозволяє контролювати наявність.

Order (Замовлення)

Атрибути:

- id: Int – номер замовлення
- total: Decimal – загальна сума
- status: Enum – статус (наприклад, "Нове", "Оплачене", "Відправлене")
- createdAt: Date – дата створення

Призначення: Фіксує інформацію про замовлення, його стан та суму.

OrderItem (Позиція замовлення)
Атрибути:

- quantity: Int – кількість товару
- price: Decimal – ціна за одиницю на момент замовлення

Призначення: Зв'язує товари (Product) із замовленнями (Order). Зберігає актуальну ціну та кількість, щоб історія замовлень залишалася незмінною (навіть якщо ціна товару зміниться).
2. Методи системи

- placeOrder() – відповідає за оформлення замовлення (переведення кошика в статус "Замовлення").
- calculateTotal() – розраховує загальну суму на основі позицій (OrderItem).

3. Зв'язки між класами

- User → Order: Користувач може мати багато замовлень (1-to-many).
- Order → OrderItem: Одне замовлення містить кілька позицій (1-to-many).
- OrderItem → Product: Кожна позиція посилається на конкретний товар (many-to-1).

4. Пояснення ключових особливостей
Інкапсуляція даних: Атрибути класів (наприклад, password) захищені від прямого доступу (+ вказує на public-рівень, але на практиці часто використовуються гетери/сетери).
Гнучкість:

- status: Enum дозволяє легко додавати нові стани замовлень (наприклад, "Скасоване").
- OrderItem.price фіксує ціну на момент покупки, що важливо для історії.

Нормалізація даних: Класи розділені для уникнення дублювання (наприклад, товари не повторюються в замовленнях).

Use case діаграмма

Опис та аналіз Use Case діаграми інтернет-магазину

Дана діаграма випадків використання відображає ключові сценарії взаємодії між покупцем (основним актором) та системою інтернет-магазину. Вона охоплює всі основні функціональні можливості, що надаються кінцевому користувачу, а також деякі адміністративні функції, які, ймовірно, потребують уточнення.

Структура діаграми

Діаграма включає три ключові компоненти:

Актор "Покупець" - основний користувач системи, який взаємодіє з магазином.

Система "Магазин" - набір функціональних можливостей, що реалізують логіку роботи інтернет-магазину.

Випадки використання (Use Cases) - конкретні дії, доступні покупцю в системі.

Основні випадки використання та їх значення

➔ Перегляд товарів

- Дозволяє користувачам ознайомлюватися з асортиментом, переглядати описи, ціни та наявність товарів.
- Значення: базовий функціонал для здійснення вибору.

➔ Реєстрація/Вхід

- Реєстрація передбачає створення облікового запису з введенням персональних даних.
- Вхід забезпечує доступ до особистого кабінету та історії замовлень.
- Значення: обов'язкова умова для оформлення замовлень.

➔ Додати до кошика

- Функція додавання обраних товарів до віртуального кошика.
- Значення: формування списку товарів для подальшого замовлення.

➔ Оформити замовлення

- Процес підтвердження замовлення з вказівкою деталей доставки та оплати.
- Значення: фіналізація процесу покупки.

➔ Оплатити замовлення
- Проведення платежу через інтегровані платіжні системи.
- Значення: завершальний етап оформлення замовлення.

➔ Управління товарами (для адміністратора)
- Додати товар: розширення асортименту магазину.
- Редагувати товар: корекція інформації про товари.
- Видалити товар: вилучення позицій з каталогу.
- Примітка: ці функції, ймовірно, належать до ролі адміністратора і потребують винесення в окрему діаграму.

Взаємозв'язки між випадками використання
Система передбачає чітку послідовність дій:
- Для оформлення замовлення користувач повинен пройти автентифікацію (реєстрацію/вхід).
- Оплата завжди відбувається після успішного оформлення замовлення.
- Додавання товарів до кошика є обов'язковою умовою для подальшого оформлення.

