

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ З
ПРОДАЖУ ПОСЛУГ РОЗРОБКИ ЛОГОТИПУ
SOFTWARE IMPLEMENTATION OF AN ONLINE STORE FOR SELLING
LOGO DESIGN SERVICES

Виконав: студент(ка) 4 курсу, групи КНТ-111сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

БАТАНОВА М.С.

(прізвище та ініціали)

Керівник ШИТІКОВА О.В.

(прізвище та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БАТАНОВОЇ Марії Сергіївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інтернет-магазину з продажу послуг розробки логотипу. Software Implementation Of An Online Store For Selling Logo Design Services

керівник проєкту (роботи) к.т.н., ШИТІКОВА Олена Вікторівна.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “24” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 01 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Характеристика об'єкта проєктування. 2. Вибір і обґрунтування структури системи. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво системного адміністратора. 6. Керівництво користувача

5. Перелік графічного матеріалу (із точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ШИТІКОВА О.В., доцент каф.ПЗ		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач каф.ПЗ		

7. Дата видачі завдання “ 24 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 5, 6
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент

Марія БАТАНОВА
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

Олена ШИТІКОВА
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра: 92 с., 36 рис., 5 табл., 15 джерел.

ЛОГОТИП, ІНТЕРНЕТ-МАГАЗИН, ПРОДАЖ, БАЗА ДАНИХ, SQL – ЗАПИТИ, JAVA-SCRIPT, КЛЮЧОВЕ ПОЛЕ, MONGODB.

Об'єкт дослідження – процеси електронної комерції.

Предмет дослідження – інтернет-магазин з продажу послуг розробки логотипу.

Мета роботи – розробка програмного продукту, який реалізує функції інтернет-магазину з індивідуальним наповненням.

Вхідними даними для Інтернет-магазину є інформація про послуги, які продаються в інтернет-магазині логотипів.

Вихідними даними для Інтернет-магазину є сконфігуровані сторінки сайту з інформацією про набори послуг розробки логотипу, які продаються в інтернет-магазині та замовлення, зроблені користувачем.

Інтернет магазин розроблений мовою програмування Java-Script із застосуванням системи керування базами даних MongoDB. Вебдодаток коректно працює й відображається у всіх сучасних браузерах.

Висновки. Розроблений вебдодаток забезпечить клієнтам зручний доступ до високоякісних послуг з розробки унікальних логотипів, сприяючи їхньому успіху та визнанню на ринку.

Галузь використання – вебдодаток з продажу послуг розробки логотипів може обслуговувати різні сектори бізнесу та сприяти розвитку їх брендової ідентичності.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 92 pages, 36 figures, 5 tables, 15 sources.

LOGO, ONLINE STORE, SALE, DATABASE, SQL - QUERIES, JAVA-SCRIPT, PRIMARY KEY, MONGODB.

The object of research is electronic commerce processes.

The subject of the study is an online store selling logo development services.

The purpose of the work is to develop a software product that implements the functions of an online store with individual content.

The input data for the online store is information about the services that are sold in the logo online store.

The initial data for the online store are configured website pages with information about the logo design service sets sold in the online store and orders placed by the user.

The online store is developed in the Java-Script programming language using the MongoDB database management system. The web application works correctly and is displayed in all modern browsers.

Conclusions. The developed web application will provide customers with convenient access to high-quality services for the development of unique logos, contributing to their success and recognition in the market.

Industry of Use – A logo design service sales web application can serve various business sectors and help develop their brand identity.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Характеристика об'єкта проєктування	12
1.1 Поняття інтернет-магазину	12
1.2 Основні способи ведення бізнесу за допомогою системи онлайн-покупок.....	15
1.3 Огляд існуючих аналогів онлайн-платформ з продажу послуг розробки логотипу	20
1.4 Основні функціональні можливості інтернет-магазину	24
1.5 Висновки за розділом	25
2 Вибір і обґрунтування засобів та середовища розробки.....	26
2.1 Вибір середовища програмування для серверної частини сайту.....	26
2.2 Вибір середовища програмування для клієнтської частини сайту	31
2.3 Бази даних	36
2.5 Висновки за розділом	39
3 Основні рішення щодо реалізації компонентів системи.....	40
3.1 Функціональна схема роботи інтернет-магазину	40
3.2 Принцип роботи кошика товарів у інтернет-магазині	41
3.3 Алгоритм роботи головної сторінки інтернет-магазину.....	41
3.4 Алгоритм функціонування авторизації.....	43
3.5 Висновки за розділом	44
4 Керівництво програміста.....	45
4.1 Призначення й умови використання інтернет-магазину з продажу послуг розробки логотипу	45
4.2 Характеристики розробленого продукту	46
4.3 Структура бази даних	49
4.4 Структурна схема компонентів інтернет-магазину	50

4.5 Висновки за розділом	51
5 Керівництво системного адміністратора	52
5.1 Розгортання інтернет-магазину	52
5.2 Створення віртуального хоста на сервері.....	52
5.3 Розгортання бази даних вебсайту.....	53
5.4 Висновки за розділом	56
6 Керівництво користувача	57
6.1 Призначення сайту	57
6.2 Вхід до вебсайту та процес авторизації й замовлення товару.....	57
6.3 Реалізація особистого кабінету користувача.....	61
6.4 Висновки за розділом	63
Висновки	64
Перелік джерел посилання	65
Додаток А Текст програми.....	67
Додаток Б Слайди презентації	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

B2C – Business-to-consume;

C2B – Consumer-to-business;

HTML – HyperText Markup Language;

IE – Internet Explorer;

JS – JavaScript;

PHP – Hypertext Preprocessor;

БД – база даних;

ОС – операційна система;

ПО – програмне забезпечення;

СКБД – система керування базами даних;

ЦП – центральний процесор.

ВСТУП

На сьогоднішній день автоматизація продаж через інтернет-магазини в Україні набуває ключового значення, надаючи безліч переваг для компаній, що прагнуть до ефективного та конкурентоспроможного бізнесу [1]-[3].

Розширення ринку стає основною перевагою автоматизації. Вона дозволяє компаніям досягти світового рівня, звертаючись до аудиторії не лише в Україні, а й за її межами. Це відкриває нові можливості для зростання та розвитку.

Крім того, автоматизація продажів сприяє підвищенню ефективності бізнесу. Звільнення від рутинних операцій, таких як обробка замовлень та управління інвентарем, дозволяє компаніям сконцентруватись на більш важливих завданнях, таких як стратегічне планування та маркетингові ініціативи. Крім того, автоматизація онлайн-магазинів може позитивно впливати на продаж послуг розробки логотипів, забезпечуючи швидкий доступ до інформації про доступні пакети послуг, зручну систему замовлення та спрощену оплату, що збільшує ймовірність покупки [1]-[3].

Аналітика та прогнозування стають більш точними та ефективними завдяки зібраним даним про покупки та поведінку клієнтів. Це дозволяє компаніям адаптувати свої стратегії продажів, передбачати попит та покращувати свій асортимент.

Нарешті, автоматизація продажів сприяє зниженню витрат на бізнес. Вкладення в системи автоматизації можуть здатися значними, проте в довгостроковій перспективі вони окупаються за рахунок зменшення витрат на персонал та оптимізації процесів.

Автоматизація продажів через інтернет-магазини стає невід'ємною частиною бізнесу в Україні у 2024 році. Переваги цього підходу очевидні та різноманітні: від розширення ринку до підвищення ефективності та покращення обслуговування клієнтів. Компанії, що вміло використовують

можливості автоматизації, можуть очікувати стійкого зростання та успіху в довгостроковій перспективі [1]-[3].

Об'єктом дослідження є процеси електронної комерції.

Предмет дослідження інтернет-магазин з продажу послуг розробки логотипу.

Мета роботи розробка програмного продукту, який реалізує функції інтернет-магазину з індивідуальним наповненням.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- провести аналіз сучасних тенденцій і технологій у сфері електронної комерції;
- провести проєктування архітектури інтернет-магазину з урахуванням функціональних та нефункціональних вимог;
- вивчити вимоги користувачів до функціоналу та дизайну інтернет-магазинів;
- розробити технічне завдання для створення онлайн-магазину з продажу послуг розробки логотипів;
- обрати та обґрунтувати вибір технологій та інструментів для розробки вебдодатку;
- реалізувати структуру бази даних для зберігання інформації про послуги та замовлення;
- розробити інтерфейс користувача, який забезпечить зручну навігацію та замовлення послуг;
- підготувати документацію та інструкції для користувачів і адміністраторів системи.

Результатом цієї дипломної кваліфікаційної роботи буде створення власного інтернет-магазину з продажу послуг розробки логотипу. У такий час активної автоматизації різних бізнесів ринок послуг з продажем готової упаковки, в яку входять: логотип, назва та ілюстрація для інтернет-магазину - дуже затребуваний.

Інтернет-магазин здійснюватиме продаж готових пакетів послуг, оптимізуватиме роботу працівників магазину, надаватиме вибір стати покупцем чи продавцем послуг, матиме функції комунікації покупця із продавцем за допомогою онлайн-чату [1]-[3].

Для розробки Інтернет-магазину було обрано мову розмітки гіпертекстових сторінок HTML 5, для оформлення стилів додатку було обрано мову візуального представлення html-документів CSS 3. Для реалізації функціоналу Вебдодатку клієнт-серверної архітектури було обрано платформу Node.js. Для реалізації функціональних ідей на стороні клієнта було обрано фреймворк мови JavaScript – React.js.

1 ХАРАКТЕРИСТИКА ОБ'ЄКТА ПРОЄКТУВАННЯ

1.1 Поняття інтернет-магазину

Інтернет-магазин – це електронна торговельна платформа, яка дозволяє споживачам здійснювати покупки товарів та послуг через Інтернет. Ці магазини можуть бути відокремленими вебсайтами або частиною більшої торговельної платформи, і вони надають зручну можливість покупцям здійснювати покупки без присутності у фізичному магазині [3]-[4].

