Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтернет — магазину іноваційної техніки “FutureGadget” З використанням засобів HTML5, CSS, JavaScript, MySQL, Python, C

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Данченко Д.Д.
(підпис) (прізвище та ініціали)

Керівник _____ Ясній О.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Данченкові Данилу Денисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи

Керівник роботи Доктор тех. наук, проф. кафедри МН. Ясній О.П.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-47*

2. Термін подання студентом завершеної роботи 29 червня 2024р.

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Перелік елементів змісту (З номерами, через крапку. Примітка: Не достатньо вказати тільки назви розділів!). 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки (якщо такі є). Вступ, 1 Аналіз предметної області та постановка завдання. 1.1 Вступ. 1.2 Аналіз сучасного стану ринку інтернет — магазинів. 1.3 Аналіз предметної Області. 1.4 Постановка завдання. 1.5 Огляд технологій та інструментів. 1.6 Висновки. 2 Проектна частина. Проектування та розробка системи. 2.1 Вступ. 2.2 Проектування архі Тектури та бази даних. 2.3 Розробка клієнтської частини. 2.4 Розробка серверної частини та Реалізація з нею та базою даних зв'зку клієнтської частини. 2.5 Висновки до Розділу 2 3 Практична частина. Реалізація та тестування. 3.1 Тестування системи, види тестувуання, Результати тестів. 3.2. Впровадження та розгортання системи. 3.3. Моніторинг та обслугову Вання. 3.4 Висновки та рекомендації. 4 Безпека життєдіяльності, основи охорони праці. Висновки, Перелік джерел, Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Повний перелік слайдів у презентації, включаючи перший та останній слайди (з номерами, через крапку). 1. Титульний слайд. 2. Мета кваліфікаційної роботи. 3. Комерція — це. 4. Тенденції розвитку інтернет - магазинів. 5. Цільова аудиторія. 6. Аналіз гравців на ринку, Що є потенційними конкурентами. 7. Вимоги до інтернет — магазину. 8. Структура інтернет - магазину. 9.Архітектура "Клієнт — сервер — база даних". 10. Проектування бази даних, ER -діаграма. 11. Головна сторінка. 12.Форма для оформлення замовлення. 13. Сторінка Доставки і оплати. 14. Кошик. 15. Висновки. 16. Дякую за увагу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	доц. кафедри МТ. Сенчишин В.С.	12.06.2024	15.06.2024

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про технології системи інтернет — магазину, мікросервісну архітектуру, клієнт — серверну модель, такі інстументи, як MySQL, HTML5, CSS, JavaScript, Python, мова С	31.01.2024-03.02.2024	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	*Виконано*
4.	Виконання дослідження щодо сучасного стану ринку інтернет — магазинів, аналіз конкурентів, аналіз предметної області, постановка завдання, визначення вимог до системи.	07.02.2024-11.02.2024	*Виконано*
	Розроблення архітектури системи, бази даних, HR-діаграми, Розробка користувацького інтерфейсу та серверної частини	11.02.2024-03.06.2024	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та постановка завдання	03.06.2024-05.06.2024	*Виконано*
	»		
6.	Оформлення розділу «Проектна частина”	06.06.2024-08.06.2024	*Виконано*
	»		
7.	Оформлення розділу “Практична частина”	09.06.2024-11.06.2024	*Виконано*
	»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2024-13.06.2024	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	14.06.2024-15.06.2024	*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	*Виконано*
11.	Нормоконтроль	18.06.2024-19.06.2024	*Виконано*
12.	Перевірка на плагіат	20.06.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	21.06.2024	*Виконано*
14.	Захист кваліфікаційної роботи	25.06.2024	

Студент _____ Данченко Д.Д
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ясній О.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтернет — магазину інноваційної техніки Future – Gadget з використанням засобів HTML, CSS, JavaScript, MySQL, Python та C // Кваліфікаційна робота освітнього рівня «Бакалавр» // Данченко Данило Денисович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2024 // С. 77 , рис. –22 , табл. – 4 , кресл. – 16, додат. – 1 , бібліогр. – 35 .

Ключові слова: Інтернет — магазин, веб — розробка, клієнт — серверна архітектура, база даних, HTML, JavaScript, CSS, аналіз даних

Кваліфікаційна робота присвячена дослідженню дослідженню розробки та оптимізації інтернет-магазину інноваційної техніки з використанням сучасних технологій веб-розробки. В першому розділі кваліфікаційної роботи описано теоретичні основи та актуальні тенденції електронної комерції. Висвітлено роль інтернет-магазинів у сучасному економічному середовищі , розглянуто ключові аспекти їхнього функціонування та проаналізовано вимоги до їхнього розвитку та підтримки.

В другому розділі кваліфікаційної роботи розглянуто методи дослідження оптимізації функціональності інтернет-магазину за допомогою модулів та інструментів, подано практичні приклади впровадження оптимальних рішень.

В третьому розділі описано практичні етапи розробки і впровадження інтернет-магазину, проаналізовано результати оптимізації та проведено оцінку їхнього впливу на продуктивність та ефективність бізнес-процесів. Об'єктом дослідження є інтернет-магазин, предметом - методи і технології його оптимізації.

ANNOTATION

Developing the Future-Gadget online store using HTML, CSS, JavaScript, MySQL, Python, and C tools. // Qualification work of the educational level "Bachelor" // Danchenko Danylo // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2024 // P. 77 , fig. - 22 , tabl. - 4 , chair. - 16 , annexes. – 1 , references - 35 .

Keywords: Online store, web development, client-server architecture, database, HTML, JavaScript, CSS, data analysis

The qualification work is dedicated to exploring the development of an innovative technology online store named "Future – Gadget" using HTML, CSS, JavaScript, MySQL, Python, and C.

The goal of the work is to design and implement a robust e-commerce platform capable of showcasing and selling cutting-edge technological products.

The first section of the qualification paper considered the theoretical foundations of web development, emphasizing client-server architecture, database management, and the essential roles of HTML, JavaScript, and CSS in creating dynamic web interfaces.

In the second section of the qualification work, it is considered the practical implementation aspects, detailing the integration of MySQL for database management and the utilization of Python and C for backend functionalities. The development process and challenges encountered during the creation of "Future – Gadget" are analyzed, focusing on ensuring scalability, security, and user experience optimization.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CSS - Cascading Style Sheets (CSS - Каскадні таблиці стилів)

HTML - Hypertext Markup Language (HTML - Мова гіпертекстової розмітки)

JavaScript - JavaScript (JavaScript - JavaScript)

MySQL - MySQL (MySQL - MySQL)

Python - Python (Python - Python)

REST - Representational State Transfer (REST - Передача стану представлення)

SQL - Structured Query Language (SQL - Мова структурованих запитів)

UX - User Experience (UX - Взаємодія користувача)

HTTPS - Hypertext Transfer Protocol Secure (HTTPS - Захищений протокол передачі гіпертексту)

MVC - Model-View-Controller (MVC — Модель-Вид-Контролер)

JWS – Json Web Token.

ЗМІСТ

ВСТУП

Розвиток сучасних технологій та інтернет-комерції створює нові можливості для бізнесу, що дозволяють ефективно залучати клієнтів та збільшувати обсяги продажу. Інтернет-магазини стали невід'ємною частиною повсякденного життя, забезпечуючи зручність, доступність та швидкість покупки товарів. В умовах високої конкуренції на ринку інноваційної техніки виникає потреба у створенні функціональних, безпечних та привабливих для користувачів інтернет-магазинів, здатних задовольнити потреби сучасного споживача.

Актуальність даної роботи зумовлена стрімким зростанням популярності інтернет-магазинів, а також постійно зростаючими вимогами до їх функціональних можливостей та зручності використання. Створення інтернет-магазину інноваційної техніки «Future Gadget» дозволить забезпечити споживачів якісними та сучасними товарами, а також покращити процеси покупки та обслуговування.

Метою даної кваліфікаційної роботи освітнього рівня «бакалавр» є розробка інтернет-магазину інноваційної техніки «future gadget» з використанням сучасних технологій веб-розробки, таких як html5, css, javascript, mysql, python та c. Реалізація даного проекту передбачає створення зручного інтерфейсу для користувачів, забезпечення високої продуктивності та безпеки системи, а також інтеграцію з базою даних для ефективного управління товарними позиціями та замовленнями. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

-Аналіз предметної області та визначення вимог до системи.

-Розробка архітектури інтернет-магазину та проектування бази даних.

-Створення користувацького інтерфейсу з використанням HTML5, CSS та JavaScript.

-Реалізація серверної частини системи з використанням Python та інтеграція з базою даних MySQL.

-Оптимізація критичних компонентів за допомогою мови програмування C.

-Тестування системи та забезпечення її безпечної та стабільної роботи.

-У даній роботі будуть розглянуті сучасні підходи та методи розробки веб-додатків, а також проведено аналіз ефективності використання обраних технологій. Особлива увага буде приділена забезпеченню безпеки даних користувачів та надійності системи.

Розробка інтернет-магазину «Future Gadget» має велике практичне значення, оскільки дозволить забезпечити високу якість обслуговування клієнтів та підвищити конкурентоспроможність компанії на ринку інноваційної техніки. Використання сучасних технологій забезпечить гнучкість та масштабованість системи, що дозволить легко адаптувати її до змінних вимог бізнесу та зростаючих потреб споживачів.

Таким чином, дана сертифікаційна робота сприятиме розвитку навичок у галузі веб-розробки та надасть можливість отримати цінний досвід у створенні сучасних інтернет-магазинів, здатних конкурувати на висококонкурентному ринку інноваційної техніки.

Розробка інтернет-магазину «Future Gadget» також матиме позитивний вплив на маркетингові стратегії компанії. Завдяки можливості збору та аналізу даних про поведінку користувачів на сайті, можна буде розробити більш цілеспрямовані рекламні кампанії та персоналізовані пропозиції для клієнтів. Крім того, інтеграція з соціальними мережами та використання сучасних інструментів аналітики дозволить компанії постійно удосконалювати свої маркетингові підходи, підвищуючи рівень залучення та лояльності споживачів.

Особливу увагу буде приділено мобільній оптимізації інтернет-магазину. Враховуючи зростаючу популярність мобільних пристроїв, важливо забезпечити зручність користування сайтом на смартфонах та планшетах. Адаптивний дизайн дозволить клієнтам легко знаходити та купувати товари, незалежно від типу пристрою, що сприятиме збільшенню продажів та покращенню загального досвіду користувачів.

Не менш важливим аспектом є забезпечення доступності сайту для користувачів з обмеженими можливостями. Впровадження стандартів веб-доступності (WCAG) дозволить зробити сайт більш інклюзивним та привабливим для ширшого кола споживачів, що також позитивно вплине на імідж компанії та її репутацію.

Важливим елементом успішного функціонування інтернет-магазину є система обробки платежів. Інтеграція з різними платіжними системами, такими як PayPal, Visa, MasterCard та іншими, дозволить забезпечити клієнтам зручні та безпечні способи оплати. Захист фінансових даних клієнтів буде забезпечений завдяки використанню сучасних криптографічних методів та протоколів безпеки.

Окремо слід зазначити важливість системи управління запасами та логістики. Впровадження автоматизованої системи відстеження запасів допоможе уникнути проблем з нестачею товарів та знизити витрати на їх зберігання. Система логістики дозволить оптимізувати процес доставки товарів клієнтам, забезпечуючи швидке та надійне обслуговування.

Таким чином, кваліфікаційна робота з розробки інтернет-магазину інноваційної техніки «Future Gadget» охоплює широкий спектр питань, від технічних аспектів створення та налаштування сайту до маркетингових стратегій та забезпечення високого рівня обслуговування клієнтів. Реалізація цього проекту сприятиме розвитку нових знань та навичок у галузі веб-розробки, а також підвищенню конкурентоспроможності компанії на ринку інноваційної техніки.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Вступ

Стрімкий розвиток інформаційних технологій та інтернет-комерції створює нові можливості для бізнесу, які дозволяють ефективно залучати клієнтів і збільшувати обсяги продажу. Інтернет-магазини стали невід'ємною частиною повсякденного життя, забезпечуючи зручність, доступність та швидкість покупки товарів. В умовах високої конкуренції на ринку інноваційної техніки виникає потреба у створенні функціональних, безпечних та привабливих для користувачів інтернет-магазинів, здатних задовольнити потреби сучасного споживача. У цьому розділі буде здійснено детальний аналіз предметної області, визначено основні вимоги до системи та сформульовано завдання для розробки інтернет-магазину інноваційної техніки «Future Gadget».

1.2 Аналіз сучасного стану ринку інтернет - магазинів

Ринок інтернет-магазинів постійно зростає, зокрема завдяки зручності покупок в онлайні, широкому асортименту товарів та можливості порівнювати ціни та характеристики. За даними аналітичних досліджень, обсяг продажів через інтернет-магазини щороку збільшується на 15-20%. Особливо це стосується ринку інноваційної техніки, де споживачі очікують високої якості продуктів та сервісу. Основними гравцями на цьому ринку є Amazon, eBay, Best Buy та інші великі платформи, які пропонують широкий асортимент технічних новинок та аксесуарів.

Також, ринок інтернет – магазинів дуже швидко розвивається завдяки розвитку інформаційних технологій та збільшення популярності електронної комерції, тобто торгових та бізнес – процесів із використанням мережі.

1.2.1 Основні гравці на ринку

Основними гравцями на ринку інтернет-магазинів є великі міжнародні платформи, такі як:

-Amazon: Найбільший інтернет-магазин у світі, який пропонує широкий асортимент товарів, швидку доставку та високий рівень обслуговування клієнтів.

-eBay: Аукціонний сайт, який також надає можливість прямого продажу товарів від продавця до покупця.

-Alibaba: Китайський гігант електронної комерції, який включає такі платформи, як Taobao та Tmall, орієнтовані на споживачів та бізнес-користувачів відповідно.

-Walmart: Великий роздрібний торговець, який активно розвиває свій онлайн-бізнес для конкуренції з Amazon.

Amazon - найбільший інтернет-магазин у світі, який заснував Джефф Безос у 1994 році. Зараз компанія належить до найцінніших брендів у світі і пропонує широкий асортимент товарів, включаючи книги, електроніку, одяг, харчові продукти та багато іншого. Основна місія Amazon полягає в тому, щоб надавати клієнтам найширший вибір товарів та найкращий сервіс. Компанія прагне забезпечити зручність покупок, пропонуючи швидку доставку через свою мережу складів та логістичних служб.

Amazon також відомий своєю платформою Amazon Web Services (AWS), яка є провідним провайдером хмарних обчислень у світі. AWS забезпечує широкий спектр послуг у сфері обчислень, зберігання даних та інфраструктури для бізнесу та веб-розробки. Ця додаткова галузь бізнесу Amazon є значним джерелом доходу і сприяє фінансовій стабільності компанії.

На Рисунку 1.1 зображено логотип компанії Amazon.

Рисунок 1.1 – Логотип компанії Amazon

eBay — це світовий онлайн-аукціон та платформа для покупки і продажу товарів між користувачами. Заснований у 1995 році, eBay спеціалізується на аукціонних продажах, де користувачі можуть ставити свої ціни на товари або придбати їх за фіксовану ціну (Buy It Now). Платформа є популярною серед частних продавців, які можуть виставляти товари на продаж безпосередньо для широкої аудиторії.

Основна концепція eBay полягає в створенні глобального ринку, де покупці та продавці з усього світу можуть здійснювати торгівлю без посередників. Платформа надає можливість для участі в аукціонах, де товари можуть продаватися згідно з правилами конкурентної торгівлі. eBay також надає різні інструменти для підтримки та захисту покупців і продавців, зокрема системи оцінювання користувачів та рішень у суперечках.

Логотип платформи зображено на рисунку 1.2.

Рисунок 1.2 – Логотип платформи eBay

Ці дві компанії, Amazon і eBay, представляють різні підходи до електронної комерції: Amazon спеціалізується на прямих продажах через

власну платформу, в той час як eBay відкритий для різних форм торгівлі, зокрема аукціонних продажів. Обидві платформи займають важливе місце на світовому ринку електронної комерції і є ключовими гравцями у своїх сегментах.

1.2.2 Тенденції розвитку ринку

Серед основних тенденцій розвитку ринку інтернет-магазинів можна виділити наступні:

Мобільна комерція: Зростання використання мобільних пристроїв для здійснення покупок. За даними eMarketer, до 2023 року понад 72.9% продажів у сфері електронної комерції будуть здійснюватися через мобільні пристрої. Це вимагає від інтернет-магазинів адаптивного дизайну та оптимізації для мобільних платформ.

Соціальна комерція: Використання соціальних мереж як платформи для продажу товарів. Платформи, такі як Instagram, Facebook та Pinterest, впроваджують інструменти для покупок безпосередньо зі сторінок соцмереж.

Персоналізація: Інтернет-магазини використовують дані про поведінку користувачів для створення персоналізованих рекомендацій та пропозицій. Це підвищує рівень задоволеності клієнтів та збільшує обсяги продажів.

Використання штучного інтелекту та машинного навчання: Інструменти AI та ML застосовуються для покращення пошукових алгоритмів, прогнозування попиту, управління запасами та надання підтримки клієнтам через чат-боти.