3.4. Тестування функціональності серверного середовища

Тестування серверної частини включає кілька рівнів перевірки. Модульні тести покривають окремі компоненти системи, перевіряючи коректність роботи кожної функції. Інтеграційні тести демонструють правильність взаємодії між різними частинами системи, включаючи з'єднання з базою даних та зовнішніми сервісами.
Для перевірки продуктивності проводять навантажувальні тести, які імітують роботу системи в умовах, близьких до реального використання. Особливу увагу приділяють тестуванню безпеки, включаючи перевірку на вразливості (SQL-ін'єкції, XSS, CSRF тощо). Всі тести автоматизуються та інтегруються в процес CI/CD для забезпечення стабільності кодової бази.

3.5. Інтерфейс користувача: огляд реалізованих сторінок та інструкція

Інтерфейс користувача розробляється з урахуванням принципів юзабіліті та доступності. Кожна сторінка проходить кілька ітерацій дизайну та тестування з

реальними користувачами для виявлення потенційних проблем у взаємодії. Основні екрани включають головну сторінку, панель керування, форми введення даних та сторінки з результатами.
Для забезпечення зручності використання створюється детальна інструкція, яка описує основні сценарії роботи з системою. Вона містить покрокові керівництва, знімки екранів та пояснення ключових функцій. Інструкція розробляється як для початкових користувачів, так і для адміністраторів системи, з урахуванням їх різних потреб.
3.6. Розгортання проекту та підготовка до експлуатації

Процес розгортання включає налаштування виробничого середовища, яке відрізняється від розробничого більш суворими вимогами до безпеки та продуктивності. Виконується налаштування веб-сервера (Nginx, Apache), конфігурація балансувальників навантаження та налаштування системи моніторингу (Prometheus, Grafana).
Для забезпечення безперебійної роботи реалізується стратегія розгортання без простоїв (blue-green deployment або canary releases). Готовиться документація для адміністраторів, яка включає процедури резервного копіювання, відновлення після збоїв та масштабування системи. На завершальному етапі проводиться навчання персоналу, який буде експлуатувати систему, та розробляється план технічної підтримки.

Висновок:

У ході виконання даної курсової роботи було розроблено повноцінний веб-додаток, що відповідає сучасним вимогам до інформаційних систем. Проект включав повний цикл розробки - від аналізу предметної галузі та вивчення аналогічних рішень до практичної реалізації та розгортання готового продукту.

Основним результатом роботи став функціональний веб-застосунок, архітектура якого базується на мікросервісному підході з використанням сучасного технологічного стеку. Вдалося реалізувати всі заплановані функції, забезпечивши при цьому високу продуктивність та зручність користувацького інтерфейсу. Особливу увагу було приділено аспектам безпеки та масштабованості системи.

Практичний досвід, отриманий під час виконання проекту:
Робота над цим проектом надала мені безцінний практичний досвід у кількох ключових аспектах сучасної веб-розробки. Одним із найважливіших досягнень

стало освоєння принципів проектування складних структур даних, де я навчився створювати ефективні моделі баз даних, оптимізувати зв'язки між сутностями та застосовувати методи нормалізації для забезпечення цілісності інформації. Особливу цінність мав досвід роботи з сучасними фреймворками та бібліотеками. Практичне застосування таких інструментів як React для фронтенду та Node.js для бекенду дозволило глибше зрозуміти їх архітектурні особливості, переваги та обмеження. Це включало вивчення компонентного підходу до розробки інтерфейсів, роботу зі станом додатку та використання сучасних патернів програмування. Важливим етапом стало інтегрування різних компонентів системи, де я набув навичок створення API, роботи з мікросервісною архітектурою та забезпечення взаємодії між клієнтською та серверною частинами. Цей досвід особливо цінний тим, що дозволив зрозуміти всі етапи життєвого циклу веб-додатка - від ідеї до реалізації. Значний прогрес було досягнуто у сфері автоматизації тестування та розгортання. Робота з CI/CD інструментарієм, системами контролю версій та хмарними платформами розгортання дала чітке уявлення про сучасні підходи до DevOps, що є критично важливим для будь-якого професійного розробника. Окремо варто відзначити досвід з оптимізації продуктивності веб-додатків. Практичні навички з аналізу швидкодії, виявлення вузьких місць, кешування даних та оптимізації запитів до бази значно розширили моє розуміння того, як створювати ефективні та масштабовані рішення. Цей проект став для мене не лише можливістю закріпити теоретичні знання, а й справжньою школою професійного зростання, що сформувала комплексне розуміння всіх аспектів сучасної веб-розробки. Отриманий досвід безумовно стане міцною основою для моєї подальшої професійної діяльності у сфері інформаційних технологій.