Інтернет-покупки – це діяльність або дія, пов'язана з купівлею товарів чи послуг через Інтернет. Цей спосіб покупок дозволяє здійснювати порівняння цін, вибирати товари з широкого асортименту, отримувати зручні доставки та користуватися різноманітними способами оплати [4].

Термін включає не лише купівлю речей в Інтернеті, а й пошук їх в Інтернеті.

Інтернет-покупки стали надзвичайно популярними завдяки зручності, доступності та широкому вибору товарів, які можна придбати онлайн.

Сьогодні в Інтернеті можна знайти майже будь-що для покупки. Фактично, фахівці з роздрібної торгівлі передбачають, що онлайн-шопінг незабаром стане більш популярним, ніж традиційні покупки за готівку.

Інтернет-магазини є частиною "E-Commerce", що означає електронна комерція [5].

Електронна комерція (E-Commerce) - це процес купівлі та продажу товарів і послуг через Інтернет. Електронна комерція дозволяє компаніям та споживачам здійснювати торгівлю з будь-якого місця, де є доступ до Інтернету, і надає можливість широкого вибору товарів і послуг, зручної оплати та доставки.

При покупці товару через інтернет покупець робить електронний правочин через інтернет [5].

Електронний правочин – дія особи, спрямована на придбання, зміну або припинення цивільних прав та обов'язків, здійснену з використанням інформаційно-телекомунікаційних систем (пп. 7 ч. 1 ст. 3 Закону № 675) [6].

Електронні угоди у сфері електронної торгівлі стосуються бізнесу через Інтернет, де товари купують та продаються дистанційно шляхом укладання угод за допомогою інформаційно-телекомунікаційних систем.

Електронна комерція, як визначено у Законі № 675 [6], охоплює відносини, спрямовані на здобуття прибутку через угоди з придбання, зміни або припинення цивільних прав та обов'язків, які здійснюються дистанційно за допомогою інформаційно-телекомунікаційних систем, і призводять до виникнення прав та обов'язків майнового характеру у її учасників.

Купівля в Інтернеті стає можливою, коли клієнт виконує покупку за допомогою цифрової платформи. Прикладом такої платформи є shopify.com, це універсальна торгова платформа для пошуку, дизайну, друку та продажу ваших товарів у Канаді. Вам не потрібно жити в цій країні, щоб придбати один із продуктів, які продає цей магазин. Глобальне покриття є однією з головних переваг онлайн-покупок [7].

Головна привабливість полягає в тому, що ми можемо знайти та купити потрібні нам речі, не виходячи з дому.

На сьогоднішній день ключовим гравцем у сфері електронної комерції є Сполучені Штати Америки.

По-перше, багато з найбільших та найвідоміших платформ електронної торгівлі, таких як Amazon, eBay та Walmart, базуються в США. Ці компанії мають значний вплив як на внутрішній, так і на міжнародний ринок, завдяки своїм широким асортиментам товарів, зручним сервісом та інноваційним підходам до електронної торгівлі [7].

Крім того, в США існують сприятливі умови для розвитку електронної комерції. Національна інфраструктура Інтернету в країні досить розвинена, що забезпечує швидкий та надійний доступ до мережі для

мільйонів користувачів. Також в США існують великі ринки споживчих товарів та послуг, що стимулює попит на електронну торгівлю.

Додатково, великий рівень технологічних інновацій у Сполучених Штатах сприяє розвитку новітніх технологій у сфері електронної комерції, таких як штучний інтелект, розпізнавання образів та аналіз даних, що покращують користувацький досвід та ефективність продажів.

У 2024 році спостерігається різке зростання обсягів продажів у сфері електронної комерції [7].

Цей ріст може бути пояснений декількома факторами.

По-перше, продовжується тенденція до зростання використання Інтернету та мобільних пристроїв. За останні роки споживачі все більше віддають перевагу зручності та швидкості онлайн-покупок, а також розширенню можливостей онлайн-шопінгу [7].

По-друге, впровадження новітніх технологій, таких як штучний інтелект, аналітика даних та персоналізовані рекомендації, дозволяє покращити користувацький досвід і зробити процес покупок більш ефективним.

По-третє, глобальна криза, така як пандемія COVID-19, підштовхнула багатьох споживачів до переходу до онлайн-покупок, змушуючи більше компаній розглядати можливості електронної комерції для забезпечення неперервного доступу до своїх товарів та послуг.

Усі ці фактори в сукупності стимулюють різке зростання продажів електронної комерції у 2024 році, що робить цей період перспективним для розвитку та інвестування в цей сектор [7].

1.2 Основні способи ведення бізнесу за допомогою системи онлайн-покупок

Сьогодні структура комерційних відносин відповідно змінюється у силу того, що десятки мільйонів персональних комп'ютерів підключаються до інтернету, об'єднуючи підприємства одне з одним та з усіма споживачами.

Старі компанії мають невеликий досвід використання можливостей електронної комерції та в результаті програють у конкурентній боротьбі.

Хоч Україна і відстає від західних країн за рівнем електронної комерції, сьогодні український інтернетбізнес розвивається досить активно, навіть швидше, ніж у країнах із вищим рівнем доходу.

У відповідь на стрімке зростання кількості користувачів, український ринок повторює світові тенденції розвитку глобальної мережі та робить успіхи у збільшенні, інтеграції та зниженні вартості послуг. Розроблено спеціальну нормативну базу [6]-[7], без якої неможливий розвиток електронної комерції в Україні. Також, вчені та практики [6]-[7], приділяють багато уваги методам та моделям ведення бізнесу в мережі інтернет, результати узагальнення представлено у таблиці 1.1.

Таблиця 1.1 – Бізнес-моделі електронного бізнесу

Бізнес-модель	Опис
B2C (business to consumer)	Бізнес для споживача
B2B (business to business)	Бізнес для бізнесу
C2C (consumer to consumer)	Бізнес, що забезпечує взаємодію між споживачами

Продовження таблиці 1.1

Бізнес-модель	Опис
C2B (consumer to business)	Системи опрацювання цінових заявок, за якими споживачі хочуть придбати товарні послуги
B2A (business to administration)	Бізнес між приватними компаніями і урядовими організаціями
C2A (consumer to administration)	Бізнес фізичних осіб та державних служб
G2C (government to consumer)	Надання урядовими установами послуг кінцевим споживачам через інтернет
C2G (consumer to government)	Надання громадянами послуг державі
G2G (government-to-government)	Автоматизація відносин і документообігу між державними відомствами
B2G (business-to-government)	Надання комерційними компаніями товарів та послуг державним установам
G2B (government-to-business)	Надання послуг державними органами бізнес-організаціям (автоматизація розрахунків за податками, сертифікацію)

Що краще зрозуміти дану таблицю потрібно заглибитися в кожне з понять і що воно з себе представляє.

B2C (бізнес до споживача) - це модель бізнесу, в якій підприємство прямо продає товари або послуги кінцевому споживачеві через онлайн-магазини, вебсайти або інші канали зв'язку. Спрямованно на задоволення потреб та вимог індивідуальних споживачів, і взаємодіє з ними безпосередньо. Наприклад, електронні роздрібні магазини, сервіси доставки їжі, онлайн-платформи для покупки книг або електроніки - все це приклади B2C бізнесу [6]-[7].

B2B (бізнес для бізнесу) - це модель бізнесу, в якій одне підприємство надає товари або послуги іншому підприємству. У цій моделі бізнес спрямований на задоволення потреб та вимог інших компаній або організацій.

C2C (споживач до споживача) - це модель бізнесу, в якій споживачі продають товари або послуги один одному через спеціалізовані онлайн-платформи або ринки. Бізнес спрямований на взаємодію між індивідуальними споживачами, які самостійно виступають як продавці та покупці [6]-[7].

C2B (споживач до бізнесу) - це модель бізнесу, в якій індивідуальні споживачі або користувачі надають товари або послуги бізнесам. Споживачі виступають як продавці, а бізнеси як покупці. Наприклад, фриланс-платформи, де фахівці пропонують свої послуги компаніям або іншим клієнтам, або платформи для створення та продажу контенту, які користувачі продають бізнесам для використання в їхній рекламі або маркетингових кампаніях.

B2A (бізнес до адміністрації) - це модель бізнесу, в якій підприємства надають товари або послуги державним установам або адміністрації. Бізнес спрямований на задоволення потреб та вимог урядових органів. Наприклад, компанії, які постачають програмне забезпечення для урядових відомств, або послуги консалтингу для державних установ.

C2A (споживач до адміністрації) - це модель взаємодії, при якій індивідуальні споживачі надають свої товари або послуги державним органам або адміністрації. Споживачі виступають як постачальники, а адміністрація як отримувач. Наприклад, фізичні особи, які подають документи для отримання дозволів або послуг від урядових органів, або надають свої відгуки та пропозиції стосовно роботи адміністрації.

G2C (уряд до споживача) - це модель взаємодії, при якій уряд надає послуги та інформацію безпосередньо споживачам через онлайн-канали. Уряд виступає як постачальник, а індивідуальні громадяни як отримувачі.

Наприклад, онлайн-платформи для оплати податків або електронні сервіси для отримання державних послуг є прикладами такої взаємодії.

C2G (споживач до уряду) - це коли громадяни використовують онлайн-засоби для спілкування з урядовими органами, надаючи їм інформацію, звертаючись з запитаннями або подавати заяви. Громадяни виступають як постачальники, а урядові органи - як отримувачі. Наприклад, це може бути подача заяв онлайн або надсилання скарг через вебпортали уряду [6]-[7].

G2G (уряд-до-уряду) - це коли урядові органи взаємодіють між собою за допомогою онлайн-систем, обмінюючись інформацією та здійснюючи різні операції. Один орган виступає як постачальник послуг або інформації, а інший - як отримувач. Наприклад, це може бути електронний обмін даними між міністерствами або онлайн-передача інформації для здійснення різних адміністративних процедур між урядовими структурами.