Швидка доставка: Споживачі все більше цінують швидкість доставки, що спонукає інтернет-магазини інвестувати в логістичні рішення та складські технології. Amazon Prime є прикладом успішної програми швидкої доставки, яка стала еталоном для інших компаній.

Екологічність: Зростає увага до екологічної відповідальності, тому інтернет-магазини шукають способи зменшення вуглецевого сліду,

використання екологічних упаковок та пропонують товари від сталих виробників.

Ринок інтернет-магазинів інноваційної техніки стрімко зростає, що обумовлено підвищеним попитом на новітні технологічні рішення. Сучасні технології та глобалізація дозволяють споживачам купувати товари з усього світу, що збільшує конкуренцію серед продавців та вимагає від них постійного вдосконалення своїх платформ. За даними дослідницьких агентств, світовий ринок інноваційної техніки демонструє стабільне зростання на рівні 15-20% щорічно. Основними факторами, що сприяють цьому зростанню, є розвиток нових технологій, таких як інтернет речей (IoT), штучний інтелект (AI), віртуальна та доповнена реальність (VR/AR).

1.2.3 Переваги та недоліки(виклики) інтернет - магазинів

До основних переваг інтернет-магазинів належать:

-Зручність та доступність: Покупці можуть здійснювати покупки у будь-який час та з будь-якого місця, де є доступ до інтернету.

-Широкий асортимент: Інтернет-магазини часто мають більший вибір товарів порівняно з фізичними магазинами.

-Конкурентні ціни: Споживачі можуть легко порівнювати ціни та обирати найвигідніші пропозиції.

-Інформаційна підтримка: Детальні описи товарів, відгуки та рейтинги допомагають споживачам приймати обґрунтовані рішення.

Попри численні переваги, інтернет-магазини також стикаються з певними викликами:

-Безпека: Забезпечення захисту персональних даних та фінансових транзакцій є критично важливим завданням.

-Логістика: Організація швидкої та надійної доставки товарів, особливо у віддалені регіони.

-Конкуренція: Велика кількість конкурентів вимагає постійного вдосконалення сервісу та пропозицій.

-Підтримка клієнтів: Забезпечення якісної підтримки для вирішення проблем клієнтів, включаючи швидке оброблення повернень та скарг.

1.3Аналіз предметної області

Аналіз предметної області є важливим етапом у розробці інтернет-магазину, оскільки дозволяє виявити ключові аспекти, які необхідно врахувати під час проектування системи. Основні етапи аналізу включають дослідження ринку, визначення цільової аудиторії та аналіз конкурентів. Дослідження ринку інноваційної техніки включає аналіз попиту на різні види техніки, визначення найбільш популярних категорій товарів та вивчення тенденцій розвитку ринку. Для цього використовуються статистичні дані, звіти аналітичних агентств та інформація від виробників техніки. Інші етапи аналізу предметної області розглянемо більш детально.

1.3.1 Визначення цільової аудиторії

Цільова аудиторія інтернет-магазину «Future Gadget» включає:

Ентузіасти техніки: люди, які цікавляться новітніми технологіями та постійно оновлюють свої гаджети.

Професіонали: спеціалісти, які використовують інноваційну техніку у своїй професійній діяльності.

Молодь: активні користувачі інтернету та мобільних додатків, які цінують зручність онлайн-покупок.

Студенти: Студенти технічних спеціальностей, що ціуавляться інноваційною та робототехнікою.

Допитливі люди, яким просто цікаво дізнатись щось нове.

1.3.2 Аналіз конкурентів

Аналіз конкурентів дозволяє виявити сильні та слабкі сторони існуючих інтернет-магазинів, а також визначити можливості для покращення власного сервісу. Нище наведено основні аспекти, які розглядаються під час аналізу конкурентів.

-*Асортимент товарів:* різноманітність та кількість пропозицій.

-*Ціноутворення:* порівняння цін на аналогічні товари.

-*Функціональність сайту:* зручність користування, дизайн, навігація.

-*Обслуговування клієнтів:* швидкість та якість підтримки/

-*Маркетинг:* рекламні кампанії, програми лояльності, акції та знижки.

Одним з ключових факторів успіху інтернет-магазину є розуміння потреб споживачів та їх купівельної поведінки. Відповідно до опитувань та аналітичних досліджень, покупці інноваційної техніки найбільше цінують зручність навігації сайтом, швидкість доставки, наявність відгуків про товари та можливість порівняння характеристик. У Таблиці 1.1 показано результати опитування споживачів щодо важливих факторів при виборі інтернет-магазину.

Таблиця 1.1 – Важливі фактори при виборі інтернет - магазину

Фактор	**Відсоток респондентів (%)**
Зручність навігації сайтом	45%
Швидкість доставки	30%
Наявність відгуків	15%
Можливість порівняння товарів	10%

Аналіз основних конкурентів, таких як Amazon, AliExpress та Best Buy, дозволяє виділити кілька ключових компонентів успішного інтернет-магазину: широкий асортимент товарів, конкурентні ціни, зручний інтерфейс та високий

рівень обслуговування клієнтів. Важливою також є інтеграція з соціальними мережами та використання маркетингових інструментів для залучення нових клієнтів.

1.4 Постановка завдання

У цьому підрозділі детально розглянемо основні завдання, які необхідно виконати для успішної реалізації проекту інтернет-магазину інноваційної техніки «Future Gadget».

1.4.1 Вимоги до системи

Розробка інтернет-магазину інноваційної техніки «Future Gadget» вимагає чіткого визначення вимог до системи, що дозволить створити продукт, який задовольняє потреби користувачів та відповідає сучасним стандартам. На основі проведеного аналізу предметної області визначено основні вимоги до інтернет-магазину «Future Gadget». Вимоги поділяються на функціональні та нефункціональні.

Функціональні вимоги визначають основні можливості системи, необхідні для забезпечення її працездатності:

-*Каталог товарів:* забезпечення можливості перегляду та пошуку товарів за різними категоріями та фільтрами.

-*Кошик та оформлення замовлення:* зручний інтерфейс для додавання товарів до кошика, оформлення замовлення та вибору способу оплати.

-*Реєстрація та авторизація користувачів:* можливість створення облікового запису, входу в систему та управління персональними даними.

-*Система обробки платежів:* інтеграція з платіжними системами для безпечної обробки транзакцій.

-*Особистий кабінет:* надання користувачам можливості переглядати історію замовлень, відстежувати стан поточних замовлень та управляти налаштуваннями облікового запису.

-*Підтримка клієнтів:* інтеграція з системою підтримки клієнтів для оперативного вирішення питань та проблем.

Нефункціональні вимоги визначають якісні характеристики системи:

-*Продуктивність:* забезпечення швидкої роботи сайту навіть при високому навантаженні.

-*Безпека:* захист персональних даних користувачів та фінансових транзакцій.

-*Масштабованість:* можливість розширення функціональності та підключення нових модулів без значних змін в архітектурі системи.

-*Доступність:* забезпечення безперебійної роботи сайту з мінімальним часом простою.

-*Юзабіліті:* зручність використання сайту для різних категорій користувачів, включаючи мобільну оптимізацію та доступність для людей з обмеженими можливостями.

1.4.2 Аналіз ринку та визначення цільової аудиторії

Аналіз ринку є першочерговим завданням, яке передбачає вивчення поточних тенденцій у сфері інноваційної техніки, дослідження конкурентного середовища та визначення найперспективніших категорій товарів. Цей етап дозволяє зрозуміти, які продукти користуються найбільшим попитом, які новинки з'являються на ринку, та які маркетингові стратегії використовують конкуренти. Визначення цільової аудиторії є критично важливим для розробки ефективної стратегії просування. Основними групами цільової аудиторії можуть бути ентузіасти техніки, професіонали, що використовують техніку у своїй роботі, та молодь, яка активно користується інтернетом для здійснення покупок.

1.4.3 Проєктування архітектури системи та бази даних

Наступним етапом є проектування архітектури системи та бази даних. Це включає розробку структурної схеми інтернет-магазину, визначення основних компонентів системи та їх взаємодії. Архітектура системи повинна забезпечувати високу продуктивність, надійність та масштабованість. База даних повинна бути спроектована таким чином, щоб ефективно зберігати та обробляти інформацію про товари, замовлення та користувачів. Визначення основних таблиць, зв’язків між ними та оптимізація структури даних є ключовими аспектами цього етапу.

1.4.4 Створення користувацького інтерфейсу

Створення користувацького інтерфейсу є важливим аспектом проекту, оскільки саме від зручності та функціональності інтерфейсу залежить задоволеність користувачів. На цьому етапі розробляються візуальні макети сторінок сайту, визначається загальний дизайн та навігаційна структура. Впровадження інтерфейсу здійснюється за допомогою сучасних технологій веб-розробки, таких як HTML5, CSS та JavaScript. Важливо забезпечити адаптивний дизайн, щоб сайт коректно відображався на різних пристроях, включаючи комп’ютери, планшети та смартфони.

1.4.5 Реалізація серверної частини системи

Реалізація серверної частини системи передбачає розробку серверної логіки та інтеграцію з базою даних. На цьому етапі створюються API для обробки запитів клієнтів, управління даними та забезпечення безпеки системи. Використання мови програмування Python дозволяє створювати ефективні та гнучкі рішення для обробки даних та взаємодії з базою даних MySQL. Важливо

забезпечити високу продуктивність серверної частини, оскільки вона відповідає за обробку великої кількості запитів та управління товарними позиціями.

1.4.6 Оптимізація та тестування системи

Після реалізації основних компонентів системи проводиться їх оптимізація та тестування. Оптимізація продуктивності включає вдосконалення критичних компонентів за допомогою мови програмування C, що дозволяє досягти високої швидкості обробки даних та зниження навантаження на сервер. Тестування системи включає проведення функціонального тестування для перевірки коректності роботи всіх функцій, навантажувального тестування для оцінки здатності системи витримувати великі обсяги трафіку та безпекового тестування для виявлення вразливостей та забезпечення захисту даних користувачів.

1.4.7 Впровадження та підтримка системи

Останнім етапом є впровадження системи та її подальша підтримка. Впровадження передбачає установку системи на сервер, конфігурацію програмного забезпечення та запуск інтернет-магазину у робочому режимі. Підтримка системи включає забезпечення технічної підтримки користувачів, моніторинг роботи системи та регулярне оновлення програмного забезпечення для підвищення його функціональності та безпеки. Забезпечення надійної роботи системи та своєчасне вирішення технічних проблем є ключовими факторами успішної експлуатації інтернет-магазину.

1.5 Огляд технологій та інструментів

Для розробки сучасного інтернет-магазину необхідно використовувати комплекс сучасних технологій та інструментів, що забезпечать високу продуктивність, зручність використання та безпеку системи. Для реалізації виконання поставленого завдання даної роботи, будемо використовувати такі інструменти, як: HTML5, CSS, JavaScript, MySQL, Python та мова C. Кожна група цих інструментів та технологій відіграє певну окрему роль та виконує певну функцію у нашому проєкті. У пункті 1.5 розглянемо кожну мову програмування більш детально та проведемо детальний аналіз технічних характеристик інструментів.

HTML5 є основною мовою розмітки, що використовується для створення структури веб-сторінок. Вона забезпечує семантичну розмітку, що покращує SEO-оптимізацію та доступність контенту для пошукових систем. На Рисунку 1.3 показано приклад структури HTML5 документа.

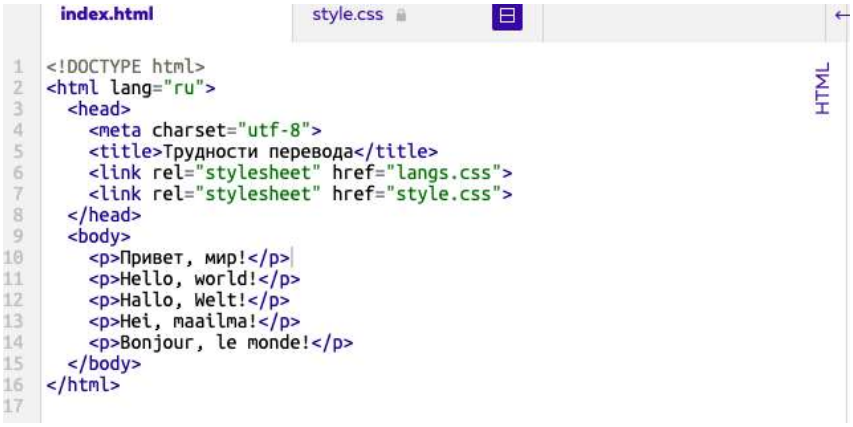

Рисунок 1.3 – Приклад структури HTML5 документу

CSS використовується для стилізації та оформлення інтерфейсу користувача, дозволяючи створювати адаптивний дизайн. Використання Flexbox та Grid Layout забезпечує гнучке розташування елементів на сторінці. На Рисунку 1.4 зображено приклад коду CSS файлу, за допомогою якого

прописана вся стилістика елементів веб — сторінки (таблиці шрифтів, кольори фону і т.п.).

Рисунок 1.4 – CSS- файл та приклад оформлення стилів вмісту веб-сторінки за допомогою CSS коду.

JavaScript забезпечує динамічність та інтерактивність веб-сторінок. Використання фреймворків, таких як React або Vue.js, дозволяє створювати складні та високопродуктивні користувацькі інтерфейси. У Таблиці 1.2 наведено порівняння популярних JavaScript фреймворків.

Таблиця 1.2 – Порівняння популярних JavaScript фреймворків

Фреймворк	**Основні переваги**	**Недоліки**
React	Висока продуктивність, гнучкість	Складність налаштування
Vue.js	Простота вивчення, гнучкість	Менша підтримка з боку спільноти
Angular	Повний набір інструментів	Висока складність навчання

MySQL є реляційною базою даних, що використовується для зберігання інформації про товари, користувачів та замовлення. Вона забезпечує високу швидкість запитів та надійність зберігання даних. На Рисунку 1.5 зображено схему бази даних для інтернет-магазину.

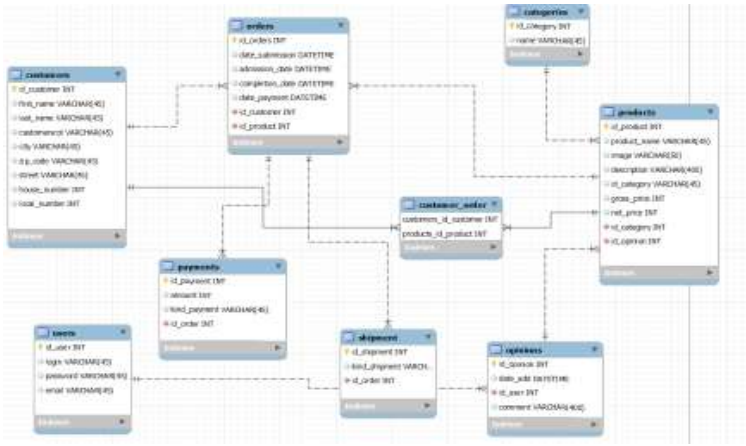

Рисунок 1.5 – Схема “сутнтість — зв’язок” для інтернет — магазину, створена з використанням MySQL

Python використовується для серверної логіки та обробки запитів. Фреймворк Django забезпечує швидку розробку та інтеграцію з базою даних, а також підтримує високу безпеку. На Рисунку 1.5 зображено приклад структури проекту на Django.

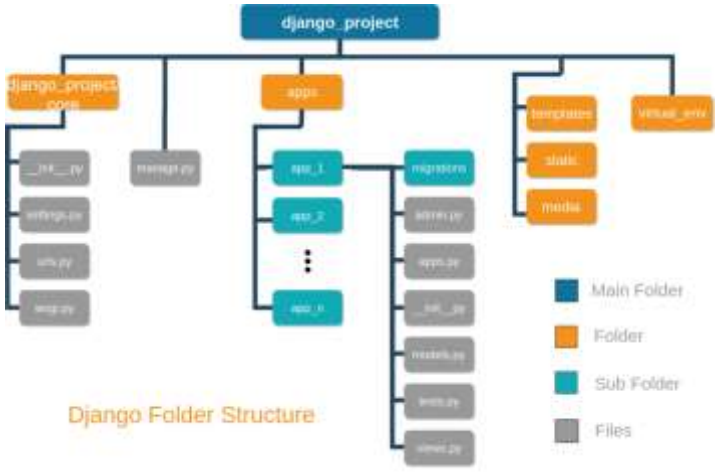

Рисунок 1.6 – Структура проекту на Django

C використовується для оптимізації критичних компонентів системи, що вимагають високої продуктивності. Використання C дозволяє знизити затримки та підвищити швидкість обробки даних. У Таблиці 1.3 наведено порівняння продуктивності різних мов програмування.