Робота дозволила поглибити знання з архітектури програмних систем та відпрацювати навички командного проектування. Отримані результати демонструють готовність до вирішення складних завдань у галузі веб-розробки та можуть стати основою для подальших досліджень у рамках дипломного проекту.

Напрямки для подальшого вдосконалення системи:
Розроблений веб-додаток має значний потенціал для подальшого розвитку та вдосконалення. Одним із перспективних напрямків є реалізація додаткових модулів аналітики, які дозволять отримувати детальну статистику використання системи, аналізувати поведінку користувачів та ухвалювати обґрунтовані бізнес-рішення на основі зібраних даних. Іншим важливим аспектом є підвищення продуктивності роботи системи за рахунок оптимізації запитів до

бази даних, впровадження кешування часто використовуваних даних та використання більш ефективних алгоритмів обробки інформації. Також варто розглянути можливість розширення функціональності мобільної версії додатка, оскільки зростаюча кількість користувачів взаємодіє з системами саме через мобільні пристрої. Це передбачає не лише адаптацію інтерфейсу, а й реалізацію додаткових функцій, специфічних для мобільних платформ, таких як push-сповіщення або офлайн-режим. Ще одним важливим напрямком розвитку є інтеграція з додатковими зовнішніми сервісами, що дозволить розширити функціональність системи без необхідності розробки власних рішень для кожного окремого завдання. Наприклад, підключення платіжних систем, сервісів електронної пошти, соціальних мереж або хмарних сховищ даних може значно підвищити зручність користування та привабливість продукту для кінцевих користувачів. Таким чином, система має значний потенціал для подальшого вдосконалення, що дозволить не лише підвищити її ефективність, а й забезпечити конкурентоспроможність на ринку. Реалізація цих напрямків може стати основою для майбутніх оновлень та розширень функціоналу.

Загалом, виконана робота підтвердила ефективність обраних підходів та технологій для вирішення поставлених завдань.

Список використаних джерел:

1. Крамаренко, С. С. Основи веб-розробки: HTML, CSS, JavaScript. — Київ: Видавництво Ліра-К, 2021.
2. Яскевич, Владислав Олександрович (2023) *JavaScriptбібліотека React Навчальний посібник для студентів спеціальності Комп'ютерні науки* Київський університет імені Бориса Грінченка, Україна, Київ. https://elibrary.kubg.edu.ua/id/eprint/48587/
3. І.В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
4. Братусь, А. І. Системний аналіз та проектування інформаційних систем. — Львів: Видавництво ЛНУ, 2020.
5. Freeman, E., & Robson, E. Head First Web Design. — O'Reilly Media, 2020.
6. W3Schools. https://www.w3schools.com — Довідковий ресурс з HTML, CSS, JavaScript, React.
7. Mozilla Developer Network (MDN). https://developer.mozilla.org — Документація і гіди з веб-технологій.
8. React.js — офіційна документація: https://reactjs.org
9. Vue.js — офіційна документація: https://vuejs.org
10. Bootstrap — офіційна документація: https://getbootstrap.com
11. Tailwind CSS — офіційна документація: https://tailwindcss.com
12. MongoDB — офіційна документація: https://www.mongodb.com/docs
13. MySQL — офіційна документація: https://dev.mysql.com/doc
14. Nielsen Norman Group. UX Design Guidelines: https://www.nngroup.com
15. Baymard Institute. E-Commerce UX Research: https://baymard.com
16. Krug, S. Don't Make Me Think: A Common Sense Approach to Web Usability. — New Riders, 2014.
17. Stack Overflow Developer Survey 2024: https://survey.stackoverflow.co/2024 — статистика популярності технологій.
18. Gartner Research. Trends in Web Application Development (2023).