B2G (бізнес-до-уряду) - коли бізнес-структури надають товари або послуги безпосередньо державним установам або органам через онлайн-платформи. Бізнес виступає як постачальник, а уряд як отримувач. Наприклад, це може бути постачання програмного забезпечення для урядових відомств або надання будівельних послуг державним органам.

G2B (уряд-до-бізнесу) - коли урядові органи надають послуги або інформацію приватним компаніям або підприємствам через інтернетові ресурси або інші електронні засоби. Уряд виступає як постачальник, а бізнес як отримувач. Наприклад, це може включати в себе електронну реєстрацію бізнесу, вебплатформи для торгівлі з урядовими закупівлями або доступ до інформації про регуляторні вимоги.

Повертаючись до аналізу світового ринку, можна дійти висновку, що модель B2B є найпоширенішою та основною серед компаній, що працюють на міжкорпоративному ринку, де одні юридичні особи роблять послуги і продають товари іншим юридичним особам [6]-[7].

Самим поширеним методом ведення бізнесу в онлайн просторі є інтернет-магазин. Це вебсайт, на якому встановлена спеціальна програма, відома як "Віртуальний магазин". Тут представлені товари, рекламні матеріали, форма для замовлення та варіанти оплати.

В інтернет-магазині користувачі можуть вибрати товар, оформити замовлення, сплатити за нього за допомогою електронних платіжних систем або іншими способами, вказаними на сайті продавця. Інтернет-магазини зазвичай використовують прямий продаж, тобто організацію продажу товарів від виробника безпосередньо до споживача. Можна сказати, що це класичний бізнес, тільки майже вся його діяльність ведеться в онлайн-середовищі.

Наступний варіант ведення бізнесу в інтернеті – створення лендінгу. Лендінги дещо схожі з інтернет-магазинами. Як і інтернет-магазини, лендінги створюються для продажу певних товарів через сайт. На відміну від перших, лендінги складаються тільки з однієї сторінки.

Досить часто вітчизняні компанії використовують для просування свого бізнесу різні платформи і сайти оголошень. Вони можуть використовуватися як основне джерело генерації продажів, так і як додаткове, особливо на етапі розвитку бізнесу [6]-[7].

Соціальні мережі в плані продажів мають досить обмежений функціонал, але користуються значним попитом на сьогоднішній день. Суть цього методу полягає в тому, щоб створити публічну сторінку або спільноту, працювати над збільшенням кількості учасників та пропонувати товари або послуги залученій аудиторії. Такий спосіб ведення бізнесу зазвичай використовується у сегменті B2C.

Електронна комерція включає всі напрямки ділової активності, що здійснюється в кіберпросторі.

Ведення бізнесу в інтернеті дозволяє компаніям пропонувати конкурентні переваги пропозицій, які знижують витрати, розширюють ринки та сфери бізнесу, а також знаходять нові канали продажу, залучають нових клієнтів та покращують їх обслуговування. Перевагою електронної комерції є

її мобільність та оперативність, завдяки яким приймаються ефективні управлінські рішення.

Таким чином, вивчення специфіки розвитку та функціонування ринку електронної комерції ґрунтується безпосередньо на вивченні інформаційного середовища. Тож виділення інтернет-середовища як основи торгового простору електронної комерції необхідно проаналізувати її організаційну структуру, характер взаємодії учасників ринку, специфіку конкурентного середовища та інфраструктури [6]-[7].

Отже можна сказати, що за допомогою використання методів ведення бізнесу в мережі інтернет, компанії можуть отримувати додатковий дохід від продажу фізичних чи електронних товарів, знижувати витрати та скорочувати час транзакцій у різних сферах діяльності: у дистрибуції товарів, в управлінні взаємовідносинами з клієнтами, у створенні нових продуктів, в поширенні інформації тощо.

Підприємства, які прагнуть створити або зміцнити свої власні бренди, використовують інтернет як канал маркетингових комунікацій, постійного діалогу та довгострокових відносин з клієнтами, поширення інформації та багато іншого.

1.3 Огляд існуючих аналогів онлайн-платформ з продажу послуг розробки логотипу

У сучасному світі електронної комерції все більше компаній і підприємців шукають шляхи для створення унікальних і привабливих брендових ідентичностей.

Одним із ключових елементів бренду є логотип, який допомагає компанії виділитися на ринку та запам'ятатися клієнтам. Зростаючий попит на професійні послуги з розробки логотипів спричинив появу численних онлайн-платформ, що спеціалізуються на наданні таких послуг.

У цьому розділі розглянемо основні існуючі аналоги українських онлайн-платформ з продажу послуг розробки логотипів, їхні особливості та функціональні можливості [6]-[7].

Freelance.ua – одна з найпопулярніших українських платформ для фріланс-послуг, де можна знайти професійних дизайнерів для розробки логотипів.

Переваги Freelance.ua включають широкий вибір виконавців, різноманітні цінові категорії та можливість перегляду портфоліо дизайнерів. Платформа також пропонує зручну систему оплати та забезпечує комунікацію між клієнтом і виконавцем на всіх етапах проекту.

Вигляд вебсайту Freelance.ua зображено на рис 1.1.

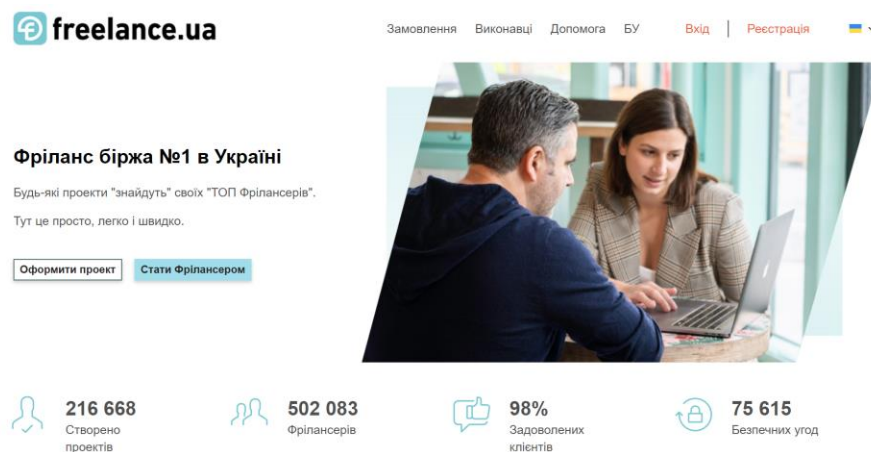


Рисунок 1.1 - Freelance.ua

Kabanchik.ua – платформа, що надає можливість замовляти послуги у фрілансерів з різних сфер, включаючи графічний дизайн.

Клієнти можуть створювати завдання для розробки логотипів, на які відгукуються дизайнери з усієї України.

Перевагою Kabanchik.ua є велика кількість виконавців та зручна система оцінок, що допомагає обрати найкращого спеціаліста для виконання замовлення.

Вигляд вебсайту Kabanchik.ua зображено на рис 1.2.

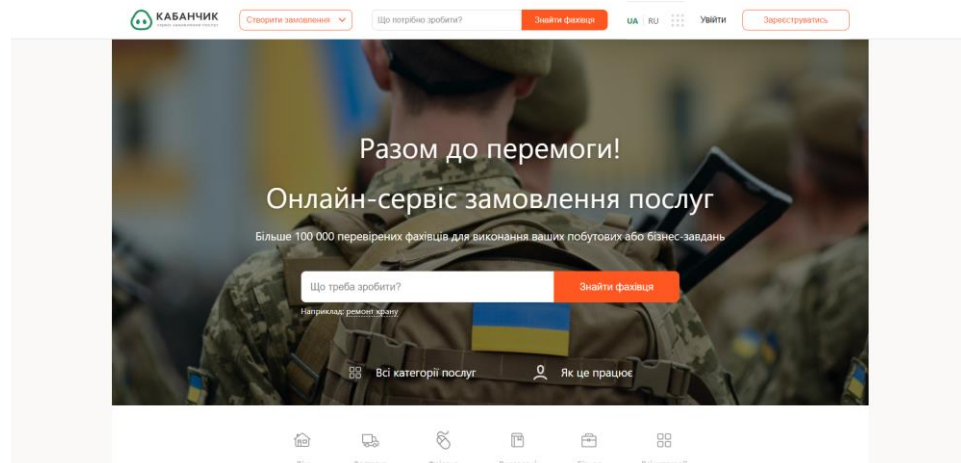


Рисунок 1.2 - Kabanchik.ua

Weblancer.net – це одна популярна платформа для фрілансерів в Україні, яка пропонує послуги з розробки логотипів. Клієнти можуть розміщувати проєкти, отримувати пропозиції від дизайнерів, переглядати їхні портфоліо та вибирати найкращого виконавця.

Платформа забезпечує зручну комунікацію та систему оплати, що дозволяє ефективно керувати проєктами.

Вигляд вебсайту Weblancer.net зображено на рис 1.3.