Таблиця 1.3 – Порівняння продуктивності різних мов програмування

Мова	**Час виконання (мс)**	**Споживання пам'яті (МБ)**
C	1	10
Python	10	50
JavaScript	5	30

1.6 Висновки

У цьому розділі було детально розглянуто основні завдання, які необхідно виконати для створення інтернет-магазину інноваційної техніки «Future Gadget». Включення аналізу ринку та визначення цільової аудиторії, проектування архітектури системи та бази даних, створення користувацького інтерфейсу, реалізація серверної частини, оптимізація та тестування, а також впровадження та підтримка системи є критично важливими етапами для успішної реалізації проекту. Виконання цих завдань забезпечить створення функціонального, безпечного та зручного інтернет-магазину, здатного задовольнити потреби сучасних споживачів та підвищити конкурентоспроможність компанії на ринку інноваційної техніки.

Це є надзвичайно важливим для того, щоб створити систему, я ка дійсно буде задовольняти вимоги користувачів та замовників, знати всі позитивні сторони магазинів і компаній конкурентів, врахувати всі особливості цільової аудтиторії, та створити магазин, що буде мати унікальну функціональність та перевершувати інших потенційних гравців на ринку.

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА. ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

2.1 Вступ

У цьому розділі буде детально описано процес проектування та розробки інтернет-магазину інноваційної техніки «Future Gadget». Ми розглянемо всі ключові етапи створення системи, починаючи з розробки архітектури і проектування бази даних та закінчуючи впровадженням та підтримкою готового продукту. Особлива увага буде приділена вибору технологій, проектуванню архітектури та бази даних, створенню зручного користувацького інтерфейсу, реалізації серверної частини та оптимізації системи для забезпечення високої продуктивності та безпеки. Кожен етап буде проілюстрований відповідними схемами, таблицями та лістингами коду, що дозволить детально показати всі аспекти розробки та впровадження інтернет-магазину.

2.2 Проектування архітектури системи та бази даних

Проектування архітектури системи та бази даних є ключовим етапом у розробці інтернет-магазину «Future Gadget». Від правильного вибору архітектури залежить не лише продуктивність і масштабованість системи, але й її здатність задовольняти потреби користувачів і забезпечувати високий рівень безпеки даних.

Архітектура інтернет-магазину базується на клієнт-серверній моделі, де основні компоненти системи розділені на серверну і клієнтську частини. Серверна частина відповідає за обробку запитів користувачів, управління даними, забезпечення безпеки та виконання бізнес-логіки. Клієнтська частина представляє собою інтерфейс користувача, який взаємодіє з сервером через API. Така архітектура дозволяє розподілити навантаження між клієнтськими

пристроями і сервером, що забезпечує високу продуктивність і надійність системи.

Для розробки клієнтської частини обрані технології HTML5, CSS і JavaScript. HTML5 забезпечує структурування вмісту веб-сторінок, CSS використовується для стилізації і надання естетичного вигляду, а JavaScript дозволяє реалізувати інтерактивні елементи та динамічний вміст. Використання цих технологій дозволяє створити адаптивний дизайн, який коректно відображатиметься на різних пристроях, включаючи комп’ютери, планшети та смартфони.

Серверна частина системи реалізується за допомогою мови програмування Python, яка відома своєю гнучкістю і багатофункціональністю. Python забезпечує ефективну обробку запитів і інтеграцію з базою даних MySQL. Крім того, для підвищення продуктивності критичних компонентів системи використовуються модулі, написані на мові програмування C. Це дозволяє значно збільшити швидкість обробки даних і зменшити навантаження на сервер.

Одним із найважливіших аспектів проектування системи є безпека. Забезпечення безпеки даних користувачів і захист від можливих атак є пріоритетним завданням. Для цього використовуються сучасні методи шифрування, багаторівнева автентифікація користувачів та захист від SQL-ін'єкцій. Особлива увага приділяється також забезпеченню захисту персональних даних користувачів відповідно до міжнародних стандартів і нормативних вимог.

Проектування бази даних є невід'ємною частиною архітектури системи. База даних повинна забезпечувати зберігання та ефективне управління великими обсягами інформації про товари, користувачів і замовлення. Для цього обрана реляційна база даних MySQL, яка забезпечує високу продуктивність і надійність. Структура бази даних включає основні таблиці для зберігання інформації про продукти, категорії, користувачів, замовлення та

оплати. Всі таблиці взаємопов’язані між собою за допомогою первинних і зовнішніх ключів, що забезпечує цілісність даних.

На етапі проектування бази даних створюється ER-діаграма, яка відображає основні сутності та зв’язки між ними. Це дозволяє візуалізувати структуру даних і оптимізувати їх організацію для забезпечення швидкого доступу та обробки. Особлива увага приділяється нормалізації даних для уникнення дублювання та забезпечення цілісності інформації. Крім того, визначаються основні запити SQL, які будуть використовуватися для доступу до даних, їх обробки та оновлення.

Оскільки база даних є критичним компонентом системи, особлива увага приділяється також питанням резервного копіювання і відновлення даних. Регулярне створення резервних копій забезпечує можливість швидкого відновлення системи у разі збоїв або втрати даних. Для цього використовуються як стандартні інструменти MySQL, так і додаткові програмні засоби для автоматизації процесів резервного копіювання.

У проектуванні архітектури системи важливим аспектом є забезпечення масштабованості. Система повинна мати можливість легко адаптуватися до зростаючих вимог бізнесу та збільшення кількості користувачів. Це досягається за рахунок модульної структури архітектури, де кожен компонент може бути розширений або замінений без впливу на інші частини системи. Використання мікросервісної архітектури дозволяє розподілити функціональність системи на окремі сервіси, що взаємодіють між собою через API.

Проектування архітектури системи та бази даних є комплексним і багатоступеневим процесом, який вимагає ретельного аналізу та планування. Від правильного вибору архітектури залежить не лише функціональність та надійність системи, але й її здатність задовольняти потреби користувачів та забезпечувати високий рівень обслуговування. Всі етапи проектування повинні бути задокументовані, а результати тестування – ретельно проаналізовані для забезпечення максимальної якості та стабільності роботи інтернет-магазину «Future Gadget».

2.2.1 Проектування архітектури інтернет — магазину “Future Gadget”

Як було згадано вище, архітектура інтернет-магазину «Future Gadget» базується на клієнт-серверній моделі, яка є однією з найпоширеніших і ефективних у сучасних веб-додатках. Клієнт-серверна модель передбачає розподіл системи на дві основні частини: *клієнтську* та *серверну*.

Клієнтська частина відповідає за інтерфейс користувача і взаємодію з користувачем. Вона реалізована за допомогою технологій HTML5, CSS і JavaScript, які забезпечують створення сучасного та інтерактивного веб-інтерфейсу. HTML5 використовується для структуризації вмісту веб-сторінок, CSS – для їхнього стилізування та забезпечення привабливого вигляду, а JavaScript – для додавання динамічної поведінки та інтерактивних елементів.

Основними компонентами клієнтської частини є:

-*Інтерфейс користувача:* Включає головну сторінку, сторінки каталогу товарів, сторінку товару, кошик, сторінку оформлення замовлення та профіль користувача.

-*Взаємодія з сервером:* Реалізована через AJAX-запити, що дозволяє динамічно оновлювати вміст сторінки без повного її перезавантаження.

-*Адаптивний дизайн:* Забезпечує коректне відображення інтерфейсу на різних пристроях, включаючи комп’ютери, планшети та смартфони.

Серверна частина відповідає за обробку запитів від клієнтської частини, управління даними та виконання бізнес-логіки. Вона реалізована за допомогою мови програмування Python, яка забезпечує високу продуктивність і гнучкість.

Серверна частина включає такі компоненти:

-*Веб-сервер:* Приймає HTTP-запити від клієнтської частини і надсилає відповідні відповіді.

-*Контролери та маршрутизація:* Визначають, які дії виконувати у відповідь на конкретні запити.

-*Модулі обробки даних:* Відповідають за взаємодію з базою даних, обробку даних та їхню валідацію.

-*Модулі безпеки:* Забезпечують автентифікацію та авторизацію користувачів, шифрування даних та захист від атак.

На Рисунку 2.1 зображено детальну схему архітектури системи на базі клієнт — серверної моделі.

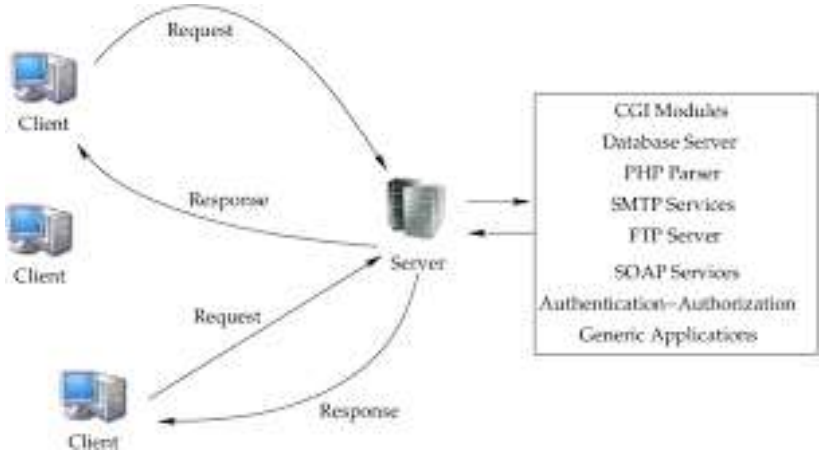

Рисунок 2.1 – Клієнт - серверна модель

Масштабованість є важливим аспектом архітектури інтернет-магазину, оскільки система повинна мати можливість адаптуватися до зростаючих навантажень і збільшення кількості користувачів. Забезпечення масштабованості досягається кількома способами.

Горизонтальне масштабування передбачає додавання нових серверів до існуючої інфраструктури. Це дозволяє розподілити навантаження між кількома серверами, забезпечуючи стабільну роботу системи навіть при значному збільшенні кількості запитів. Для реалізації горизонтального масштабування використовуються балансувальники навантаження, які розподіляють запити між серверами таким чином, щоб кожен сервер отримував оптимальне навантаження.

Вертикальне масштабування передбачає збільшення ресурсів одного сервера (процесорної потужності, оперативної пам'яті, об'єму дискового простору). Це дозволяє підвищити продуктивність сервера і обробляти більшу кількість запитів. Проте, вертикальне масштабування має обмеження, оскільки ресурси одного сервера можуть бути обмежені.

Кешування є важливим методом підвищення продуктивності системи. Зберігання часто запитуваних даних у кеші дозволяє значно зменшити навантаження на сервер і базу даних, прискорюючи обробку запитів. Використання кеш-сервісів, таких як Redis або Memcached, дозволяє ефективно реалізувати цей підхід.

Мікросервісна архітектура є сучасним підходом до розробки великих і складних систем, що забезпечує їхню гнучкість, масштабованість і простоту підтримки. У мікросервісній архітектурі функціональність системи розподіляється між незалежними сервісами, кожен з яких виконує окрему бізнес-логіку і взаємодіє з іншими сервісами через добре визначені API. Приклад мікросервісної архітектури зображено на Рисунку 2.2.

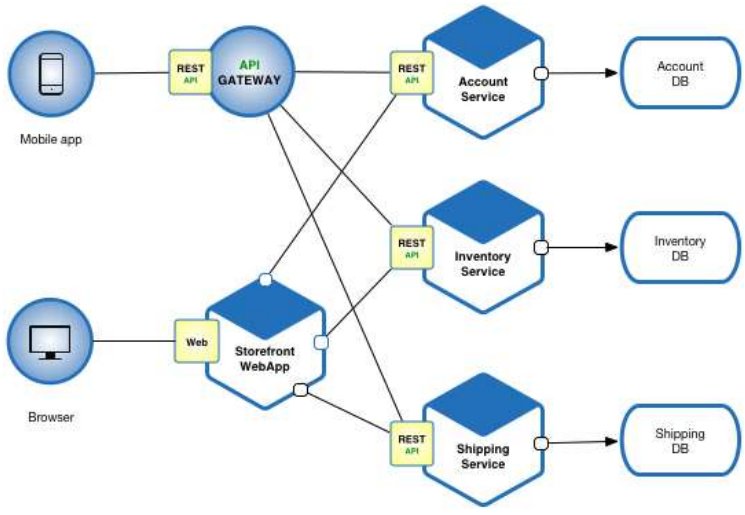

Рисунок 2.2 – Мікросервісна архітектура

Основні принципи мікросервісної архітектури:

-Модульність**:** Кожен мікросервіс відповідає за конкретну функціональність, наприклад, управління користувачами, обробка замовлень, управління товарами тощо. Це дозволяє легко змінювати або замінювати окремі частини системи без впливу на інші компоненти.

-Інтерфейси API: Мікросервіси взаємодіють між собою через API, що забезпечує чітко визначені контракти для обміну даними. Це полегшує інтеграцію нових сервісів і забезпечує незалежність від конкретних реалізацій сервісів.

-Незалежне розгортання: Кожен мікросервіс може бути розгорнутий і масштабований незалежно від інших, що дозволяє більш гнучко управляти ресурсами і забезпечувати безперервність роботи системи.

-Технологічна різноманітність**:** Мікросервіси можуть бути реалізовані на різних мовах програмування і з використанням різних технологій, що дозволяє вибирати найбільш підходящі інструменти для кожного конкретного завдання.

Проаналізувавши всі принципи, характеристики та переваги мікросервісної архітектури, було вирішено здійснити її впровадження в наш проект.

Для інтернет-магазину «Future Gadget» мікросервісна архітектура дозволяє створити гнучку і масштабовану систему, яка легко адаптується до змінних вимог бізнесу. Основні мікросервіси, які можуть бути виділені в системі, включають:

-Сервіс користувачів**:** Відповідає за реєстрацію, автентифікацію та управління профілями користувачів.

-Сервіс товарів: Управляє інформацією про товари, категорії, ціни та наявність на складі.

-Сервіс замовлень: Обробляє замовлення, управляє кошиком покупок та опрацюванням платежів.

-Сервіс аналітики**:** Збирає і аналізує дані про продажі, поведінку користувачів та ефективність маркетингових кампаній.

2.2.2 Проектування бази даних

Проектування бази даних є критичним етапом розробки програмної системи, особливо для інтернет-магазину, такого як Future Gadget. Вибір відповідної системи управління базами даних (СУБД) є важливим рішенням, яке впливає на продуктивність, масштабованість та надійність системи. Ми обрали реляційну базу даних MySQL з кількох вагомих причин.

По-перше, MySQL є однією з найбільш популярних і широко використовуваних реляційних СУБД у світі. Це забезпечує широку підтримку, велику кількість доступних ресурсів та активну спільноту розробників, що спрощує вирішення будь-яких технічних проблем, які можуть виникнути. MySQL є відкритим програмним забезпеченням, що дозволяє уникнути ліцензійних витрат та забезпечує гнучкість у налаштуванні та використанні. Крім того, MySQL добре інтегрується з різними веб-технологіями, такими як PHP, що є важливим для веб-орієнтованого проекту, як наш інтернет-магазин.

Особлива увага у процесі проектування бази даних приділяється нормалізації даних. Нормалізація даних є процесом організації полів і таблиць бази даних для зменшення дублювання і забезпечення цілісності даних. Завдяки нормалізації, ми розбиваємо великі таблиці на менші, які є більш керованими і логічно пов'язаними, що дозволяє уникнути аномалій при вставці, оновленні та видаленні даних. Це також покращує продуктивність бази даних, оскільки зменшується кількість дублюючих записів, що потребує менше простору для зберігання і полегшує пошук необхідної інформації.

Не менш важливими аспектами проектування бази даних є питання резервного копіювання та відновлення даних. У випадку збою системи, втрати даних або будь-яких інших непередбачених обставин, можливість швидкого і надійного відновлення даних є критично важливою. Тому ми впроваджуємо стратегію регулярного резервного копіювання бази даних, що включає як повне,

так і інкрементне копіювання. Це дозволяє мінімізувати втрати даних і забезпечує безперервність бізнес-процесів інтернет-магазину.

Забезпечення цілісності даних також є ключовим аспектом нашого проектування. Цілісність даних означає, що дані у базі є точними, повними і узгодженими. Це досягається за допомогою використання обмежень цілісності, таких як первинні ключі, зовнішні ключі, унікальні обмеження та перевірки (CHECK). Наприклад, зовнішні ключі забезпечують цілісність посилань між таблицями, гарантуючи, що запис в одній таблиці має відповідний запис в іншій.

У підсумку, проектування бази даних для інтернет-магазину Future Gadget базується на виборі надійної і потужної СУБД MySQL, яка забезпечує високу продуктивність, масштабованість і гнучкість. Використання методів нормалізації, стратегій резервного копіювання та відновлення, а також забезпечення цілісності даних дозволяє створити стабільну і ефективну систему, яка відповідає всім вимогам сучасної електронної комерції.

Проектування бази даних для інтернет-магазину Future Gadget є важливим етапом, що забезпечує ефективне управління даними та підтримує різноманітні функції магазину, такі як обробка замовлень, управління товарами, взаємодія з клієнтами та аналітика. Нижче розглянемо основні сутності бази даних, зв'язки між ними, їхні атрибути та деталі побудови ER-діаграми.

2.2.2.1 Сутності бази даних, зв’язки між ними та атрибути сутностей

Користувачі (Users)**:**

Attributes: ID користувача (UserID), ім'я (FirstName), прізвище (LastName), електронна пошта (Email), пароль (Password), адреса (Address), телефонний номер (PhoneNumber), дата реєстрації (RegistrationDate).