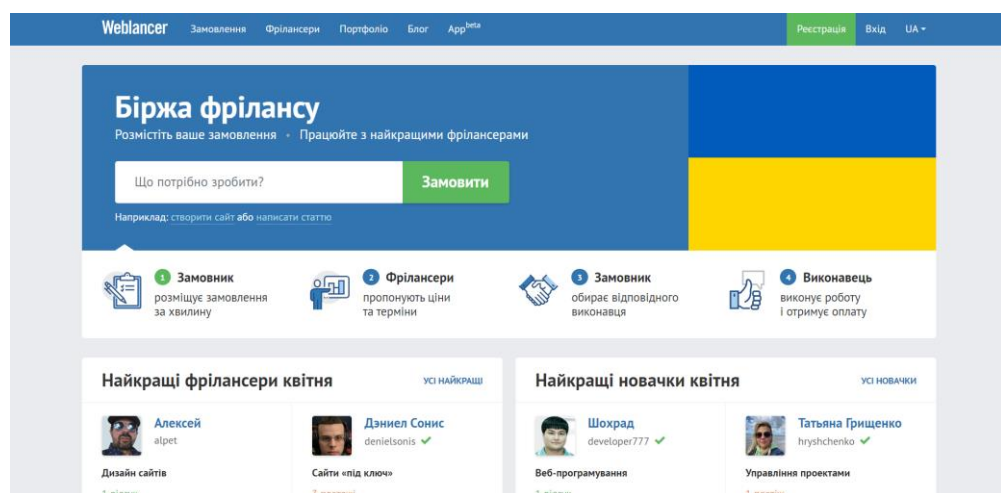


Рисунок 1.3 - Weblancer.net

Designer.ua – спеціалізована платформа для дизайнерів, яка пропонує послуги з розробки логотипів, брендингу та іншої графічної продукції [6]-[7].

Перевагою цієї платформи є високий рівень професіоналізму виконавців, що гарантує якісне виконання замовлень. Designer.ua також надає можливість перегляду портфоліо дизайнерів та зручну систему оплати.

Вигляд вебсайту Weblancer.net зображено на рис 1.4.

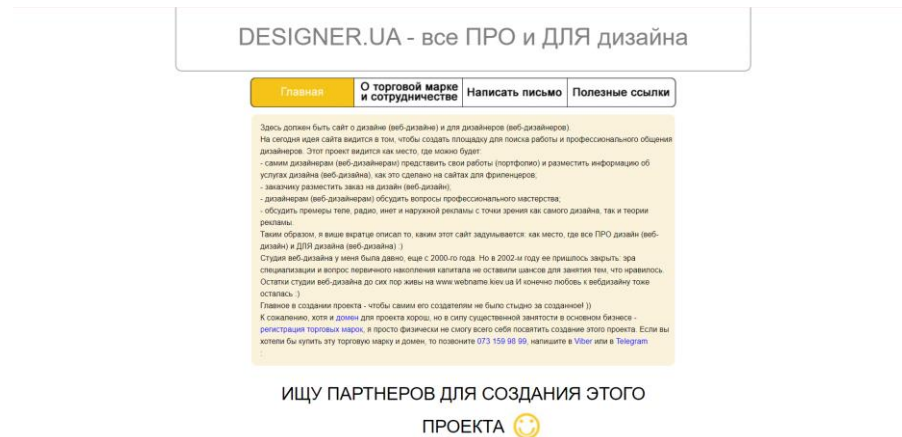


Рисунок 1.4 - Designer.ua

Таблицю порівняння існуючих аналогів онлайн-платформ неавдено нижче у таблиці 1.2.

Таблиця 1.2 – Порівняння існуючих аналогів онлайн-платформ таких як Freelance.ua, Kabanchik.ua, Weblancer.net та Designer.ua.

Критерій	Freelance.ua	Kabanchik.ua	Weblancer.net	Designer.ua
Комісія за послуги	10-20%	5-15%	7-10%	5-10%
Підтримка користувачів	Обмежена	Цілодобова	Робочі години	Обмежена
Кількість проектів	Середнє	Високе	Середнє	Низьке
Перевірка виконавців	Є, але обмежена	Ретельна	Є, але обмежена	Є, але обмежена
Інтерфейс і зручність	Застарілий	Зручний	Сучасний, але складний	Простий, але незручний
Безпека угод	Гарантується, але з обмеженням	Повна гарантія	Обмежена гарантія	Гарантується, але з обмеженням

Ринок українських онлайн-платформ з продажу послуг розробки логотипів пропонує широкий вибір сервісів, кожен з яких має свої унікальні особливості та переваги. Від платформ для фрілансерів до спеціалізованих дизайнерських майданчиків, користувачі мають безліч можливостей для створення своїх брендових ідентичностей.

Вибір платформи залежить від індивідуальних потреб клієнта, бюджету та вимог до унікальності та якості логотипу [6]-[7].

1.4 Основні функціональні можливості інтернет-магазину

Інтернет-магазин - це вебсайт, який має спеціально розроблені функції для зручних та автоматизованих онлайн-покупок користувачів. Під час розробки такого магазину важливо звертати увагу на його функціональність, зручність в користуванні та надійність [6]-[7].

Проаналізувавши аналоги в попередньому підрозділі було вирішено у майбутньому інтернет-магазині реалізувати такі можливості:

- відображення категорій товарів;
- можливість придбання товару та додавання його у кошик;
- можливість вибору валюти;
- можливість вибору способу отримання товару;
- можливість вибору способу оплати товару;
- формування онлайн-чеку.

З використанням таких технологій, як безконтактна оплата або ідентифікація користувача за допомогою відбитку пальця або розпізнавання обличчя, інтернет-магазини мають можливість не лише конкурувати з традиційними магазинами, але і вийти вперед. Принципи функціонування інтернет-магазину надають широкі можливості для досягнення цієї мети:

- залишити відгуки о продавцях;
- ставити через електронний чат питання та отримати відповідь;
- замовлення послуги в будь який день так будь який час;

- економити власнику на оренді та зарплаті персоналу;
- робити продаж послуги без фактичної наявності.

З розгляду того, що продаж товару можна сприймати як надання послуги користувачеві, в якій зручність та корисність грають важливу роль, можна зробити висновок, що завдяки переліченим перевагам і можливостям клієнту більше не потрібно витрачати час на пошук товару, очікування у черзі та порівняння цін. Тепер він може легко перейти за посиланням в Інтернеті у будь-який зручний для нього момент, ознайомитися з широким вибором товарів, звернутися за допомогою в онлайн-чаті або прочитати відгуки, і, нарешті, зробити замовлення з використанням зручних способів оплати та доставки, будучи в будь-якій точці світу.

1.5 Висновки за розділом

Було проведено вичерпне дослідження об'єкта проєктування. Визначено основні процеси електронної комерції та принципи функціонування інтернет-магазинів. Проаналізовано ключові етапи ведення бізнесу за допомогою систем онлайн-покупок.

Крім того, були проаналізовані існуючі аналоги інтернет-магазинів з продажу такі як Kabanchik.ua, Freelance.ua, Weblancer.net та Designer.ua, виділені їх основні функціональні можливості, що відзначаються своєю перевагою перед традиційними магазинами.

2 ВИБІР І ОБҐРУНТУВАННЯ ЗАСОБІВ ТА СЕРЕДОВИЩА РОЗРОБКИ

2.1 Вибір середовища програмування для серверної частини сайту

Для розробки цього вебдодатку була обрана мова JavaScript, оскільки вона дозволяє створювати динамічні та інтерактивні інтерфейси користувача.

Використання JavaScript забезпечує широкі можливості для реалізації функціоналу та забезпечує високу швидкість виконання вебдодатка [8].

JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування, яка найчастіше використовується для створення сценаріїв Вебсторінок. Дає можливість змінення структури та зовнішнього вигляду сторінки, асинхронного обмінювання даних з сервером та керування браузером. JS характеризують як прототипну, скриптову мову програмування з динамічною типізацією. Є реалізацією стандарту ECMAScript. На даний момент реалізовує стандарт ES6 [8].

Таблицю порівняння мови JavaScript з іншими відомими мовами програмування такими як JAVA та Python наведено нижче таблиця 2.1.

Таблиця 2.1 – Порівняння можливостей JavaScript, PHP та JAVA.

№ п/п	Можливості	JavaScript	PHP	Java
1	2	3	4	5
1	Використання фреймворків	JavaScript має доступ до таких фреймворків, як React, Angular, Vue	Користувачі PHP мають доступ до таких фреймворків як Laravel, Zend, Yii	Користувачі Java мають доступ до фреймворків, як Spring, Spring boot, Hybernate
2	Популярність використання	Вибір мови для веброзробки, особливо для фронтенду, але також і для бекенду з Node.js	Вибір мови для веброзробки	Широко використовується в додатках безпеки, додатках Andoid, вебдодатках AI

Продовження таблиці 2.1

№ п/п	Можливості	JavaScript	PHP	Java
1	2	3	4	5
3	Підключення до бази даних	Можливий доступ до різних баз даних за допомогою бібліотек, таких як Sequelize, Mongoose	Можливий доступ до більш ніж 20 різних баз даних.	Можливий доступ майже до всіх баз даних.
4	Крива здатності освоєння мови	JavaScript відрізняється дуже низькою кривою навчання, легко почати.	PHP відрізняється дуже низькою кривою навчання, легко почати. Вивчати	Більше підходить для довгострокових проєктів
5	Майбутнє мови програмування	Довготривале, оскільки JavaScript широко використовується у веброзробці	Довготривале, понад 70% вебсайтів створено за допомогою PHP	Безперечно довготривале завдяки своїй безпеці та швидкості
6	Читабельність	Мова JavaScript не дуже типізована і дотримується класичного підходу	Мова PHP не дуже типізована і дотримується класичного підходу	Нелегко читати.
7	Спеціалізація мови	Спеціалізована мова для програмування веброзробки.	Спеціалізована мова для програмування веброзробки.	Мова програмування Java – це універсальна, одночасна, строго типізована, об'єктно-орієнтована мова на основі класів.
8	Синтаксис	Вбудована бібліотека містить широкий вибір умов імен	Вбудована бібліотека містить широкий вибір умов імен	Синтаксис чистий, але не простий, як Python

Ці переваги роблять JavaScript ідеальним вибором для створення сучасних, динамічних і інтерактивних вебдодатків, що відповідають вимогам дипломного проєкту [8].

Сценарій - це послідовність дій або інструкцій, які програма виконує для досягнення певної мети або реалізації певного функціоналу.

Мова сценаріїв – це мова програмування, яка використовується для написання сценаріїв або скриптів, які виконують певні дії або операції. Скрипти зазвичай вбудовуються в інші програмні середовища.

Метою скриптів зазвичай є підвищення продуктивності або виконання рутинних завдань для програми.

Серверні скрипти виконуються на сервері, у той час як клієнтські сценарії виконуються у браузері користувача.

JavaScript є сценарієм, що виконується на стороні клієнта і інтерпретується браузером, у той час як PHP - це сценарій, що виконується на сервері і інтерпретується на сервері. Обидва можуть бути вбудовані в сторінки HTML [8].

Ілюстрацію функціонування мови JavaScript на рисунку 2.1.

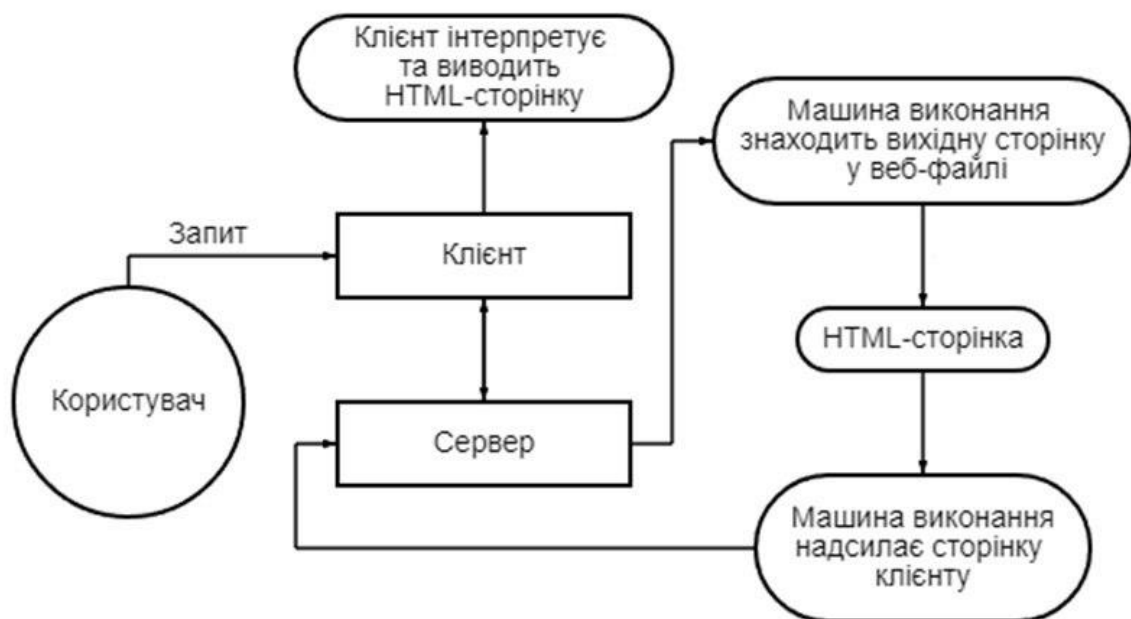


Рисунок 2.1 – Блок-схема функціонування мови JavaScript

Популярність та розповсюдженість.

JavaScript є основною мовою для веброзробки, особливо для фронтенду, і також активно використовується для бекенду завдяки Node.js. Це робить його універсальним вибором для створення як інтерфейсів користувача, так і серверних додатків [8].

Низька крива навчання.

JavaScript відрізняється дуже низькою кривою навчання, що дозволяє швидко розпочати роботу та ефективно використовувати його для реалізації різних функцій. Це особливо важливо при обмеженому часі для розробки дипломного проєкту.

Широкий вибір фреймворків та бібліотек.

JavaScript має багатий набір фреймворків та бібліотек, таких як React, Angular, Vue для фронтенду та Node.js для бекенду. Це дозволяє швидко і ефективно створювати потужні та масштабовані додатки.

Кросплатформеність.

JavaScript працює у всіх сучасних браузерах і може використовуватися для розробки кросплатформених додатків, що забезпечує широку доступність і зручність використання результатів роботи на різних пристроях.

Активна спільнота та підтримка.

JavaScript має одну з найбільших і найактивніших спільнот розробників, що забезпечує широкий доступ до навчальних матеріалів, інструментів, плагінів та підтримки. Це робить процес розробки ефективнішим та більш керованим.

Динамічність і гнучкість.

JavaScript дозволяє створювати динамічні та інтерактивні вебдодатки, що забезпечує більш високий рівень залучення користувачів і покращує їх взаємодію з вебсайтом чи додатком.

Можливості асинхронного програмування.

Засоби, такі як проміси та асинхронні функції, дозволяють ефективно керувати асинхронними операціями, що робить JavaScript ідеальним для розробки додатків з високою швидкістю.

Можливості мобільної розробки.

JavaScript дозволяє використовувати різні фреймворки та платформи для створення мобільних додатків, таких як React Native або Ionic, що спрощує розробку кросплатформених додатків для різних мобільних пристроїв.

Розвинуті можливості обробки даних.

За допомогою бібліотек і фреймворків, таких як D3.js або Chart.js, JavaScript може бути використаний для створення складних візуалізацій даних, що дозволяє користувачам зрозуміти та аналізувати великі обсяги інформації зручним способом [8].

Таким чином, JavaScript стане надійним фундаментом для реалізації вебпроєкту.

Node.js – це платформа з відкритим початковим кодом для виконання високопродуктивних мережових застосунків, написаних мовою JavaScript. Node.js дає можливість виконання JS-скриптів на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання.

Node.js має наступні властивості:

- асинхронна одно-поточкова модель виконання запитів;
- неблокуюче введення/виведення;
- система модулів CommonJS;
- JavaScript Google V8;
- керування модулями за допомогою npm (node packet manager).

React.js - це потужна бібліотека JavaScript, яка широко використовується для розробки інтерфейсів користувача вебдодатків.

Особливості React.js включають у себе:

- компонентний підхід: react дозволяє будувати вебінтерфейси шляхом створення малих ізольованих компонентів, які можна повторно використовувати і складати разом для створення складних інтерфейсів;
- віртуальний dom: react використовує віртуальний dom для ефективного оновлення вебсторінок. він автоматично визначає, які частини сторінки потрібно оновити, що забезпечує оптимальну швидкодію додатка;
- jsx синтаксис: react використовує jsx - розширений синтаксис javascript, що дозволяє писати html-подібні розмітки безпосередньо всередині javascript коду, що полегшує створення компонентів;
- односторінкові додатки: react ідеально підходить для створення односторінкових додатків (spa), оскільки дозволяє ефективно керувати станом додатка та рендерити компоненти залежно від змін;
- розширені можливості: завдяки величезній спільноті та екосистемі, react має багато розширень, бібліотек та інструментів, що полегшують розробку та підтримку вебдодатків.

SPA - це скорочення від "Single Page Application" (односторінковий додаток). Це тип вебдодатка, який працює в браузері та замість того, щоб завантажувати нові сторінки з сервера при переході на нові URL-адреси, він динамічно оновлює лише частини сторінки за допомогою JavaScript. Такий підхід дозволяє створити додатки, які працюють швидше та більш плавно, оскільки вони завантажують лише ті дані, які потрібні в даний момент, та не потребують повного перезавантаження сторінки.

2.2 Вибір середовища програмування для клієнтської частини сайту

Для створення, редагування та налагодження програми було обрано середовище розробки Visual Studio Code [9].

Visual Studio Code – засіб для створення, редагування та налагодження Вебзастосунків та програм для хмарних систем. Visual Studio

Code доступний у версіях для платформ Windows, Linux та OS X. За основу VS Code використовується вільний редактор коду – Atom.

Visual Studio Code простий в освоєнні, наділений зручним інтерфейсом (рисунок 2.2) і всіма необхідними функціями для створення додатків та можливістю встановлення додаткових розширень.

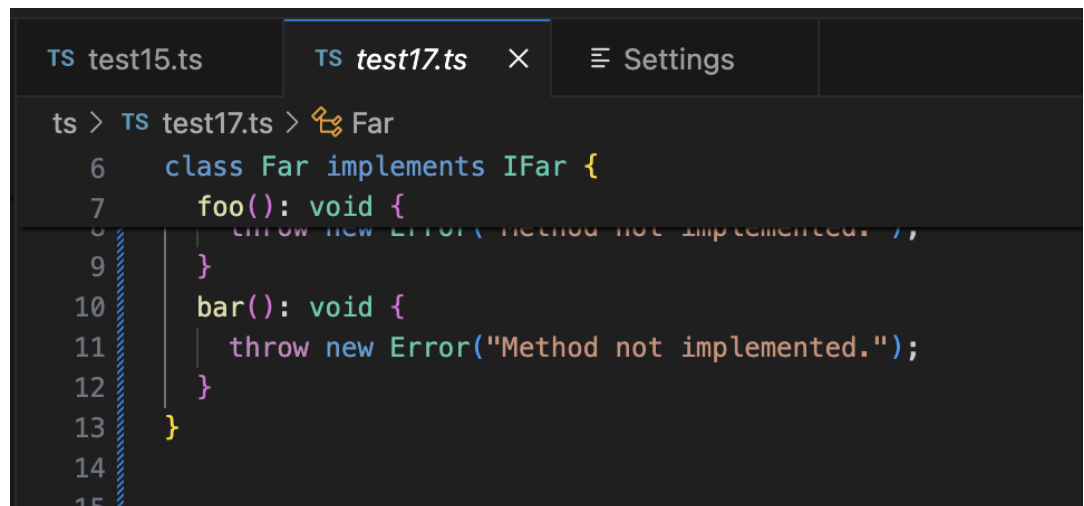


Рисунок 2.2 – Інтерфейс Visual Studio Code

Так само Visual Studio Code включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга.

Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації.