Опис: Користувачі є основними суб'єктами системи. Вони можуть бути клієнтами або адміністраторами. Кожен користувач має унікальний ID, особисті дані та дані для входу.

Товари (Products)**:**

Attributes: ID товару (ProductID), назва (ProductName), опис (Description), ціна (Price), кількість на складі (StockQuantity), категорія (CategoryID), зображення (ImageURL).

Опис: Товари представляють продукцію, що продається в інтернет-магазині. Важливо відслідковувати кожен товар, його характеристики, кількість та категорію.

Категорії (Categories)**:**

Attributes: ID категорії (CategoryID), назва (CategoryName), опис (Description).

Опис: Категорії дозволяють організувати товари в логічні групи для зручності навігації та пошуку.

Замовлення (Orders)**:**

Attributes: ID замовлення (OrderID), ID користувача (UserID), дата замовлення (OrderDate), статус замовлення (OrderStatus), загальна сума (TotalAmount), адреса доставки (ShippingAddress).

Опис: Замовлення представляють покупки, зроблені користувачами. Кожне замовлення має зв'язок з користувачем, який його створив, та зберігає інформацію про замовлені товари та статус доставки.

Замовлені товари (OrderItems)**:**

Attributes: ID замовлення (OrderID), ID товару (ProductID), кількість (Quantity), ціна за одиницю (UnitPrice).

Опис: Замовлені товари є деталями конкретних замовлень, що дозволяє відслідковувати які саме товари були замовлені і в якій кількості.

Відгуки (Reviews)**:**

Attributes: ID відгуку (ReviewID), ID товару (ProductID), ID користувача (UserID), рейтинг (Rating), коментар (Comment), дата створення (CreationDate).

Опис: Відгуки надають інформацію про думки користувачів щодо товарів, що дозволяє іншим користувачам приймати більш інформовані рішення щодо покупок.

Зв’язки між сутностями в HR – діаграмі будуть такими:

-Користувачі - Замовлення**:** Кожен користувач може мати багато замовлень (один до багатьох).

-Замовлення - Замовлені товари**:** Кожне замовлення може містити багато товарів (один до багатьох).

-Товари - Замовлені товари**:** Кожен товар може бути частиною багатьох замовлень (один до багатьох).

-Категорії - Товари: Кожна категорія може містити багато товарів (один до багатьох).

-Користувачі - Відгуки**:** Кожен користувач може залишити багато відгуків (один до багатьох).

-Товари - Відгуки**:** Кожен товар може мати багато відгуків (один до багатьох).

Також, кожна сутність має атрибути:

-Користувачі**:** UserID, FirstName, LastName, Email, Password, Address, PhoneNumber, RegistrationDate.

-Товари**:** ProductID, ProductName, Description, Price, StockQuantity, CategoryID, ImageURL.

-Категорії: CategoryID, CategoryName, Description.

-Замовлення**:** OrderID, UserID, OrderDate, OrderStatus, TotalAmount, ShippingAddress.

-Замовлені товари**:** OrderID, ProductID, Quantity, UnitPrice.

-Відгуки**:** ReviewID, ProductID, UserID, Rating, Comment, CreationDate.

Усі сутності, зв’язки між ними та атрибути можна подати у вигляді таблиці. Це все нам потрібно для побудови ER діаграми. У ній буде детально описано всі елемени проектованої бази даних

Усі сутності, зв’язки між ними та атрибути, створені під час проектування бази даних можна подати у вигляді таблиці 2.1.

Таблиця 2.1 – Сутності, атрибути та зв’язки

Сутність	Атрибути	Зв’язки
Користувачі	user_id (PK), ім'я, прізвище, email, пароль, адреса, телефон	Зв'язок "один до багатьох" з таблицею "Замовлення"
Продукти	product_id (PK), назва, опис, ціна, категорія, наявність	Зв'язок "один до багатьох" з таблицею "Замовлення_деталі"
Категорії	category_id (PK), назва	Зв'язок "один до багатьох" з таблицею "Продукти"
Замовлення	order_id (PK), user_id (FK), дата_замовлення, статус	Зв'язок "один до багатьох" з таблицею "Замовлення_деталі"
Замовлення_деталі	order_detail_id (PK), order_id (FK), product_id (FK), кількість, ціна	Зв'язок "багато до одного" з таблицями "Замовлення" та "Продукти"
Ролі	role_id (PK), назва	Зв'язок "багато до багатьох" з таблицею "Користувачі" через "Користувачі_Ролі"
Користувачі_Ролі	user_role_id (PK), user_id (FK), role_id (FK)	Зв'язок "багато до одного" з таблицями "Користувачі" та "Ролі"

Важливо детально описати зв'язки між сутностями, оскільки вони визначають, як дані будуть зберігатися та взаємодіяти між собою. Почнемо з відносин між таблицями "Користувачі" та "Замовлення". У нашій системі один користувач може мати багато замовлень, що створює зв'язок типу "один до багатьох". Це означає, що кожен запис у таблиці "Користувачі" може відповідати багатьом записам у таблиці "Замовлення", але кожне замовлення

пов'язане тільки з одним користувачем. Така структура дозволяє нам зберігати історію покупок кожного користувача окремо, що є критично важливим для персоналізації сервісу та аналізу поведінки користувачів.

Наступний зв'язок існує між таблицями "Продукти" та "Замовлення_деталі". Один продукт може бути включений в багато деталей замовлень, що створює зв'язок "один до багатьох". Це означає, що кожен запис у таблиці "Продукти" може бути пов'язаний з багатьма записами в таблиці "Замовлення_деталі", але кожен запис у таблиці "Замовлення_деталі" посилається лише на один продукт. Це дозволяє нам детально відстежувати, які продукти були замовлені в яких кількостях в кожному окремому замовленні, що є важливим для управління запасами та аналізу продажів.

Відносини між таблицями "Категорії" та "Продукти" також є зв'язком типу "один до багатьох". Це означає, що кожна категорія може містити багато продуктів, але кожен продукт належить лише до однієї категорії. Така структура допомагає організувати продукти в логічні групи, що полегшує навігацію по каталогу товарів для користувачів і спрощує управління продуктами для адміністраторів системи.

Таблиці "Замовлення" та "Замовлення_деталі" також пов'язані зв'язком типу "один до багатьох". Одне замовлення може мати багато деталей замовлень, але кожна деталь замовлення належить лише до одного замовлення. Це дозволяє точно відстежувати, які продукти були включені в кожне замовлення, що потрібно для обробки замовлень і підтримки точного обліку товарів.

На рисунку 2.3 подано інтерфейс для створення нової бази даних та можливість перегляду їх списку.

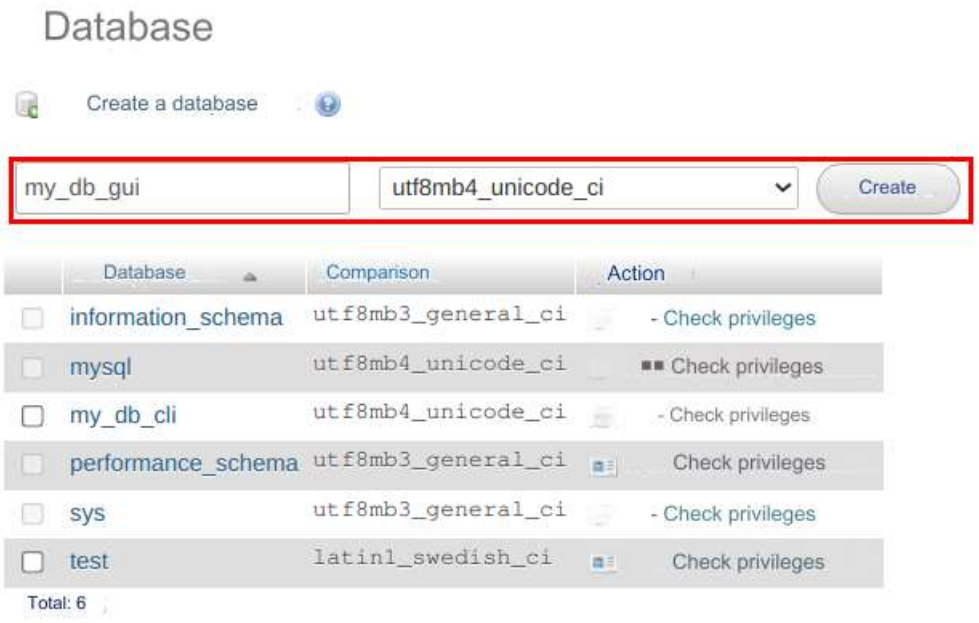

Рисунок 2.3 – Інтерфейс MySQL для створення нової бази даних

Приклад створення таблиці “Користувачі” та вставлення нового користувача в таблицю за допомогою SQL – запиту CREATE TABLE та INSERT INTO зображено на Рисунку 2.4.

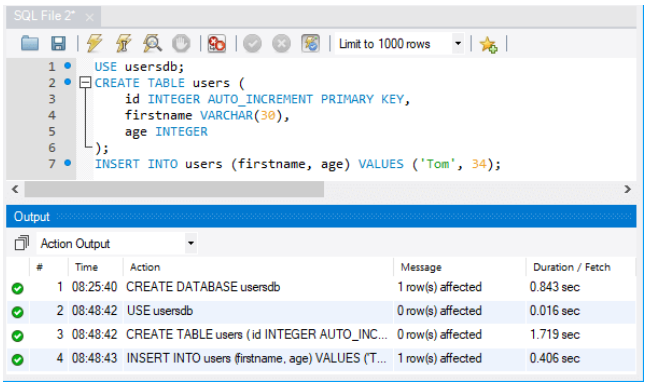

Рисунок 2.4 – Створення таблиці “Користувачі” та вставка нового елемента в середовищі MySQL

Також, на ілюстрації 2.5 подано основне середовище роботи з поточною базою даних та його структура.

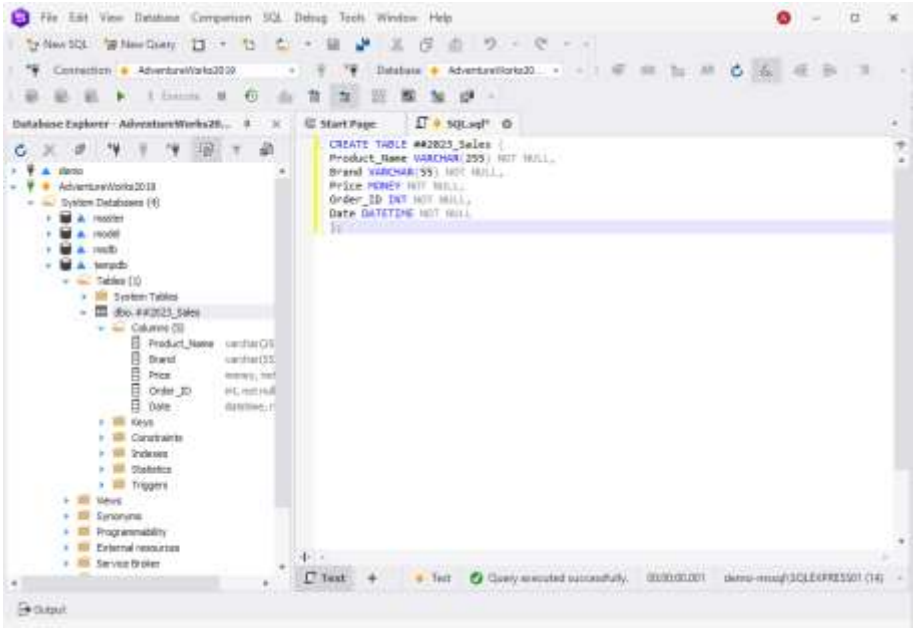

Рисунок 2.5 – Середовище роботи з базою даних в MySQL

Визначення основних таблиць, сутностей та зв’язків між ними, а також первинних та зовнішніх ключів дає нам можливість побудувати ER – діаграму бази даних

2.2.2.2 Побудова ER – діаграми

ER-діаграма надає графічне уявлення про структуру бази даних і дозволяє зрозуміти взаємозв'язки між сутностями. Це важливий інструмент для розробників і допомагає уникнути помилок у проектуванні бази даних.

Етап побудови ER – діаграми передбачає:

-*Визначення сутностей*: Всі сутності, які були описані вище, повинні бути представлені у вигляді блоків.

-*Визначення атрибутів:* Кожен блок повинен містити атрибути, які належать до відповідної сутності.

-*Визначення зв'язків:* Необхідно намалювати зв'язки між сутностями, які відображають відношення "один до багатьох" або "багато до багатьох".

-*Розробка кардинальності:* Кожен зв'язок має бути доповнений кардинальністю, щоб чітко показати кількісні відношення між сутностями.

-*Розглядання обмежень:* Визначення первинних ключів та зовнішніх ключів для підтримки цілісності даних.

-Побудовану ER – діаграму, що відображає схему сутність — зв’язок бази даних нашого інтернет — магазину з первинними та зовнішніми ключами зображено на Рисунку 2.6.

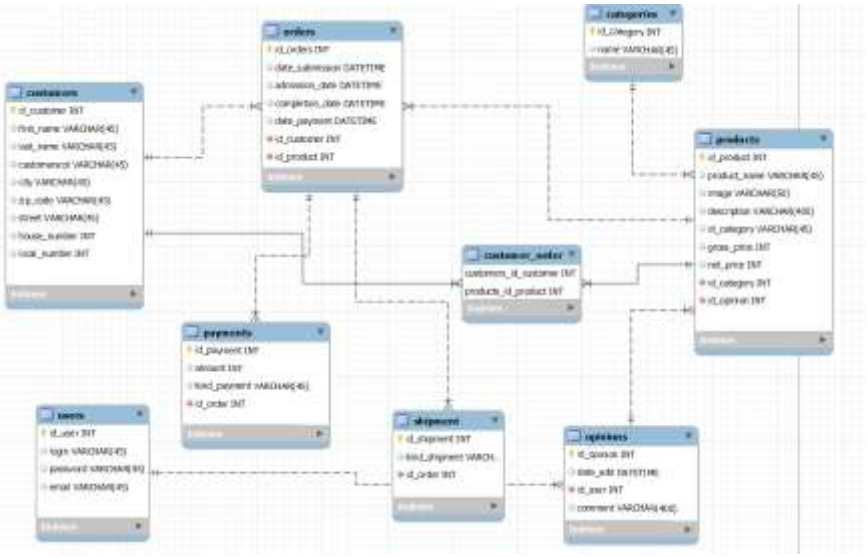

Рисунок 2.6 – Побудована ER – діаграма

При проектуванні бази даних для інтернет-магазину Future Gadget важливим аспектом є використання первинних ключів. Кожна сутність повинна мати унікальний первинний ключ для ідентифікації кожного запису. Це дозволяє чітко визначити кожен окремий запис у таблиці і забезпечує швидкий доступ до даних. Наприклад, у таблиці "Користувачі" це може бути "Ідентифікатор користувача", у таблиці "Продукти" – "Ідентифікатор продукту".

Зовнішні ключі також відіграють критично важливу роль, забезпечуючи відносини між сутностями. Вони дозволяють зв'язувати записи з різних таблиць, що є необхідним для підтримки логічних зв'язків у базі даних. Наприклад, у таблиці "Замовлення" може бути поле "Ідентифікатор користувача", яке є зовнішнім ключем, що посилається на "Ідентифікатор користувача" в таблиці "Користувачі". Це забезпечує зв'язок між замовленням та відповідним користувачем.

Нормалізація бази даних є ще одним важливим етапом проектування. Вона включає процес організації даних, щоб зменшити дублювання даних і забезпечити цілісність. Це досягається шляхом розділення великих таблиць на менші, більш зв'язані таблиці і визначення відносин між ними. Нормалізація сприяє зниженню надлишковості даних і уникненню аномалій при оновленні бази даних.

Індексація є ключовою для підвищення продуктивності запитів до бази даних. Вона передбачає створення індексів на полях, які часто використовуються у пошуку та сортуванні. Індекси дозволяють швидше знаходити і обробляти дані, що значно покращує ефективність роботи бази даних. Наприклад, індексація поля "Назва продукту" в таблиці "Продукти" може значно прискорити пошук товарів за назвою.

Забезпечення цілісності даних є ще одним критичним аспектом проектування бази даних. Це досягається за допомогою обмежень, таких як NOT NULL, UNIQUE, CHECK, і FOREIGN KEY. Обмеження NOT NULL гарантує, що певні поля не можуть залишатися порожніми, UNIQUE забезпечує унікальність значень у стовпці, CHECK дозволяє встановлювати умови для допустимих значень, а FOREIGN KEY підтримує цілісність зв'язків між таблицями. Ці обмеження допомагають зберігати коректність і консистентність даних у базі даних, забезпечуючи надійну роботу системи.

На Рисунку 2.7 проілюстрована схема нормалізації бази даних.