У Visual Studio Code підтримуються такі мови і технології як: JavaScript, C++, C#, TypeScript, PHP, Python, XML, Java, Objective-C, PowerShell, Visual Basic, JSON, HTML, CSS, LESS і SASS [9].

Таблицю порівняння Visual Studio Code з іншими редакторами такими як Sublime Text та JetBrains IntelliJ IDEA неавдено нище у таблиці 2.2.

Таблиця 2.2 – Порівняння основних переваг Visual Studio Code, Sublime Text та JetBrains IntelliJ IDEA.

Критерій	Visual Studio Code	Sublime Text	JetBrains IntelliJ IDEA
Ціна	Безкоштовно	\$80 (одноразово)	Від \$149 на рік (є безкоштовні версії для студентів та Open Source)
Відкритий вихідний код	Так	Ні	Ні
Підтримка плагінів	Так	Так	Так
Продуктивність	Висока, але іноді ресурсомістка	Дуже висока	Висока
Підтримка мов	Широка	Широка	Широка (особливо для JVM-мов)
Інтеграція з Git	Вбудована	Плагін	Вбудована
Редактор коду	Потужні функції редагування	Основні функції	Потужні функції
Налагодження	Вбудоване налагодження	Потребує плагінів	Вбудоване налагодження
Автодоповнення	Потужне та інтелектуальне	Основне	Дуже потужне та інтелектуальне
Інтерфейс користувача	Інтуїтивно зрозумілий	Мінімалістичний	Багатий функціонал
Спільнота та підтримка	Дуже велика спільнота	Середня спільнота	Велика спільнота
Кросплатформеність	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux

Для розробки Вебдодатку було обрано мову розмітки гіпертекстових сторінок HTML 5, для оформлення стилів додатку було обрано мову візуального представлення html-документів CSS 3.

HTML (англ. HyperText Markup Language) – це стандартна мова розмітки для створення Вебсторінок та Вебдодатків. За допомогою неї розробник визначає як мультимедіа, текст, гіперпосилання будуть відображені серед іншого контенту у браузері. З CSS та JavaScript, вона утворює тріаду основних технологій для Веброзробки [10].

На даний момент актуальна версія – HTML5, опублікована 28 жовтня 2014 року, є рекомендованим стандартом. Ціль розробки HTML5 – покращення рівня підтримки мультимедіа-технологій та скорочення використання заснованих на плагінах технологій, як Adobe Flash, Microsoft Silverlight і Sun JavaFX. Специфікації HTML5 не обмежуються тільки розміткою і включають в себе низку Вебтехнологій, котрі у сукупності формують відкриту Вебплатформу – програмне оточення для роботи крос-платформових застосунків, здатних взаємодіяти з обладнанням, і які підтримують засоби для роботи з відео, графікою і анімацією, що надає розширені можливості. Нові можливості HTML5:

- з нової версії розмітки прибрано близько 15 тегів;
- було додано елементи header, footer, nav, aside, section, article;
- підтримка тегів video, audio;
- робота з векторною графікою;
- збереження даних на стороні клієнта.

CSS (англ. Cascading Style Sheets) – це спеціальна мова стилізації Вебсторінки. CSS використовують для візуальної презентації Вебсторінок написаних на HTML (визначення кольорів, шрифтів, верстки та інших аспектів вигляду сторінки). За допомогою неї розробник визначає які стилі буде містити той чи інший елемент сайту. Вона являє собою набір властивостей, які описуються розробником залежно від того які стилі вимагає макет Вебсторінки. CSS (каскадна, або блочна верстка) прийшла на заміну табличній верстці Вебсторінок і визначеним стандартом [11].

У процесі розробки було прийнято рішення використовувати один з процесорів CSS, а саме SASS. Завдяки цьому процесу є можливість

структурувати CSS-код більш організовано за допомогою вкладених правил, що допомогло зберігати логічну структуру стилів та зробило їх легшими для читання і підтримки. Використання змінних дозволило легко змінювати стилі глобально, забезпечуючи консистентність в усьому проєкті, а міксини та функції зменшили кількість дублювання коду. Крім того, підтримка імпорту файлів сприяла розбиттю стилів на окремі модулі, що зробило роботу зі стилями більш структурованою та легкою для масштабування.

SASS (Syntactically Awesome Stylesheets) — це розширення для CSS, яке додає потужні функціональні можливості та полегшує процес написання стилів.

Основні переваги SASS включають:

- змінні дозволяють зберігати значення властивостей у змінних для повторного використання, що спрощує управління стилями і забезпечує консистентність;
- вкладеність дозволяє писати css-код у вигляді вкладених правил, що робить структуру стилів більш логічною та читабельною;
- міксини дозволяють створювати блоки стилів, які можна повторно використовувати в різних частинах файлу, зменшуючи кількість дублювання коду;
- успадкування (extends) дозволяє одному селектору успадковувати стилі іншого, що допомагає уникнути дублювання коду і спрощує підтримку;
- функції та оператори дозволяють виконувати різноманітні операції прямо в css, такі як математичні обчислення, маніпуляції з кольорами та інше.

SASS компілюється в стандартний CSS, що означає, що всі ці додаткові функції використовуються під час розробки, а в кінцевому результаті ви отримуєте звичайний CSS, який підтримується всіма браузерами.

2.3 Бази даних

Для реалізації бази даних випускної кваліфікаційної роботи було обрано MongoDB.

MongoDB – є документною базою даних, яка зберігає дані у форматі JSON-подібних документів. Вона використовує гнучку структуру документів для зберігання та організації даних, що дозволяє швидко розширювати та змінювати схему даних. MongoDB підтримує ряд функцій, включаючи індексацію, запити та агрегацію даних, реплікацію та кластеризацію для забезпечення високої доступності та масштабованості. Вона широко використовується для розробки вебдодатків, аналізу даних, а також для зберігання та обробки великих обсягів даних.

Основна одиниця даних у MongoDB - це колекція, яка містить набір документів. Кожен документ може мати різну структуру, тобто він може містити різні поля з різними типами даних. MongoDB не вимагає заздалегідь визначеної схеми, що робить її гнучкою для зберігання та обробки різноманітних даних. Дані можуть бути структуровані у вкладені об'єкти, масиви та інші складні структури, що дозволяє ефективно моделювати різні сценарії додатків.

MongoDB є однією з найбільш впливових технологій у сфері баз даних у сучасному світі. Ця документна орієнтована система управління базами даних (СКБД) відкрила нові можливості для розробників, пропонуючи різноманітний та швидкий підхід до зберігання та обробки даних.

Однією з ключових особливостей MongoDB є його гнучкість у використанні. Відсутність фіксованої схеми дозволяє розробникам легко змінювати структуру даних без необхідності проведення складних міграцій. Це особливо корисно у випадку змінюваних вимог або постійно зростаючих проєктів [14].

MongoDB також славиться своєю високою продуктивністю та масштабованістю. Її архітектура дозволяє легко розширювати ресурси для відповіді на зростаючі потреби проєкту. Це робить MongoDB ідеальним вибором для великих вебдодатків та послуг з великим обсягом даних.

Крім того, MongoDB пропонує широкий набір функціональності для роботи з даними, включаючи розподілені запити, індексацію, агрегацію та безпеку. Це забезпечує високу надійність та ефективність роботи з базою даних.

Таблиця порівнянь MongoDB з іншими відомими базами даних, такими як MySQL та PostgreSQL наведена нижче у таблиці 2.3.

Таблиця 2.3 – Порівняння критерій MongoDB, MySQL та PostgreSQL.

Критерій	MongoDB	MySQL	PostgreSQL
Гнучкість схеми	Гнучка схема	Статична схема	Гнучка схема
Горизонтальне масштабування	Підтримується	Обмежена підтримка	Підтримується
Підтримка JSON	Так	Ні	Ні
Мова запитів	JSON-подібні	SQL	SQL
Швидкість операцій	Висока	Залежить від операції	Залежить від операції
Підтримка транзакцій	Так	Так	Так
Строки або колонки	Колекції	Таблиці	Таблиці
Проміжний код	JavaScript	SQL	SQL
Підтримка ACID	Часткова	Повна	Повна
Реплікація та шардування	Підтримується	Обмежена підтримка	Підтримується

MongoDB є однією з найінноваційніших систем управління базами даних (СКБД) у сучасному програмуванні та інформаційних технологіях. Вона відкриває новий рівень можливостей для розробників і дозволяє їм ефективно працювати з великими обсягами даних у різних сценаріях.

Устрій MongoDB [14].

MongoDB складається з декількох ключових компонентів, які спільно забезпечують його функціональність:

MongoDB складається з двох частин: сервера і клієнта.

Сервер баз даних (mongod): Це основний процес MongoDB, який відповідає за зберігання та керування даними. Він обслуговує запити від клієнтів та виконує операції з даними у відповідь на ці запити.

Оболонка MongoDB (mongo): Це інтерактивна оболонка командного рядка, яка дозволяє взаємодіяти з сервером баз даних MongoDB. Вона надає можливість виконувати запити, адмініструвати бази даних та виконувати інші дії безпосередньо з командного рядка.

Клієнтські бібліотеки та драйвери: MongoDB підтримує різноманітні клієнтські бібліотеки та драйвери для різних мов програмування, таких як Python, JavaScript, Java, Ruby та інші. Вони надають програмістам зручний інтерфейс для взаємодії з базою даних з їхніх програм.

MongoDB Compass: Це графічний інтерфейс користувача для адміністрування та роботи з базою даних MongoDB. Він надає можливість візуально досліджувати дані, створювати запити та керувати конфігурацією бази даних [14].

Реплікація та шардування: MongoDB надає можливості для створення реплікаційних наборів та шардованих кластерів для забезпечення високої доступності, масштабованості та стійкості до відмов.

Інструменти для адміністрування та моніторингу: MongoDB постачається з набором інструментів для адміністрування, моніторингу та налагодження баз даних, таких як mongostat, mongotop, MongoDB Atlas тощо.

Всі ці компоненти разом утворюють повнофункціональну систему управління базами даних, яка відповідає потребам сучасних розробників та адміністраторів баз даних.

2.5 Висновки за розділом

В данному розділі було обрано середовище розробки, що відповідає критеріям, зазначеним при постановці завдання, та сформований необхідний набір інструментів для вирішення проєктного завдання.

Було проведено обґрунтування вибору мови програмування та середовища функціонування системи.

Для розробки проєкту було обрано мову JavaScript для розробки серверної частини сайту, HTML та CSS було обрано для клієнтської частини інтернет-магазину.

В якості бази даних було обрано СКБД MongoDB.

Вибрано найбільш відповідну базу даних для розв'язання задачі проєктування.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Функціональна схема роботи інтернет-магазину

Алгоритм роботи інтернет-магазину для нефізичних товарів може бути узагальнений наступним чином: спочатку проходять реєстрацію на сайті, створюючи обліковий запис з унікальними даними входу, потім обирає бажану послугу або цифровий продукт і додає його до кошика. Після цього він переходить до оформлення замовлення, де вказує свої контактні дані та здійснює оплату обраного товару або послуги. Після підтвердження замовлення система автоматично надсилає користувачеві необхідні електронні ключі або посилання для отримання придбаного цифрового продукту. Користувач отримує доступ до придбаного товару або послуги, а також може залишити відгук чи оцінку, функціональна схема роботи інтернет-магазину зображена на рисунку 3.1.



Рисунок 3.1 – Функціональна схема роботи інтернет-магазину

3.2 Принцип роботи кошика товарів у інтернет-магазині

Кошик товарів у інтернет-магазині формується автоматично разом з головною сторінкою. Його функція полягає в тому, щоб користувач міг обирати та резервувати товари, що його зацікавили, і пізніше оформити замовлення [15].

Якщо кошик порожній, користувач може продовжити вибір товарів, а коли він містить товари, можна переходити до оформлення замовлення.

Текст функції замовлення було наведено в додатку А.1.

В першу чергу оформлення замовлення в магазині починається з формування сторінки кошика, сам вміст сторінки складається з послуг, які були обрані користувачем [15].

Продовження замовлення в магазині відбувається при переході на наступну сторінку, де користувач повинен погодитися зі змістом замовлення та ввести контактну інформацію у вигляді пошти(якщо він ще не зареєстрований), щоб з ним мали змогу зв'язатися для підтвердження та відправки послуги адміністратори.

Після того, як користувач закінчує додавати товари до кошика і оформлює замовлення, відображається сторінка підтвердження з повідомленням про успішне замовлення. Після цього дані замовлення додаються до бази даних для подальшої обробки.

3.3 Алгоритм роботи головної сторінки інтернет-магазину

Після запуску сайту у його корневій папці шукається файл App.js. Цей файл є головною сторінкою інтернет-магазину. Інтернет-магазин виконує код цього файлу та передає результати замовнику у вигляді скомпільованої сторінки [15].

На початку виконання App.js проводиться певний аналіз параметрів адресного рядка, робить запит до бази даних інтернет-магазину та формується сторінка на основі інформації з бази даних.

Схематичний алгоритм генерації головної сторінки зображено на рисунку 3.2.

Функція підключення до бази даних було наведено в додатку А.2.

Нижче на рисунку 3.2 подано блок-схему. Спочатку користувач намагається підключитися до бази даних. В залежності від результату цієї спроби і аналізу адресного рядка, виконується відповідна дія: якщо підключення успішне, генерується головна сторінка магазину разом зі стилями; якщо підключення невдачне або адресний рядок не вірний, формується повідомлення про неможливість відображення головної сторінки.

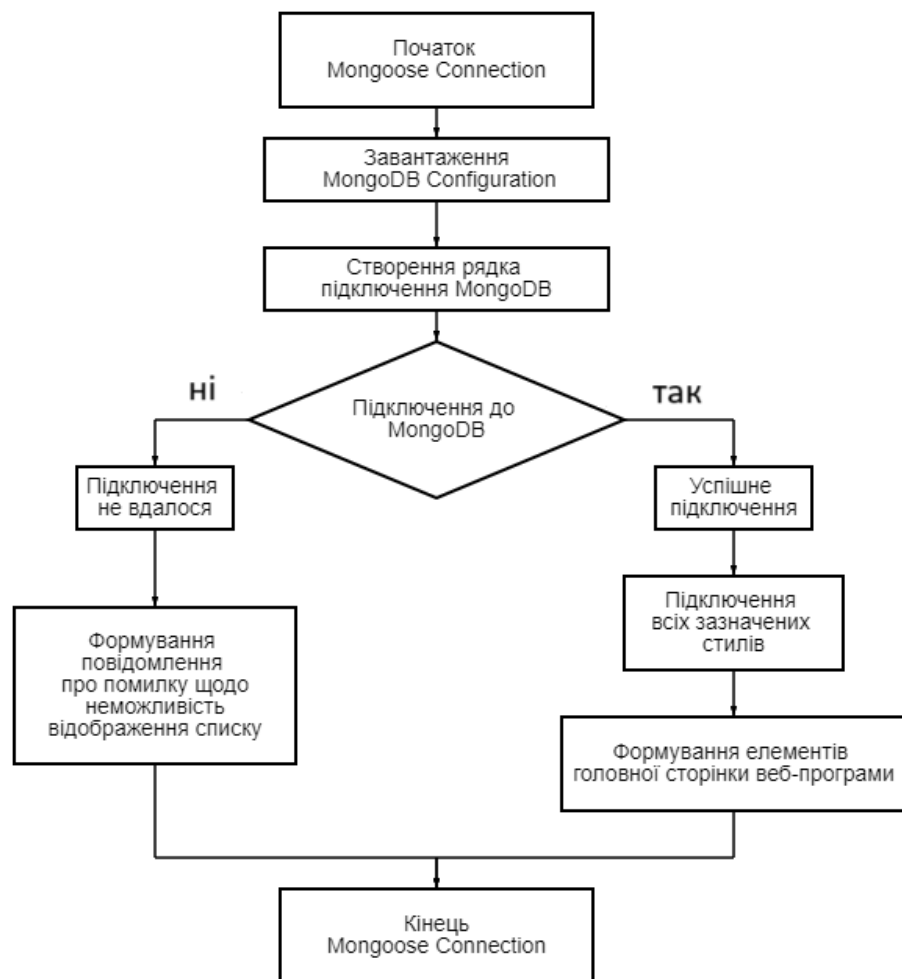


Рисунок 3.2 – Блок-схема генерації головної сторінки

3.4 Алгоритм функціонування авторизації

У рамках розробки вебсайту була створена панель авторизації, яка сприяє оптимізації процесу керування контентом сайту та відстеження покупок.

Для початку необхідно пройти процедуру авторизації. Цей процес включає введення облікових даних, які перевіряються на відповідність. У разі вірної верифікації відбувається підключення до бази даних, після чого користувач отримує доступ до особистого кабінету. У разі некоректного введення облікових даних користувач отримає повідомлення про помилку.

Функціонування панелі авторизації наведено в додатку А.3:

Блок-схема функціонування панелі авторизації наведена на рисунку 3.3.

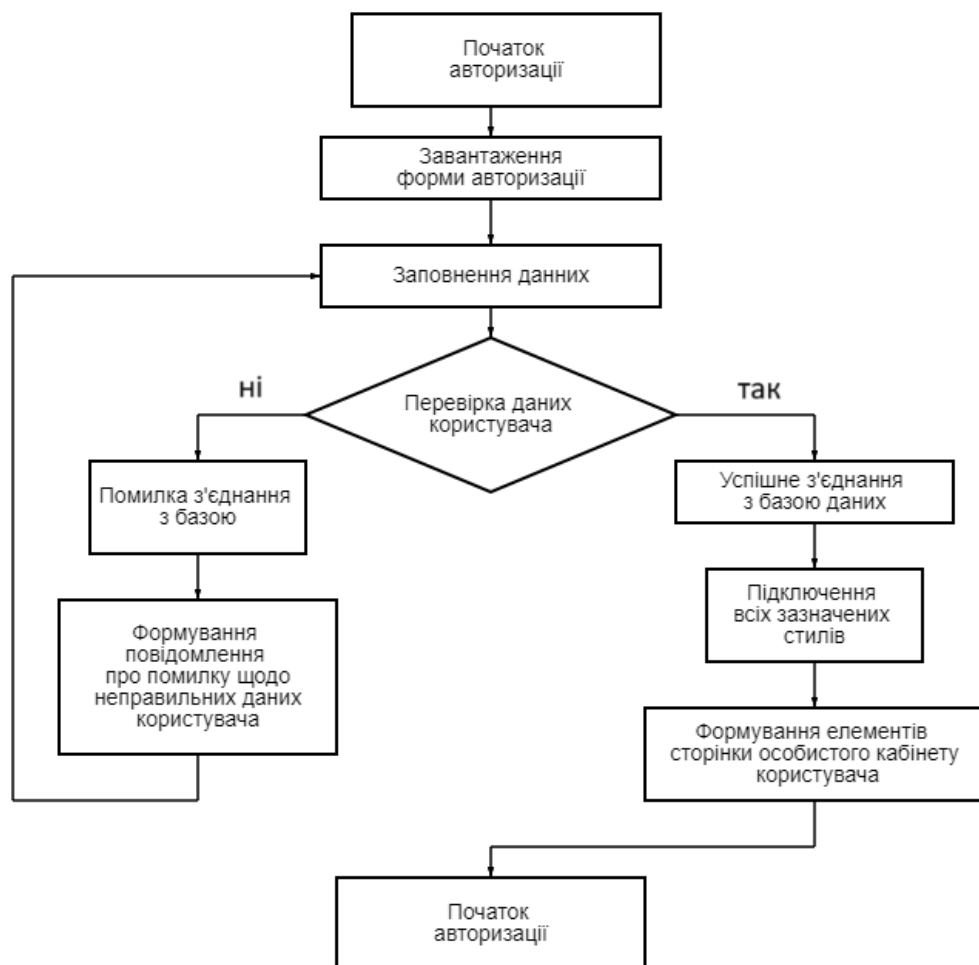


Рисунок 3.3 – Блок-схема функціонування панелі авторизації

Описуючи нижче наведену блок-схему функціонування панелі авторизації, зазначимо наступне: у першому блоці здійснюється перевірка параметрів авторизації користувача на основі введених ним даних. Якщо дані були введені правильно, проводиться подальша перевірка. У разі коректності даних здійснюється успішне з'єднання з базою даних, після чого відображаються відповідні стилі. Нарешті відбувається остаточне формування сторінки особистого кабінету користувача. Якщо ж дані були введені неправильно, формується повідомлення про некоректність введених даних.