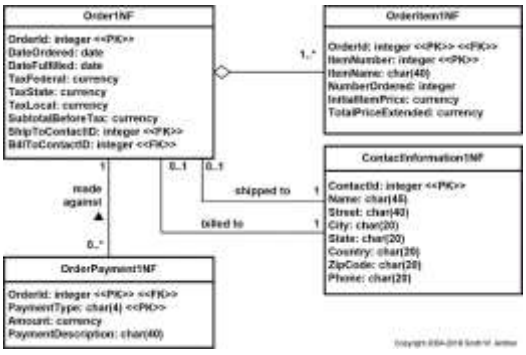

Рисунок 2.7 — Схема нормалізації бази даних

Резервне копіювання та відновлення даних є критично важливими аспектами управління базою даних для інтернет-магазину Future Gadget. Процес резервного копіювання передбачає створення копій даних, які можуть бути використані для відновлення у випадку втрати чи пошкодження оригінальних даних. Регулярне резервне копіювання дозволяє мінімізувати ризик втрати важливої інформації через технічні несправності, помилки користувачів або кібератаки. У нашій системі передбачено автоматичне резервне копіювання бази даних щодня, а також зберігання копій на окремих серверах чи у хмарних сховищах. Це забезпечує високу надійність і доступність даних, навіть у разі фізичних пошкоджень основного серверного обладнання.

Процес відновлення даних з резервних копій є також важливою складовою стратегії забезпечення безпеки та безперервності роботи системи. Відновлення даних передбачає використання останньої доступної резервної копії для повернення бази даних до стану, який існував до втрати даних. Це включає відновлення всіх таблиць, зв'язків і відповідних індексів. Для швидкого та ефективного відновлення даних необхідно ретельно тестувати процес резервного копіювання і відновлення, а також мати чіткий план дій у разі інциденту. Забезпечення можливості швидкого відновлення даних є ключовим для збереження функціональності інтернет-магазину та мінімізації часу простою, що позитивно впливає на задоволення користувачів і загальну стабільність бізнесу.

2.3 Розробка клієнтської частини

Розробка клієнтської частини інтернет-магазину Future Gadget є важливим етапом створення функціонального та привабливого веб-сайту. Основна мета цього етапу – забезпечити зручний інтерфейс для користувачів, що дозволить легко здійснювати покупки, переглядати продукти, управляти замовленнями та багато іншого. Розробка включає створення різних сторінок

сайту за допомогою HTML для структури, CSS для стилізації та JavaScript для інтерактивності.

Вона є критично важливою складовою будь-якого інтернет-магазину, адже саме вона визначає перше враження користувачів та їхній загальний досвід взаємодії з платформою. Користувачі хочуть бачити зручний, естетично привабливий та функціональний інтерфейс, який дозволяє легко та швидко знаходити необхідні товари, додавати їх у кошик, оформлювати замовлення і отримувати інформацію про компанію. Важливо забезпечити, щоб інтерфейс був інтуїтивно зрозумілим, а користувачі могли без труднощів виконувати всі необхідні дії.

HTML5 (HyperText Markup Language) є основою будь-якого веб-сайту. Він використовується для створення структури веб-сторінок, визначаючи заголовки, абзаци, списки, посилання, зображення та інші елементи. HTML надає зміст веб-сторінці, забезпечуючи її базову структуру, яка потім може бути стилізована та покращена за допомогою інших технологій.

CSS (Cascading Style Sheets) дозволяє стилізувати HTML-елементи, визначаючи, як вони мають виглядати на веб-сторінці. За допомогою CSS можна змінювати кольори, шрифти, розміри, відступи, вирівнювання і багато іншого. Це робить веб-сторінки привабливими та узгодженими візуально, забезпечуючи краще сприйняття користувачами та підвищуючи загальну зручність використання.

JavaScript є мовою програмування, яка дозволяє додавати інтерактивність до веб-сторінок. За допомогою JavaScript можна створювати динамічний контент, керувати подіями (такими як кліки або введення даних), змінювати HTML та CSS в реальному часі без необхідності перезавантаження сторінки. JavaScript також забезпечує можливість виконання складних логічних операцій на стороні клієнта, що дозволяє створювати багатофункціональні та інтуїтивні веб-застосунки.

Загалом, правильне використання HTML, CSS та JavaScript є ключовим для створення ефективного, естетично привабливого та зручного інтернет-

магазину. Вони забезпечують міцну основу, на якій будується весь користувацький досвід, гарантуючи, що веб-сайт відповідає потребам та очікуванням своїх користувачів.

2.3.1 Головна сторінка (HomePage)

Головна сторінка є обличчям інтернет-магазину. Вона повинна містити привітання для користувачів, основну навігацію, деякі зображення, а також декілька популярних чи нових товарів. Приклад HTML та CSS коду для головної сторінки наведено в лістингах 2.1 та 2.2.

Лістинг 2.1 – HTML – код Головної сторінки

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
     <meta charset="UTF-8">
     <meta name="viewport" content="width=device-width, initial-
scale=1.0">
     <title>Future Gadget - Home</title>
     <link rel="stylesheet" href="styles.css">
     </head>
     <body>
     <header>
     <h1>Welcome to Future Gadget</h1>
     <nav>
     <ul>
     <li><a href="index.html">Home</a></li>
     <li><a href="products.html">Products</a></li>
     <li><a href="cart.html">Cart</a></li>
     <li><a href="contact.html">Contact</a></li>
     </ul>
     </nav>
     </header>
```

```
<main>
<section id="new-products">
<h2>New Arrivals</h2>
<div class="product-grid">
 <!-- Product items will be inserted here dynamically -->
</div>
</section>
</main>
<script src="scripts.js"></script>
</body>
</html>
```

Лістинг 2.2 – Стилістичне оформлення головної сторінки за допомогою CSS коду

```
/* styles.css */
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
}

header {
background-color: #333;
color: white;
padding: 10px 0;
text-align: center;
}

nav ul {
list-style: none;
padding: 0;
}

nav ul li {
display: inline;
```

```
margin: 0 15px;
}

nav ul li a {
color: white;
text-decoration: none;
}

main {
padding: 20px;
}

.product-grid {
display: flex;
gap: 20px;
flex-wrap: wrap;
}

.product-grid .product-item {
border: 1px solid #ccc;
padding: 10px;
width: 200px;
}
```

Кінцевий варіант вигляду, розміщення всіх елементів, зображень та стилістичного дизайну головної сторінки інтернет — магазину FutureGadget подано на рисунку 2.8.

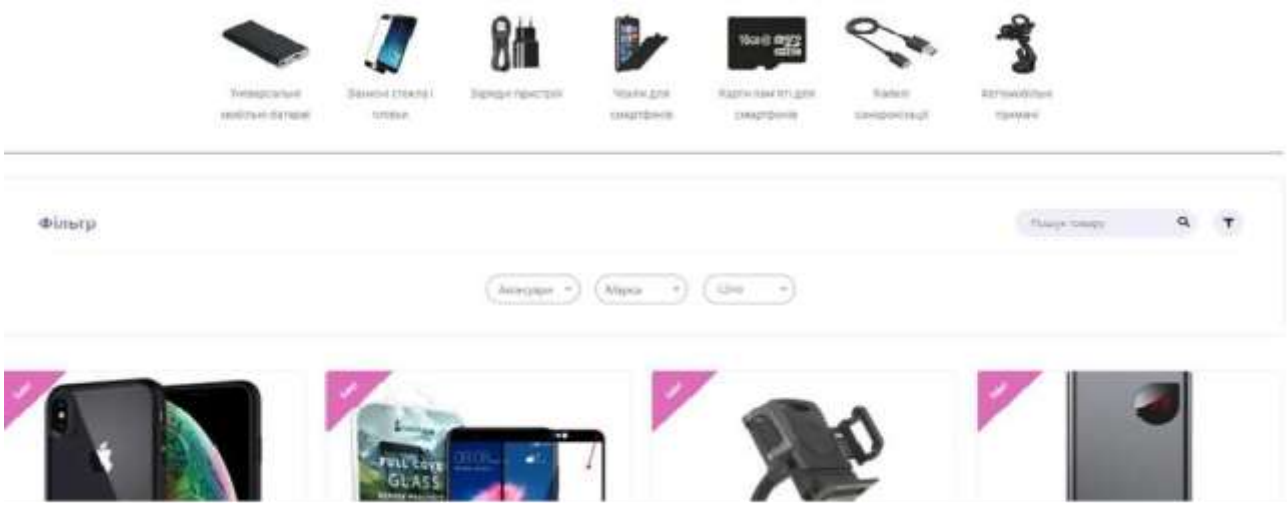

Рисунок 2.8 — Головна сторінка магазину

2.3.2 Сторінка Кошику (Cart Page)

Сторінка кошика дозволяє користувачам переглядати всі товари, які вони додали до свого кошика, змінювати кількість або видаляти товари. Вона також включає кнопку для переходу до оформлення замовлення.

Нище, на лістингах 2.3 та 2.4 наведено організацію елементів та розмітки сторінки за допомогою HTML5 та реалізацію ітерактивних функцій із застосуванням JavaScript.

Лістинг 2.3 – HTML – розмітка

```
<!-- cart.html -->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Future Gadget - Cart</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
```

```
<header>
<h1>Your Cart</h1>
<nav>
<ul>
<li><a href="index.html">Home</a></li>
<li><a href="products.html">Products</a></li>
<li><a href="cart.html">Cart</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
</nav>
</header>
<main>
<section id="cart-items">
<div class="cart-list">
<!-- Cart items will be inserted here dynamically -->
</div>
</section>
<button id="checkout-button">Proceed to Checkout</button>
    </main>
  <script src="scripts.js"></script>
</body>
</html>
```

Лістинг 2.4 – JavaScript

```
// scripts.js (continued)
document.addEventListener('DOMContentLoaded', function() {
// Example of how to add event listeners to "Add to Cart"
buttons
const cart = [];

document.querySelectorAll('button[data-id]').forEach(button
=> {
button.addEventListener('click', function() {
const productId = this.dataset.id;
```

```
    const product = products.find(p => p.id ===
parseInt(productId));
    cart.push(product);
    updateCart();
    });
    });
    function updateCart() {
    const cartList = document.querySelector('.cart-list');
    cartList.innerHTML = '';
    cart.forEach(product => {
    const cartItem = document.createElement('div');
    cartItem.classList.add('cart-item');
    cartItem.innerHTML = `
    <h3>${product.name}</h3>
    <p>Price: $${product.price}</p>
    <button                                    data-id="${product.id}"
class="remove">Remove</button>
    ;
    cartList.appendChild(cartItem);
    });
    }
    });
```

На рисунку 2.9 можна побачити кінцевий вигляд сторінки Кошик.

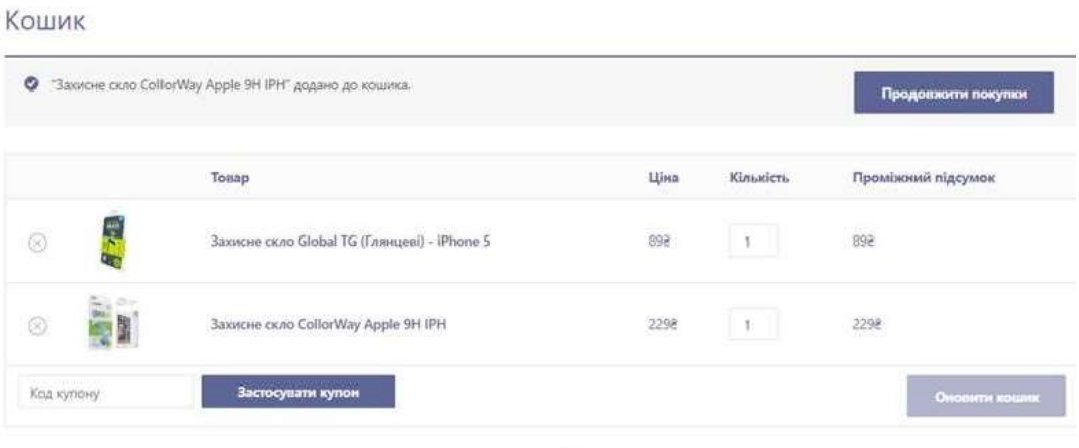

Рисунок 2.9 — Сорінка “Кошик”

2.3.3 Сторінка Доставки та оплати

Сторінка "Доставка та Оплата" забезпечує користувачів інформацією про доступні методи доставки та оплати, а також дозволяє їм вводити деталі адреси доставки. Ця сторінка допомагає користувачам вибрати найзручніший спосіб отримання товару та зручний метод оплати, що значно покращує користувацький досвід і підвищує ймовірність успішного завершення покупки.

Кінцевий вигляд проілюстровано на рисунку 2.10

Рисунок 2.10 — Доставка та оплата

2.3.4 Сторінка Оформлення замовлення

Сторінка "Оформлене Замовлення" служить підтвердженням того, що замовлення було успішно створено та прийнято системою. На цій сторінці користувачі отримують інформацію про номер замовлення, дату оформлення, загальну суму до оплати, обрану адресу доставки та метод оплати. Це дозволяє клієнтам впевнитися, що їхні замовлення зареєстровані правильно, а також надає їм можливість перевірити деталі перед остаточною обробкою та доставкою замовлення.

Сторінки оформлення замовлення та вже оформленого замовлення подані на рисунках 2.11 та 2.12 відповідно.

Рисунок 2.11 — Оформлення замовлення

Рисунок 2.12 — Сторінка оформленого замовлення

Програмні коди реалізації сторінок можна детально переглянути в Додатку А.

Завдяки використанню HTML, CSS та JavaScript, ми забезпечили структуру, візуальну привабливість та інтерактивність веб-сторінок, включаючи основні сторінки, такі як каталог товарів, сторінка кошика, сторінка доставки та оплати, а також сторінка оформленого замовлення. Дотримання принципів зручності використання та естетичного дизайну дозволяє створити інтуїтивно зрозумілий та функціональний інтерфейс, який сприяє підвищенню задоволеності клієнтів і, в кінцевому рахунку, успіху інтернет-магазину.

2.4 Розробка серверної частини та реалізація зв’язку з нею клієнтської частини та бази даних

Розробка серверної частини інтернет-магазину Future Gadget є ключовим аспектом, що забезпечує функціональність, продуктивність і безпеку веб-додатку. Серверна частина відповідає за обробку запитів від клієнтської частини, управління базою даних, виконання бізнес-логіки і забезпечення надійного зв'язку між різними компонентами системи. Для реалізації серверної частини часто використовуються мови програмування та фреймворки, такі як Node.js, Python з Django або Flask, PHP з Laravel, або Java з Spring.

Зв'язок клієнтської частини з серверною і базою даних здійснюється через API (Application Programming Interface). Клієнтська частина надсилає HTTP-запити (GET, POST, PUT, DELETE) до серверу, який обробляє ці запити, виконує необхідні операції з базою даних і повертає відповіді у форматі JSON або XML. Для забезпечення безпеки і надійності передачі даних використовуються такі протоколи, як HTTPS.

Окремим і важливим елементом архітектури серверної частини є використання шаблону MVC (Model-View-Controller). MVC розділяє додаток на три основні компоненти: модель (Model), подання (View) і контролер (Controller). Модель відповідає за роботу з даними і логіку бізнес-процесів, подання – за інтерфейс користувача, а контролер – за обробку запитів і управління потоком даних між моделлю і поданням. Використання MVC в нашому проекті Future Gadget дозволить підвищити модульність, зручність підтримки і тестування коду.

Використання MVC дозволяє розділити відповідальності між різними частинами системи, роблячи її більш організованою і легкою в підтримці. Модель відповідає за роботу з даними і бізнес-логіку, View – за відображення даних, а контролер – за обробку користувацьких запитів і взаємодію між моделлю і поданням. Це забезпечує високу гнучкість і можливість масштабування додатку.

Таким чином, використання MVC архітектури в поєднанні з сучасними технологіями для серверної частини забезпечить надійне і ефективне функціонування інтернет-магазину Future Gadget, зручне управління даними і надійний зв'язок між клієнтською частиною, сервером і базою даних.

Нище можна переглянути список усіх моделей, виглядів та контролерів, які будуть використовуватися у нашій програмній системі.

Моделі:

-User (Користувач)

-Атрибути: id, name, email, password, role. Відповідає за зберігання інформації про користувачів, їх автентифікацію і авторизацію.

-Product (Продукт)

-Атрибути: id, name, price, description, category_id, stock. Зберігає інформацію про товари, які доступні в магазині.

-Category (Категорія)

-Атрибути: id, name, description. Використовується для класифікації продуктів.

-Order (Замовлення**)**

-Атрибути: id, user_id, order_date, status, total_price. Відповідає за зберігання інформації про замовлення користувачів.

-OrderDetail (Деталі замовлення)

-Атрибути: id, order_id, product_id, quantity, price. Зберігає деталі конкретних замовлень, включаючи продукти, які були замовлені.

-Payment (Оплата)

-Атрибути: id, order_id, payment_method, payment_date, amount.

Зберігає інформацію про оплату замовлень.

Вигляди:

-Home View

-Відображає головну сторінку магазину з популярними продуктами, акціями та іншими важливими даними.

-Product List View

-Відображає список продуктів певної категорії або результат пошуку.

-Product Detail View

-Відображає детальну інформацію про конкретний продукт, включаючи його опис, ціну та наявність на складі.

-Cart View

-Відображає вміст кошика користувача, де можна переглянути обрані продукти, змінити їх кількість або видалити.

-Checkout View

-Відображає форму для оформлення замовлення, включаючи введення адреси доставки та вибір способу оплати.

-Order Confirmation View

-Відображає підтвердження замовлення, зведену інформацію про зроблене замовлення.

-User Profile View

Відображає профіль користувача з можливістю редагування особистої інформації та перегляду історії замовлень.

Контролери:

-UserController

-Методи: login(), register(), logout(), profile(), updateProfile()

-Обробляє запити, пов'язані з користувачами, включаючи автентифікацію та управління профілем.

-ProductController

-Методи: listProducts(), viewProduct(), searchProducts()

-Обробляє запити, пов'язані з продуктами, такі як перегляд списку продуктів і детальної інформації про продукт.

-CartController

-Методи: viewCart(), addToCart(), removeFromCart(), updateCart()

-Обробляє запити, пов'язані з кошиком користувача, включаючи додавання та видалення продуктів.

-OrderController

-Методи: createOrder(), viewOrder(), listOrders()

-Обробляє запити, пов'язані з замовленнями, включаючи створення нового замовлення та перегляд історії замовлень.

-PaymentController

-Методи: processPayment(), viewPayment()

-Обробляє запити, пов'язані з оплатою замовлень.

-CategoryController

-Методи: listCategories(), viewCategory()

-Обробляє запити, пов'язані з категоріями продуктів.

Зв'язок клієнтської частини з серверною і базою даних здійснюється через API (Application Programming Interface). API слугує мостом між клієнтською і серверною частинами, дозволяючи обмінюватися даними і командними запитами. Клієнтська частина надсилає HTTP-запити до серверу, який, отримуючи ці запити, виконує відповідні операції з базою даних. Ці запити можуть бути різних типів, таких як GET для отримання даних, POST для створення нових записів, PUT для оновлення існуючих даних і DELETE для видалення записів.

Після отримання HTTP-запиту сервер обробляє його, використовуючи відповідний контролер, який звертається до моделей для виконання необхідних операцій з базою даних. Наприклад, при отриманні запиту на створення нового замовлення, сервер використовує модель замовлення для збереження даних про нове замовлення в базі даних. Після виконання операції сервер формує відповідь у форматі JSON або XML, яка містить результати виконаних дій або запитані дані, і повертає цю відповідь клієнту.

XML документ зображений на рисунку 2.13

Рисунок 2.13 — Схема XML – документу

Для забезпечення безпеки і надійності передачі даних між клієнтською і серверною частинами використовуються такі протоколи, як HTTPS. HTTPS (Hypertext Transfer Protocol Secure) забезпечує шифрування даних, переданих між клієнтом і сервером, що захищає інформацію від перехоплення або зміни під час передачі. Це особливо важливо для веб-додатків, які обробляють конфіденційні дані, такі як особиста інформація користувачів або дані про замовлення.

Крім того, для підвищення безпеки даних на серверній стороні можуть використовуватися додаткові методи аутентифікації та авторизації, такі як токени доступу або OAuth. Ці методи гарантують, що тільки авторизовані користувачі можуть виконувати певні дії або доступатися до конкретних ресурсів. Усі ці заходи разом сприяють створенню надійного і безпечного зв'язку між клієнтською частиною, сервером і базою даних, забезпечуючи цілісність і конфіденційність даних.

Детальні лістинги коду наведено в Додатку А.

2.5 Висновки до Розділу 2.

У висновку можна підкреслити, що цей розділ детально описав процес проектування і розробки нашої системи. Було розглянуто архітектурні рішення,

такі як вибір реляційної бази даних MySQL, нормалізація даних для забезпечення цілісності і уникнення дублювання, а також застосування принципів масштабованості і безпеки. Окрему увагу приділено розробці клієнтської частини з використанням HTML, CSS та JavaScript для створення зручного і функціонального інтерфейсу користувача. Серверна частина забезпечує обробку запитів і взаємодію з базою даних через API, що гарантує надійний обмін даними. Використання MVC архітектури дозволяє ефективно організувати код і спростити подальшу підтримку і розвиток системи. Таким чином, розділ 2 надав всебічний огляд процесу проектування і розробки інтернет-магазину Future Gadget, закладаючи основу для його успішного впровадження і функціонування.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРНЕТ -МАГАЗИНУ FUTUREGADGET

3.1 Тестування системи, види тестів та результати тестування

Тестування системи є критично важливим етапом розробки інтернет-магазину Future Gadget, який гарантує, що всі компоненти працюють коректно та безпечно. Основними типами тестування, які ми використовуємо, є юніт-тестування, інтеграційне тестування та системне тестування. Юніт-тестування фокусується на перевірці окремих модулів коду, забезпечуючи, що кожна функція працює належним чином в ізоляції. Це дозволяє швидко ідентифікувати і виправляти помилки в окремих компонентах системи.

Інтеграційне тестування перевіряє взаємодію між різними модулями системи. Наприклад, ми тестуємо, як серверна частина взаємодіє з базою даних та клієнтською частиною. Це включає перевірку коректності обробки запитів API, передачі даних між фронтендом і бекендом, а також оновлення інформації в базі даних. Інтеграційне тестування допомагає виявити проблеми, які можуть виникнути через невідповідність у взаємодії різних компонентів системи.

Системне тестування перевіряє всю систему в цілому, включаючи всі інтегровані модулі і компоненти. На цьому етапі ми імітуємо реальні сценарії використання інтернет-магазину, такі як реєстрація користувача, додавання товарів до кошика, оформлення замовлення, і перевіряємо, чи працюють всі функції як очікується. Системне тестування також включає перевірку на сумісність з різними браузерами та пристроями, що забезпечує коректну роботу сайту на різних платформах.

Результати тестування ретельно документуються. Для кожного типу тестування створюються звіти, що містять опис тестових сценаріїв, очікувані результати та фактичні результати. У разі виявлення помилок ми створюємо завдання для їх виправлення та повторного тестування. Цей підхід гарантує, що всі проблеми будуть вирішені до впровадження системи.

Безпека і надійність передачі даних є важливими аспектами тестування. Ми перевіряємо коректність реалізації HTTPS для захисту даних, що передаються між клієнтською та серверною частинами. Також тестуємо механізми автентифікації та авторизації, щоб гарантувати, що тільки авторизовані користувачі мають доступ до своїх даних та функцій системи.

Нарешті, тестування на навантаження та продуктивність дозволяє оцінити, як система поводиться під високими навантаженнями. Ми симулюємо велику кількість одночасних користувачів і запитів, щоб перевірити стабільність та ефективність роботи сайту. Це дозволяє виявити можливі вузькі місця в продуктивності та оптимізувати їх до впровадження системи в експлуатацію.

Компактна схема видів тестування подана на рисунку 3.1

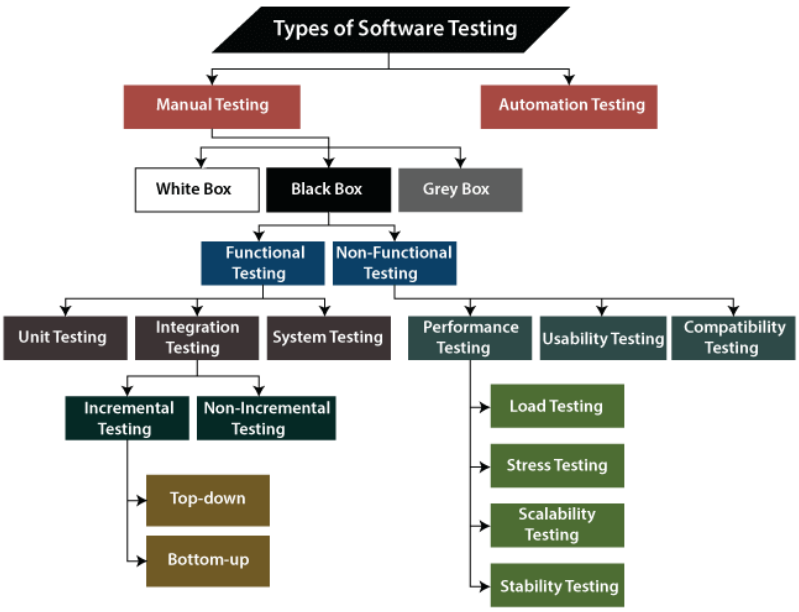

Рисунок 3.1 — Види тестування

З погляду теоретичних аспектів тестування нашої системи, можна зробити кілька висновків. По-перше, важливо перевірити функціональну стабільність, щоб упевнитися, що всі частини системи працюють правильно і відповідають вимогам. Тестування повинно включати всі ключові сценарії використання, щоб уникнути помилок.

Другий аспект стосується безпеки. Система повинна бути захищеною від різних видів атак, таких як SQL-ін'єкції, шкідливі введення даних і несанкціонований доступ. Тестування має переконати, що всі заходи безпеки ефективно застосовані і працюють належним чином.

Третій аспект - стійкість системи під навантаженням. Важливо визначити, як система поводиться під великим навантаженням користувачів, щоб уникнути збоїв та перебоїв в роботі. Стрес-тестування допомагає виявити межі можливостей системи і її реакцію на інтенсивне використання.

Четвертий аспект стосується сумісності з різними платформами і пристроями. Тестування повинно переконати, що система працює на різних ОС та браузерах без значних відхилень у функціональності або відображенні.

Завершуючий аспект - юзабіліті та зручність користування. Важливо, щоб інтерфейс системи був інтуїтивно зрозумілим і легким для використання користувачами. Тестування юзабіліті дозволяє виявити проблеми з навігацією, доступністю функцій та загальною зручністю використання системи.

Ці аспекти тестування допоможуть забезпечити високу якість і надійність нашої системи перед її впровадженням у реальне виробниче середовище.

3.2 Впровадження та розгортання системи

Розгортання системи на сервері або хмарній платформі є критичним етапом впровадження будь-якого програмного продукту. Перед початком розгортання необхідно підготувати середовище для програми, що включає в себе встановлення необхідних програмних засобів та забезпечення відповідності системних вимог програмного забезпечення. Для цього зазвичай використовують віртуальні сервери або хмарні платформи, які надають інфраструктурні послуги у вигляді віртуальних машин.

Кроки розгортання включають встановлення і конфігурацію веб-сервера (наприклад, Apache, Nginx), налаштування бази даних (наприклад, MySQL, PostgreSQL), і налаштування необхідних сервісів (наприклад, PHP, Node.js).

Після цього програмне забезпечення розгортається на сервері, з урахуванням вимог щодо безпеки та оптимізації продуктивності.

Після розгортання системи на сервері важливо налаштувати середовище для забезпечення безпеки, надійності і доступності. Один з ключових аспектів - це конфігурація сервера, яка включає в себе налаштування файрволів, контроль доступу, моніторинг системних ресурсів і логування подій.

Для забезпечення безпеки важливо встановити необхідні оновлення безпеки оперативної системи і всіх компонентів програмного забезпечення, що використовуються. Також доцільно реалізувати механізми резервного копіювання, які забезпечують автоматичне або регулярне створення резервних копій даних і конфігурацій, що зменшує ризик втрати інформації в разі аварійного відновлення або втрати даних.

Впровадження і налаштування серверного середовища відіграють критичну роль у забезпеченні стабільності та безпеки системи, що впливає на її успішне функціонування та задоволення потреб користувачів.

Рисунок 3.2 ілюструє схему розгортання програмної системи.

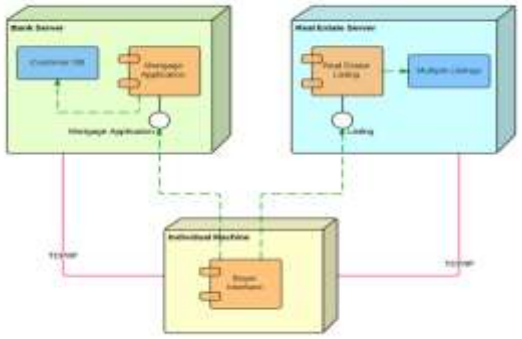

Рисунок 3.2 — схема розгортання

Перш за все, необхідно підготувати сервер або вибрану хмарну платформу, налаштувати операційну систему та встановити всі необхідні програмні компоненти, такі як веб-сервер (наприклад, Apache або Nginx), сервер баз даних (наприклад, MySQL), та інші залежності. Після цього

відбувається завантаження і розгортання коду системи, а також налаштування конфігураційних файлів для відповідності специфікаціям середовища. Особлива увага приділяється налаштуванню безпеки: конфігурація файрволів, налаштування SSL-сертифікатів для забезпечення HTTPS-з'єднань, створення політик доступу та моніторинг системи для виявлення потенційних загроз. Важливим кроком є також налаштування резервного копіювання даних для захисту інформації від втрат і забезпечення можливості швидкого відновлення у випадку збоїв. Усі ці заходи спрямовані на забезпечення надійності, безпеки та ефективності роботи системи в реальних умовах експлуатації.

3.3 Моніторинг та обслуговування

Слід також зазаначити, що окрему роль відіграють моніторинг та обслуговування для забезпечення її стабільної роботи, виявлення потенційних проблем та своєчасного їх усунення. Моніторинг включає постійний нагляд за станом сервера, використанням ресурсів (таких як CPU, пам'ять, дисковий простір), а також за працездатністю всіх компонентів системи. Для цього використовуються спеціалізовані інструменти, такі як Zabbix, Nagios, або Prometheus, які дозволяють автоматично збирати і аналізувати дані про роботу системи, встановлювати порогові значення для сповіщень і реагувати на аномалії в реальному часі.

Важливою частиною моніторингу є спостереження за продуктивністю системи. Це включає відстеження швидкості відповіді на запити користувачів, часу завантаження сторінок, а також аналіз журналів доступу і помилок. Регулярний аналіз цих показників дозволяє виявляти вузькі місця та оптимізувати роботу системи, забезпечуючи високий рівень обслуговування для кінцевих користувачів. Окрім того, варто впровадити систему алертів, яка автоматично надсилатиме повідомлення адміністраторам у разі виникнення критичних проблем або перевищення допустимих значень параметрів.

Обслуговування системи поєднує в собі регулярні оновлення програмного забезпечення для забезпечення безпеки та впровадження нових функціональних можливостей. Це може включати оновлення операційної системи, веб-сервера, серверу баз даних та інших компонентів інфраструктури. Оновлення повинні проводитися обережно, з попереднім тестуванням на тестовому середовищі, щоб мінімізувати ризики виникнення проблем у робочому середовищі. Важливим аспектом обслуговування є також резервне копіювання даних. Регулярне створення резервних копій баз даних та інших важливих даних дозволяє швидко відновити систему у разі збоїв або втрати даних.

Безпеці системи теж слід приділити увагу при моніторингу та обслуговуванні. Зокрема регулярному аудиу безпеки, перевірку на наявність вразливостей, встановлення патчів та оновлень, а також налаштування системи для захисту від потенційних атак. Важливо також забезпечити належний рівень контролю доступу, використовуючи методи аутентифікації та авторизації, щоб запобігти несанкціонованому доступу до системи. Регулярне навчання персоналу з питань безпеки та належного обслуговування також є важливим для підтримання високого рівня безпеки і надійності системи.

3.4 Висновки та рекомендації

Висновки щодо розробки інтернет-магазину "Future Gadget" відображають наші ключові знахідки та досвід, отриманий під час створення цієї складної системи. Створення інтернет-магазину потребує ретельного планування, починаючи з проектування бази даних до розробки клієнтської та серверної частин. Вибір реляційної бази даних MySQL виявився виправданим завдяки її продуктивності, надійності та можливості масштабування. Нормалізація даних забезпечила цілісність інформації та мінімізувала дублювання, що є критичним для ефективного управління великим обсягом даних.

Однією з ключових переваг системи стало її безпечне та надійне з'єднання між клієнтською частиною, сервером та базою даних за допомогою HTTPS. Це забезпечило конфіденційність та цілісність даних, що передаються між користувачами та сервером. Використання API для обміну даними між клієнтською і серверною частинами дозволило нам створити гнучку та модульну архітектуру, яка легко адаптується до змін та масштабування.

Рекомендації щодо подальшого розвитку інтернет-магазину включають кілька важливих аспектів. По-перше, слід постійно оновлювати та вдосконалювати систему безпеки, оскільки кіберзагрози еволюціонують. Впровадження додаткових механізмів аутентифікації, таких як двофакторна аутентифікація, допоможе підвищити захищеність користувачів. По-друге, варто зосередитися на оптимізації продуктивності, впроваджуючи кешування на різних рівнях системи та оптимізуючи запити до бази даних.

Звернемо увагу також на той факт, що важливим інструментом для ефективного планування, комунікації, визначення меж системи, ідентифікації взаємодій, планування інтеграції та документування проекту є контекстна діаграма інтернет-магазину. Вона забезпечує високорівневий огляд системи і може бути корисною з тих причин, що допомагає визначити основні компоненти системи і їх взаємодії з зовнішніми системами або користувачами, забезпечує зрозумілу та інтуїтивну візуалізацію, яка може бути використана для спілкування з клієнтами, менеджерами проектів, розробниками, тестувальниками та іншими зацікавленими сторонами, допомагаючи тим самим пояснити основні функції системи та її межі без необхідності вдаватися у технічні деталі, чітко визначає, які компоненти є частиною системи, а які належать до зовнішніх систем або користувачів. Це допомагає уникнути плутанини щодо відповідальності та взаємодії між різними елементами проекту.

Декомпозицію контекстної діаграми діяльності інтернет-магазину наведено на рисунку 3.3.

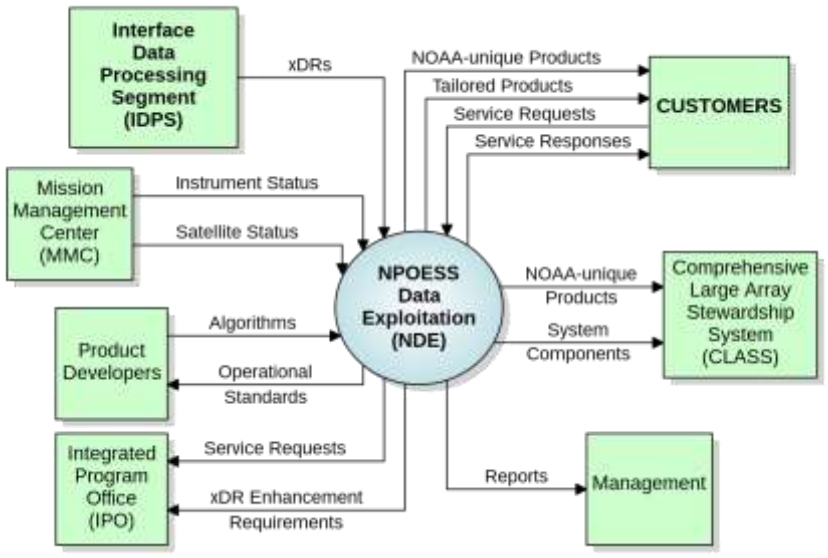

Рисунок 3.3 — Контекстна діаграма

Також важливо продовжувати розвивати користувацький досвід (UX). Регулярні опитування та аналіз поведінки користувачів дозволять виявляти слабкі місця та вносити відповідні покращення. Інтеграція з популярними платформами соціальних медіа та маркетинговими інструментами допоможе залучати нових клієнтів і підвищувати лояльність існуючих.

Ще одним аспектом є розширення функціональності магазину, зокрема впровадження персоналізованих рекомендацій на основі машинного навчання. Це дозволить пропонувати клієнтам товари, що відповідають їхнім інтересам, підвищуючи ймовірність покупки. Крім того, варто розглянути можливість впровадження нових методів оплати, таких як криптовалюти, щоб розширити аудиторію та надати більше варіантів для клієнтів.

Загалом, наш інтернет-магазин "Future Gadget" став прикладом успішної реалізації сучасного електронного бізнесу. Проте для збереження конкурентоспроможності та задоволення потреб клієнтів необхідно постійно розвиватися, впроваджуючи нові технології та підходи до управління бізнесом.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні проблеми безпеки життєдіяльності

Ергономіка є важливою складовою безпеки життєдіяльності, оскільки вона вивчає взаємодію людини з навколишнім середовищем, обладнанням та іншими людьми. У контексті інтернет-магазину «Future Gadget» ергономіка відіграє ключову роль у забезпеченні комфортних умов для роботи як співробітників, так і користувачів сайту. Ергономічні проблеми можуть впливати на ефективність праці, здоров'я та безпеку працівників, а також на задоволеність користувачів і, відповідно, на успішність бізнесу.

Однією з основних ергономічних проблем є адаптація робочого місця до фізичних можливостей та потреб працівника. Це включає вибір меблів, освітлення, розташування обладнання та програмного забезпечення, які повинні відповідати антропометричним даним і фізіологічним особливостям працівників. Неправильна організація робочого місця може призвести до стомлення, зниження працездатності та навіть до розвитку професійних захворювань. У випадку з працівниками, що займаються обслуговуванням інтернет-магазину, це може означати зручні крісла, належне освітлення і оптимальне розташування комп'ютерної техніки, щоб мінімізувати навантаження на очі та спину.

Крім того, важливим аспектом ергономіки є психологічний комфорт працівників. Це включає забезпечення сприятливого психологічного клімату в колективі, уникнення монотонності в роботі та створення умов для зменшення стресових ситуацій. Для інтернет-магазину це може означати використання програмного забезпечення, яке автоматизує рутинні завдання, забезпечує зручні інтерфейси для взаємодії з клієнтами та дозволяє швидко і легко виконувати необхідні операції. Наприклад, інтуїтивно зрозумілий і зручний інтерфейс адміністративної панелі сайту допомагає працівникам швидко знаходити

необхідну інформацію, обробляти замовлення та виконувати інші завдання, що зменшує рівень стресу і підвищує ефективність роботи.

Особлива увага в ергономіці приділяється програмним інтерфейсам. Інтерфейс інтернет-магазину повинен бути розроблений таким чином, щоб користувачі могли легко і безпечно здійснювати покупки, знаходити необхідні товари та інформацію, взаємодіяти з підтримкою клієнтів. Ергономічно недосконалі інтерфейси можуть викликати роздратування у користувачів, призводити до помилок у замовленнях та навіть до відмови від подальших покупок. Таким чином, ергономіка програмних інтерфейсів є важливою складовою безпеки життєдіяльності як для працівників, так і для користувачів інтернет-магазину.

У сучасних умовах розвитку технологій важливо також враховувати новітні ергономічні дослідження та впроваджувати інноваційні рішення. Наприклад, використання технологій віртуальної та доповненої реальності може створити нові можливості для покращення ергономіки робочих місць і взаємодії користувачів з інтернет-магазином. Впровадження таких технологій може значно підвищити ефективність роботи, забезпечити вищий рівень задоволеності користувачів і сприяти зменшенню професійних ризиків.

Ергономіка також відіграє важливу роль у забезпеченні безпеки користувачів інтернет-магазину. Інтерфейси веб-сайту повинні бути розроблені таким чином, щоб користувачі могли легко і швидко знаходити необхідну інформацію, здійснювати покупки і взаємодіяти з підтримкою клієнтів. Це включає створення логічної структури сайту, використання зрозумілих меню і навігаційних елементів, а також забезпечення високої швидкості завантаження сторінок. Зручність і доступність інтерфейсу підвищують задоволення користувачів і сприяють зростанню продажів.

У контексті сучасних технологій важливо також враховувати новітні ергономічні дослідження і впроваджувати інноваційні рішення. Наприклад, використання технологій віртуальної і доповненої реальності може створити нові можливості для покращення ергономіки робочих місць і взаємодії

користувачів з інтернет-магазином. Такі технології можуть забезпечити більш інтерактивний і зручний досвід для користувачів, а також покращити умови праці для працівників.

4.2 Організація безпечної роботи електроустановок

Безпечна робота електроустановок є одним із ключових аспектів охорони праці, оскільки неправильне використання електрообладнання може призвести до серйозних наслідків, таких як ураження електричним струмом, пожежі та вибухи. Особливо важливо дотримуватися вимог електробезпеки в умовах роботи інтернет-магазину, де електронне обладнання використовується на постійній основі.

Перш за все, всі електроустановки повинні бути встановлені та обслуговуватися відповідно до діючих нормативів і стандартів. Це включає регулярні перевірки і технічне обслуговування електрообладнання для виявлення і усунення можливих дефектів. Всі електроустановки повинні бути захищені від механічних пошкоджень, вологості та інших несприятливих факторів, що можуть вплинути на їхню роботу. Регулярні огляди та профілактичні роботи допомагають підтримувати справність обладнання та запобігати аварійним ситуаціям.

Навчання і інструктажі працівників з електробезпеки є ще одним важливим аспектом організації безпечної роботи електроустановок. Всі працівники повинні пройти відповідне навчання з електробезпеки, яке включає теоретичні знання і практичні навички. Інструктажі повинні проводитися регулярно, а також при прийнятті на роботу і при зміні умов праці. Працівники повинні знати основні правила безпечного користування електрообладнанням, методи надання першої допомоги при ураженні електричним струмом і дії у разі виникнення аварійних ситуацій.

Використання засобів індивідуального захисту (ЗІЗ) є обов'язковим для працівників, які працюють з електроустановками. Це включає діелектричні

рукавиці, боти, захисний одяг та інші засоби, що запобігають ураженню електричним струмом. ЗІЗ повинні бути сертифіковані і регулярно перевірятися на предмет їхньої справності. Заземлення та занулення електроустановок є необхідними заходами для забезпечення безпеки. Вони забезпечують безпечне відведення струму у землю у разі пошкодження ізоляції або інших аварійних ситуацій, що запобігає ураженню електричним струмом і пошкодженню обладнання. Заземлювальні пристрої повинні регулярно перевірятися і відповідати встановленим нормам і стандартам безпеки. Використання сучасних захисних пристроїв, таких як диференційні автомати, автоматичні вимикачі та реле захисту, є ще одним важливим аспектом забезпечення електробезпеки. Ці пристрої автоматично відключають електроживлення у разі виникнення аварійної ситуації, що дозволяє миттєво реагувати на зміни в електричній мережі і запобігати ураженню електричним струмом та пошкодженню обладнання. Встановлення і регулярне обслуговування таких пристроїв є необхідним для забезпечення надійної і безпечної роботи електроустановок.Організація безпечної роботи електроустановок включає також заходи з мінімізації електромагнітних випромінювань, які можуть негативно впливати на здоров'я працівників. Це включає використання екрануючих матеріалів, правильне розташування обладнання і дотримання встановлених нормативів щодо допустимих рівнів випромінювання. Зниження рівня електромагнітних випромінювань сприяє створенню безпечних умов праці і запобігає можливим негативним наслідкам для здоров'я працівників.Отже, організація безпечної роботи електроустановок є комплексним процесом, який включає технічне обслуговування, навчання працівників, використання засобів індивідуального захисту, забезпечення відповідних умов експлуатації і використання сучасних захисних пристроїв. Виконання цих заходів забезпечує безпеку працівників, надійну роботу електроустановок і запобігання аварійним ситуаціям, що є ключовими аспектами охорони праці у будь-якій організації, включаючи інтернет-магазин «Future Gadget».

4.3 Висновок до четвертого розділу

У четвертому розділі було розглянуто ключові аспекти безпеки життєдіяльності та основ охорони праці, які є критично важливими для функціонування інтернет-магазину «Future Gadget». Особлива увага була приділена ергономічним проблемам безпеки життєдіяльності, що включає в себе питання організації робочого місця, налаштування обладнання, психологічного комфорту працівників, а також адаптації програмного забезпечення до потреб користувачів. Забезпечення ергономічних умов сприяє підвищенню продуктивності праці, зменшенню кількості професійних захворювань та підвищенню загального задоволення від роботи.

Окремо було розглянуто питання організації безпечної роботи електроустановок, яке включає технічне обслуговування обладнання, навчання працівників з електробезпеки, використання засобів індивідуального захисту, заземлення і занулення електроустановок, а також застосування сучасних захисних пристроїв. Дотримання вимог електробезпеки є критично важливим для запобігання аварійним ситуаціям, зниження ризику ураження електричним струмом і забезпечення надійної роботи електрообладнання.

Таким чином, впровадження ергономічних принципів і заходів з електробезпеки є необхідними для створення безпечних і комфортних умов праці, що сприяє збереженню здоров'я працівників, підвищенню ефективності їхньої діяльності та забезпеченню безперебійної роботи інтернет-магазину. Реалізація зазначених заходів дозволить мінімізувати ризики, пов'язані з небезпечними факторами виробничого середовища, і забезпечити високу якість обслуговування клієнтів, що є важливими складовими успіху компанії «Future Gadget».

ВИСНОВКИ

У даній кваліфікаційній роботі освітнього рівня «Бакалавр» було здійснено комплексне дослідження та розробку інтернет-магазину «Future Gadget», що включає всі етапи створення програмної системи – від проектування бази даних до реалізації клієнтської і серверної частин, а також тестування та впровадження. Робота охоплює теоретичні аспекти, практичні рішення і заходи щодо забезпечення безпеки життєдіяльності та охорони праці.

Інтернет-магазин є складною системою, що вимагає інтеграції різних технологій та підходів для забезпечення його функціонування та задоволення потреб користувачів. Процес розробки включав в себе планування, проектування, реалізацію та тестування системи. Особлива увага приділялася забезпеченню безпеки даних, їх цілісності та захисту від можливих загроз. Це було досягнуто шляхом використання сучасних методів захисту даних, таких як шифрування HTTPS та аутентифікація користувачів.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано основи електронної комерції, визначено ключові аспекти та переваги створення інтернет-магазинів.

– Розглянуто сучасні технології та інструменти, що використовуються для розробки веб-застосунків.

– Висвітлено архітектурні підходи до проектування програмних систем, зокрема, модель MVC (Model-View-Controller), мікросервісна архітектура, клієнт — серверна модель, тощо.

– Проаналізовано різні платформи та фреймворки для створення інтернет-магазинів, обґрунтовано вибір MySQL для управління базою даних.

В другому розділі кваліфікаційної роботи:

– Досліджено процес проектування бази даних для інтернет-магазину, визначено основні сутності, їх атрибути та зв'язки між ними.

– Обґрунтовано вибір технологій для реалізації клієнтської та серверної частин системи, описано використання HTML, CSS, JavaScript та Python.

– Сформовано структуру веб-додатку, розроблено основні сторінки інтернет-магазину, включаючи головну сторінку, сторінку продуктів, кошик, сторінку оформлення замовлення, а також сторінки доставки і оплати.

В третьому розділі кваліфікаційної роботи:

– Розроблено серверну частину системи, яка обробляє запити від клієнтів, взаємодіє з базою даних та забезпечує виконання бізнес-логіки.

– Запропоновано методи забезпечення безпеки даних користувачів та захисту системи від можливих загроз.

– Спроєктовано інтерфейс користувача з урахуванням принципів зручності та інтуїтивності, проведено тестування з користувачами для оцінки його ефективності.

– Протестовано систему на різних етапах розробки, проведено функціональне, регресійне та навантажувальне тестування для забезпечення стабільності та надійності роботи.

У розділі «Безпека життєдіяльності, основи охорони праці» … Висвітлено заходи з охорони праці та безпеки життєдіяльності при розробці та експлуатації програмного забезпечення, розглянуто нормативні документи та стандарти, що регламентують безпеку інформаційних систем та захист персональних даних, а також проаналізовано потенційні ризики та загрози для здоров’я працівників під час роботи з комп’ютерною технікою, запропоновано рекомендації щодо їх мінімізації.

Розробка інтернет-магазину «Future Gadget» продемонструвала важливість інтеграції різних компонентів системи для досягнення високої якості продукту. Проектування бази даних, реалізація клієнтської та серверної частин, тестування і впровадження системи – всі ці етапи є критичними для успіху проекту. Застосування сучасних технологій, методів забезпечення безпеки даних та належне тестування дозволило створити надійну та ефективну систему, яка відповідає потребам користувачів і вимогам ринку.

Ця кваліфікаційна робота стала важливим кроком у професійному розвитку та здобутті практичних навичок у галузі розробки веб-застосунків.

Отримані результати можуть бути використані для подальшого вдосконалення системи та розробки нових функціональних можливостей, що сприятиме підвищенню конкурентоспроможності інтернет-магазину на ринку електронної комерції.

ПЕРЕЛІК ДЖЕРЕЛ

1. Петров А., Сидоренко Ю. ТЕНДЕНЦІЇ РОЗВИТКУ ІНТЕРНЕТ-КОМЕРЦІЇ В УКРАЇНІ. Економіка та управління. 2022. № 3. С. 12. URL: https://doi.org/10.32782/2524-0072/2022-45-55 (дата звернення: 20.02.2024).

2. Михайленко І., Ковальчук В. ОПТИМІЗАЦІЯ ФУНКЦІОНАЛУ ІНТЕРНЕТ-МАГАЗИНІВ ЗА ДОПОМОГОЮ МОДУЛІВ : навч. посіб. Київ : Нац. ун-т "Київ. політехн. ін-т", 2021. 130 с.

3. Іваненко П., Тарасенко М. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО РОЗРОБКИ ПЛАГІНІВ ДЛЯ Е-КОМЕРЦІЇ. Харків : Наук. вісн. ХНУРЕ, 2023. 110 с.

4. Литвиненко В.М. ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ СИСТЕМ УПРАВЛІННЯ КОНТЕНТОМ ДЛЯ ІНТЕРНЕТ-МАГАЗИНІВ : навч. посіб. Львів : Вісн. НУ "Львів. політехніка", 2020. 120 с.

5. Олексієнко М., Савченко Л. ПОРІВНЯЛЬНИЙ АНАЛІЗ CMS-СИСТЕМ ДЛЯ СТВОРЕННЯ ВЕБ-САЙТІВ : підручник. Київ : Вісн. НТУУ "КПІ", 2019. 108 с.

6. ПЕРЕВАГИ ТА НЕДОЛІКИ РІЗНИХ CMS. W3Schools. URL: https://www.w3schools.com/wordpress/introduction-to-content-management-system-cms (дата звернення: 02.02.2024).

7. ТЕНДЕНЦІЇ РОЗВИТКУ CMS В СФЕРІ Е-КОМЕРЦІЇ. Gartner. URL: https://www.gartner.com/reviews/market/digital-commerce (дата звернення: 01.02.2024).

8. Романенко С., Гриценко Т. СУЧАСНІ CMS-СИСТЕМИ: МОЖЛИВОСТІ ТА ПЕРСПЕКТИВИ РОЗВИТКУ : навч. посіб. Львів : Вісн. НУ "Львів. політехніка", 2021. 122 с.

9. ВИБІР CMS-СИСТЕМИ ДЛЯ СТВОРЕННЯ ІНТЕРНЕТ-МАГАЗИНУ: КРИТЕРІЇ ТА РЕКОМЕНДАЦІЇ : конспект лекцій. Харків : Вісн. НТУ "ХПІ", 2020. 125 с.

10. PrestaShop Official Documentation. Devdocs. URL: https://devdocs.prestashop-project.org/ (дата звернення: 03.06.2024).

11. PrestaShop Community Forums. PrestaShop. URL: https://www.prestashop.com/forums/ (дата звернення: 30.05.2024).

12. PrestaShop Developer Resources. Devdocs. URL: https://devdocs.prestashop-project.org/ (дата звернення: 20.05.2024).

13. PrestaShop Blog. PrestaShop. URL: https://www.prestashop.com/forums/topic/278867-what-is-the-best-way-to-run-a-blog-in-prestashop/ (дата звернення: 10.01.2024).

14. РОЗРОБКА МОДУЛІВ ДЛЯ PRESTASHOP. Nicolas Descombes. URL: https://devdocs.prestashop-project.org/8/modules/ (дата звернення: 17.02.2024).

15. Ткаченко О., Марченко П. РОЗРОБКА МОДУЛІВ ДЛЯ PRESTASHOP: ТЕОРЕТИЧНІ ТА ПРАКТИЧНІ АСПЕКТИ : навч. посіб. Київ : Вісн. НТУ "КПІ", 2019. 96 с.

16. ЩО ТАКЕ PHP? НАПИШІТЬ СВОЮ ПЕРШУ ПРОГРАМУ НА PHP. Guru99. URL: https://www.guru99.com/what-is-php-first-php-program.html (дата звернення: 22.02.2024).

17. PHP: The Complete Reference. Amazon. URL: https://www.amazon.com/PHP-Complete-Reference-Steven-Holzner/dp/0071508546 (дата звернення: 19.02.2024).

18. JavaScript and JQuery: Interactive Front-End Web Development. Amazon. URL: https://www.amazon.com/javascript/s?k=javascript (дата звернення: 22.05.2024).

19. HTML and CSS: Designing and Building Web. Amazon. URL: https://www.amazon.com/html-css/s?k=html+and+css (дата звернення: 12.04.2024).

20. Яценко І., Гончар О. ВИБІР СЕРЕДОВИЩА РОЗРОБКИ ДЛЯ PHP-ПРОЕКТІВ: КРИТЕРІЇ ТА РЕКОМЕНДАЦІЇ. : навч. посіб. Харків : НТУ "ХПІ", 2020. 130 с.

21. Документація PHPStorm. JetBrains. URL: https://www.jetbrains.com/phpstorm/resources/ (дата звернення: 06.05.2024).

22. PrestaShop Developer's Handbook. Devdocs. URL: https://devdocs.prestashop-project.org/ (дата звернення: 04.02.2024).

23. Mastering PrestaShop. Addons PrestaShop. URL: https://addons.prestashop.com/en/art-culture-themes/31468-online-book-and-stationary-store.html (дата звернення: 20.02.2024).

24. 3-Tier Architecture: A Complete Overview. Jinfonet. URL: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/ (дата звернення: 21.01.2024).

25. 5 Benefits of a 3-Tier Architecture. Izenda. URL: https://www.izenda.com/5-benefits-3-tier-architecture/ (дата звернення: 20.02.2024).

26. ER Diagram: Entity Relationship Diagram Model DBMS Example. Cacoo. URL: https://cacoo.com/templates/er-database-diagram (дата звернення: 23.02.2024).

27. ЩО ТАКЕ MYSQL? УСЕ, ЩО ВАМ ПОТРІБНО ЗНАТИ. Talend. URL: https://www.talend.com/resources/what-is-mysql/ (дата звернення: 29.01.2024).

28. Rouse M. MySQL. TechTarget. URL: https://searchoracle.techtarget.com/definition/MySQL (дата звернення: 29.01.2024).

29. Site File Structure. Web Style Guide. URL: https://webstyleguide.com/wsg3/5-site-structure/3-site-file-structure.html (дата звернення: 29.01.2024).

30. Чорток Ю. В. ВПЛИВ ФАКТОРІВ ТРУДОВОГО СЕРЕДОВИЩА НА СТАН ЗДОРОВ'Я ПРАЦІВНИКІВ ТА ЇХ ПРАЦЕЗДАТНІСТЬ. library.if.ua. URL: https://essuir.sumdu.edu.ua/bitstream/123456789/28559/1/Ekonomichni_factory.pdf (дата звернення: 07.06.2024).

31. РОЛЬ ЦЕНТРАЛЬНОЇ НЕРВОВОЇ СИСТЕМИ В ТРУДОВІЙ ДІЯЛЬНОСТІ ЛЮДИНИ : підручник. Київ : Ін-т харч. хімії і технології НАН, 2019. 67 с. URL: https://library.if.ua/book/9/920.html (дата звернення: 07.06.2024).

32. Овчаренко О. ПСИХОЛОГІЯ СТРЕСУ ТА СТРЕСОВИХ РОЗЛАДІВ. 2-ге вид. Київ : Ун-т "Україна", 2019. 266 с.

МОЗ. ЯК НАДАТИ ПЕРШУ ДОПОМОГУ ПРИ ОПІКАХ. Міністерство охорони здоров'я України. URL: https://www.kmu.gov.ua/news/moz-nagaduye-yak-nadati-pershu-dopomogu-pri-termichnih-opikah (дата звернення: 10.06.2024).

33 Андрієнко В., Дробот Т. ПРАКТИЧНІ АСПЕКТИ ВПРОВАДЖЕННЯ СИСТЕМ УПРАВЛІННЯ ЯКІСТЮ В Е-КОМЕРЦІЇ. Економічний форум. 2023. № 4. С. 89-97. URL: https://doi.org/10.32782/2524-0072/2023-45-67 (дата звернення: 20.06.2024).

34 Жуков О., Лазаренко К. РОЗРОБКА ТА ВПРОВАДЖЕННЯ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ІНТЕРНЕТ-МАГАЗИНІВ. Київ : Вид-во НТУУ "КПІ", 2022. 150 с.

35 Климчук С., Притула М. ОПТИМІЗАЦІЯ ПРОЦЕСІВ ЛОГІСТИКИ В ІНТЕРНЕТ-МАГАЗИНАХ. Харків : Вісн. ХНУРЕ, 2021. 130 с.

ДОДАТКИ

Додаток А

Програмний код.

HTML – вигляди сторінок

```
<!-- products.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Future Gadget - Products</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Our Products</h1>
        <nav>
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="cart.html">Cart</a></li>
                <li><a href="contact.html">Contact</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section id="product-list">
            <div class="product-grid">
                <!-- Product items will be inserted here dynamically -->
            </div>
        </section>
```

```
    </main>
    <script src="scripts.js"></script>
</body>
</html>

<!-- cart.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <title>Future Gadget - Cart</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Your Cart</h1>
        <nav>
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="products.html">Products</a></li>
                <li><a href="cart.html">Cart</a></li>
                <li><a href="contact.html">Contact</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section id="cart-items">
            <div class="cart-list">
                <!-- Cart items will be inserted here dynamically
-->
            </div>
        </section>
        <button id="checkout-button">Proceed to Checkout</button>
```

```
    </main>
    <script src="scripts.js"></script>
</body>
</html>


<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <title>Доставка та Оплата</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Доставка та Оплата</h1>
        <nav>
            <a href="index.html">Головна</a>
            <a href="products.html">Товари</a>
            <a href="cart.html">Кошик</a>
            <a href="delivery-payment.html">Доставка та Оплата</a>
        </nav>
    </header>
    <main>
        <section>
            <h2>Методи Доставки</h2>
            <p>Ми пропонуємо різні методи доставки для зручності
наших клієнтів:</p>
            <ul>
                <li>Кур'єрська доставка</li>
                <li>Самовивіз з магазину</li>
                <li>Пошта</li>
            </ul>
```

```
        </section>
        <section>
            <h2>Методи Оплати</h2>
            <p>Ви можете обрати зручний для вас метод оплати:</p>
            <ul>
                <li>Оплата готівкою при отриманні</li>
                <li>Оплата карткою онлайн</li>
                <li>Безготівковий розрахунок</li>
            </ul>
        </section>
        <section>
            <h2>Адреса Доставки</h2>
            <form id="deliveryForm">
                <label for="address">Адреса:</label>
                <input   type="text"   id="address"   name="address"
required>
                <label for="city">Місто:</label>
                <input type="text" id="city" name="city" required>
                <label for="postalCode">Поштовий індекс:</label>
                <input          type="text"          id="postalCode"
name="postalCode" required>
                <button type="submit">Підтвердити</button>
            </form>
        </section>
    </main>
    <script src="scripts.js"></script>
</body>
</html>
```

JavaScript

```
// scripts.js (continued)
document.addEventListener('DOMContentLoaded', function() {
    // Example of how to add event listeners to "Add to Cart"
buttons
```

```
    const cart = [];

    document.querySelectorAll('button[data-id]').forEach(button  =>
{
        button.addEventListener('click', function() {
            const productId = this.dataset.id;
            const   product   =   products.find(p   =>   p.id   ===
parseInt(productId));
            cart.push(product);
            updateCart();
        });
    });

    function updateCart() {
        const cartList = document.querySelector('.cart-list');
        cartList.innerHTML = '';
        cart.forEach(product => {
            const cartItem = document.createElement('div');
            cartItem.classList.add('cart-item');
            cartItem.innerHTML = `
                <h3>${product.name}</h3>
                <p>Price: $${product.price}</p>
                <button                    data-id="${product.id}"
class="remove">Remove</button>
            `;
            cartList.appendChild(cartItem);
         });
     }
});

document.getElementById('deliveryForm').addEventListener('submit',
function(event) {
    event.preventDefault();
    alert('Дані доставки збережені');
});
```

```
// scripts.js (continued)
document.addEventListener('DOMContentLoaded', function() {
    // Example of how to add event listeners to "Add to Cart" buttons
    const cart = [];

    document.querySelectorAll('button[data-id]').forEach(button => {
        button.addEventListener('click', function() {
            const productId = this.dataset.id;
            const    product    =    products.find(p    =>    p.id    ===
parseInt(productId));
            cart.push(product);
            updateCart();
        });
    });

    function updateCart() {
        const cartList = document.querySelector('.cart-list');
        cartList.innerHTML = '';
        cart.forEach(product => {
            const cartItem = document.createElement('div');
            cartItem.classList.add('cart-item');
            cartItem.innerHTML = `
                <h3>${product.name}</h3>
                <p>Price: $${product.price}</p>
                <button                          data-id="${product.id}"
class="remove">Remove</button>
            `;
            cartList.appendChild(cartItem);
        });
    }
});
```

CSS – стилізація

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

header {
    background-color: #333;
    color: #fff;
    padding: 15px 0;
    text-align: center;
}

header h1 {
    margin: 0;
}

nav a {
    color: #fff;
    margin: 0 10px;
    text-decoration: none;
}

main {
    padding: 20px;
}

section {
    margin-bottom: 20px;
}
```

```
h2 {
    color: #333;
}

form label {
    display: block;
    margin: 10px 0 5px;
}

form input {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
}

form button {
    padding: 10px 20px;
    background-color: #333;
    color: #fff;
    border: none;
    cursor: pointer;
}

form button:hover {
    background-color: #555;
}
/* styles.css (continued) */
form {
    display: flex;
    flex-direction: column;
}

form label {
    margin-top: 10px;
```

```
}

form input, form textarea {
    margin-bottom: 10px;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

form button {
    padding: 10px;
    border: none;
    border-radius: 5px;
    background-color: #333;
    color: white;
    cursor: pointer;
}



body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

header {
    background-color: #333;
    color: #fff;
    padding: 15px 0;
    text-align: center;
}

header h1 {
```

```
    margin: 0;
}

nav a {
    color: #fff;
    margin: 0 10px;
    text-decoration: none;
}

main {
    padding: 20px;
}

section {
    margin-bottom: 20px;
}

h2 {
    color: #333;
}

form label {
    display: block;
    margin: 10px 0 5px;
}

form input {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
}

form button {
    padding: 10px 20px;
```

```
    background-color: #333;
    color: #fff;
    border: none;
    cursor: pointer;
}

form button:hover {
    background-color: #555;
}

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

header {
    background-color: #333;
    color: #fff;
    padding: 15px 0;
    text-align: center;
}

header h1 {
    margin: 0;
}

nav a {
    color: #fff;
    margin: 0 10px;
    text-decoration: none;
}

main {
```

```
    padding: 20px;
}

section {
    margin-bottom: 20px;
}

h2 {
    color: #333;
}

ul {
    list-style-type: none;
    padding: 0;
}

ul li {
    background-color: #f4f4f4;
    margin: 5px 0;
    padding: 10px;
    border: 1px solid #ddd;
}

.button {
    display: inline-block;
    padding: 10px 20px;
    background-color: #333;
    color: #fff;
    text-decoration: none;
    margin-top: 20px;
}

.button:hover {
    background-color: #555;
}
```

Створення REST – API за дапомогою Flask і з використанням Python

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI']                                    =
'mysql://username:password@localhost/db_name'
db = SQLAlchemy(app)

class Product(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    price = db.Column(db.Float, nullable=False)
    description = db.Column(db.String(200))

@app.route('/products', methods=['GET'])
def get_products():
    products = Product.query.all()
    return jsonify([product.as_dict() for product in products])

@app.route('/products', methods=['POST'])
def add_product():
    data = request.json
    new_product   =   Product(name=data['name'],   price=data['price'],
description=data['description'])
    db.session.add(new_product)
    db.session.commit()
    return jsonify(new_product.as_dict()), 201

@app.route('/products/<int:id>', methods=['PUT'])
def update_product(id):
    data = request.json
    product = Product.query.get(id)
```

```
    if product:
        product.name = data['name']
        product.price = data['price']
        product.description = data['description']
        db.session.commit()
        return jsonify(product.as_dict())
    else:
        return jsonify({'error': 'Product not found'}), 404

@app.route('/products/<int:id>', methods=['DELETE'])
def delete_product(id):
    product = Product.query.get(id)
    if product:
        db.session.delete(product)
        db.session.commit()
        return '', 204
    else:
        return jsonify({'error': 'Product not found'}), 404

if __name__ == '__main__':
    app.run(debug=True)
```

Модуль для авторизації та автентифікації користувачів, Jason, Web Token

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash,
check_password_hash
import jwt
import datetime

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql://username:password@localhost/db_name'
```

```
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)

@app.route('/register', methods=['POST'])
def register():
    data = request.json
    hashed_password = generate_password_hash(data['password'],
method='sha256')
    new_user = User(username=data['username'],
password=hashed_password)
    db.session.add(new_user)
    db.session.commit()
    return jsonify({'message': 'User registered successfully'}), 201

@app.route('/login', methods=['POST'])
def login():
    data = request.json
    user = User.query.filter_by(username=data['username']).first()
    if user and check_password_hash(user.password, data['password']):
        token = jwt.encode({'user_id': user.id, 'exp':
datetime.datetime.utcnow() + datetime.timedelta(hours=24)},
app.config['SECRET_KEY'])
        return jsonify({'token': token})
    else:
        return jsonify({'message': 'Invalid credentials'}), 401

if __name__ == '__main__':
    app.run(debug=True)
```

Обробка запитів до бази даних та логування подій сервера

```
import pymysql

connection = pymysql.connect(
    host='localhost',
    user='username',
    password='password',
    database='db_name'
)

def get_all_products():
    with connection.cursor() as cursor:
        sql = "SELECT * FROM products"
        cursor.execute(sql)
        result = cursor.fetchall()
        return result

def add_product(name, price, description):
    with connection.cursor() as cursor:
        sql = "INSERT INTO products (name, price, description) VALUES
(%s, %s, %s)"
        cursor.execute(sql, (name, price, description))
        connection.commit()

def update_product(id, name, price, description):
    with connection.cursor() as cursor:
        sql = "UPDATE products SET name=%s, price=%s, description=%s
WHERE id=%s"
        cursor.execute(sql, (name, price, description, id))
        connection.commit()

def delete_product(id):
```

```
    with connection.cursor() as cursor:
        sql = "DELETE FROM products WHERE id=%s"
        cursor.execute(sql, (id,))
        connection.commit()


import logging

logging.basicConfig(filename='server.log',          level=logging.INFO,
format='%(asctime)s - %(levelname)s - %(message)s')

def log_event(event_message):
    logging.info(event_message)

log_event("Server started")
```