3.5 Висновки за розділом

В даному розділі представлено основну інформацію про розробку інтернет-магазину.

Описані ключові алгоритми, які були використані під час реалізації проєкту, зокрема алгоритм роботи кошика магазину та алгоритм функціонування головної сторінки магазину. Розроблено принципи роботи адміністративної панелі магазину.

Надано загальну характеристику створеного вебзастосунку та розглянуто основні блоки й процедури його функціонування.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання інтернет-магазину з продажу послуг розробки логотипу

Вебсайт призначений для автоматизації процесу продажу послуг.

Серверна частина сайту була розроблена за допомогою мови програмування JavaScript, мови гіпертекстової розмітки HTML, каскадної таблиці стилів CSS. У якості бази даних використовується СКБД MongoDB.

Для функціонування розробленого програмного продукту, а також в разі його допрацювання, необхідно мати таку мінімальну апаратну платформу:

- CPU: Intel Core i5;
- RAM: 1 Gb;
- HDD/SSD: 1Gb (мінімально, для початкового запуску), але при розростанні, збільшенні кількості користувачів, тем та відповідей, для зберігання бази даних може знадобитися чималий дисковий простір;
- порти: 80 (HTTP);
- програмне забезпечення: середовище Node.js, менеджер пакетів NPM або Yarn;
- програмне забезпечення для автоматизації розгортання та керування додатками: Docker.

На персональному комп'ютері повинно бути встановлено наступне програмне забезпечення: будь яка операційна система, наприклад Windows 7/8/8.1/10/11, програма була розроблена за допомогою операційної системи Windows 11, сучасний інтернет браузер, наприклад Google Chrome, Opera, Mozilla Firefox, Internet Explorer чи інші, пріоритетним вибором браузера стане Google Chrome.

4.2 Характеристики розробленого продукту

При розробці вебдодатку були реалізовані наступні логічні блоки, які описують певний функціонал інтернет-магазину. Основними такими логічними частинами є:

- App.js - основний логічний блок з обробки сайту;
- AgreeTermOfServiceInput.jsx - погодження з умовами надання послуг;
- Chat.jsx - блок, що відповідає за спілкування між користувачами;
- UserProfile.jsx - блок, що формує сторінку особистого кабінету користувача;
- Home.jsx - блок, PayForm.jsx що формує сторінку відображення списку послуг;
- mongoose.js - блок, який відповідає за з'єднання з базою даних.

Нижче будуть наведені певні логічні процедури наявні в блоках.

Функція отримання згоди з правилами надання послуг, ця функція в першу чергу необхідна для отримання постачальнику послуги згоду на те, що його робота буде оплачена і всі авторські права будуть збережені, код функції наведено в додатку А.4.

Нижче на рисунку 4.1 представлена блок-схема функції отримання згоди з правилами надання послуг.

Початково користувач переглядає вікно з викладеними правилами надання послуг. Якщо після ознайомлення з цими правилами користувач встановлює позначку у відповідному чекбоксі, це вважається наданням згоди, і функція завершується. У випадку, якщо користувач не погоджується з правилами, на екрані з'являється повідомлення щодо відмови від послуги, і функція припиняє свою роботу.



Рисунок 4.1 – Блок-схема функції отримання згоди з правилами надання послуг

Функція реєстрації користувача що потрібна для того щоб новий користувач магазину мав змогу зареєструватися, код функції наведено в додатку А.5.

Нижче на рисунку 4.2 зображена блок-схема функція реєстрації користувача.

Спочатку користувач надає свої авторизаційні дані. Після цього встановлюється зв'язок із базою даних. Після підключення до бази даних виконується перевірка введених даних. Якщо перевірка успішна, інформація зберігається у базі даних. У разі невдалої перевірки сторінка оновлюється.

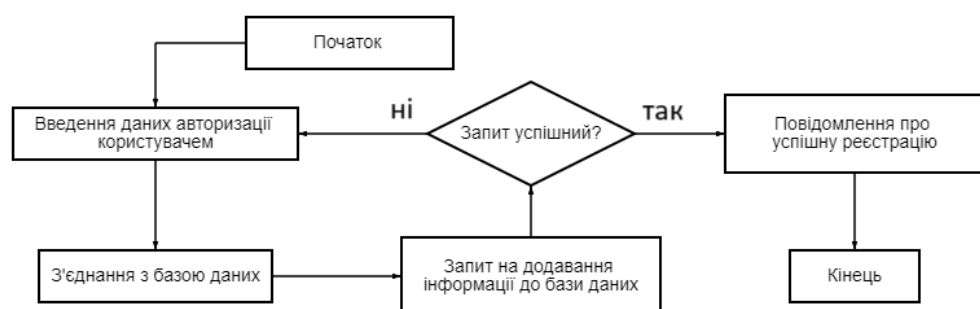


Рисунок 4.2 – Блок-схема функції реєстрації користувача

Функція оплати товару користувачем була розроблена для того щоб користувач сайту мав змогу оплатити обрану послугу, код функції наведено в додатку А.6.

На рисунку 4.3 зображену блок-схему функцію оплати послуги користувачем.

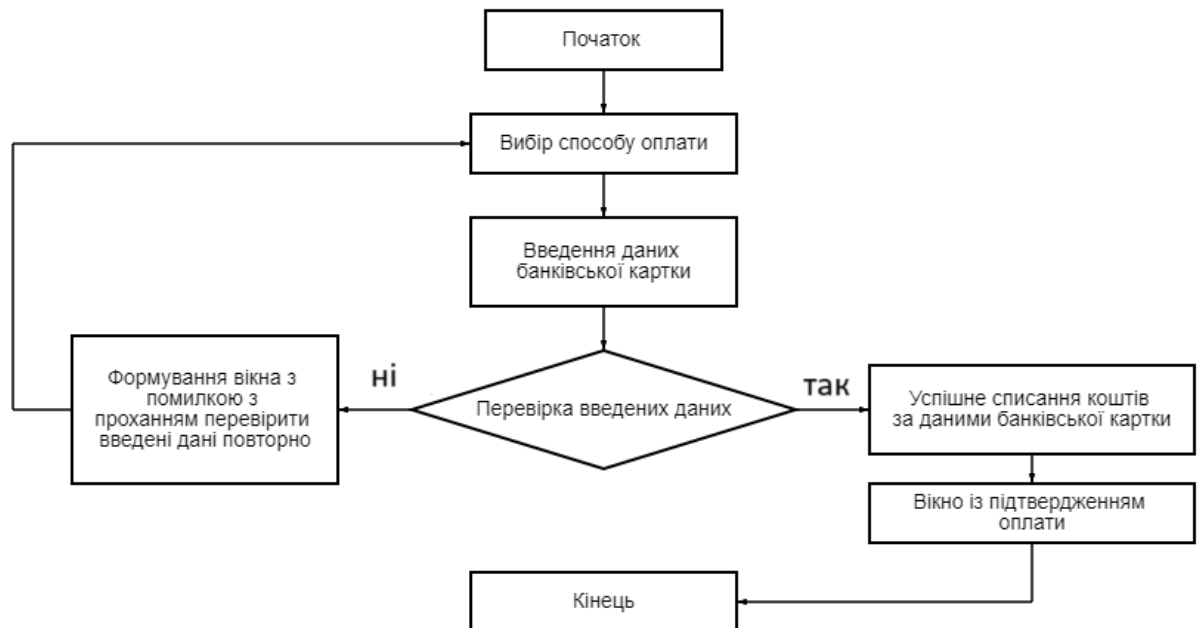


Рисунок 4.3 – Блок-схема функції додавання товару до бази даних

Ця функція потрібна для оплати вибраної послуги користувачем.

Спочатку користувач обирає зручний для нього метод оплати зі списку доступних варіантів. Далі він вводить свої банківські дані у відповідні поля форми. Якщо перевірка введених даних пройшла успішно, то реєструється оплата, і користувач отримує повідомлення про успішне здійснення оплати. У разі, якщо оплата не проходить через певні причини, користувач отримує повідомлення про помилку з проханням перевірити правильність введених даних. Після цього його повертають на початкову сторінку оплати.

Всі таблиці бази даних пов'язані одна з одною зв'язком «один до багатьох». За допомогою зовнішніх та первинних ключів таблиці зв'язані одна з одною. Таблиці бази даних мають наступні зв'язки:

- таблиця «shipment» зв'язана з таблицею «order_to_shipment» полями «id»;
- таблиця «product» зв'язана з таблицею «order» полями «code»;
- таблиця «customer» зв'язана з таблицями «shipment» та «order» полями «id»;
- таблиця «order» зв'язана з таблицею «order_to_shipment» полями «id».

4.4 Структурна схема компонентів інтернет-магазину

Якщо розглядати інтернет-магазин як вебсайт, його можна поділити на дві основні логічні частини: адміністративну та клієнтську.

Адміністративна частина дозволяє адміністратору інтернет-магазину, після входу в систему адміністрування, редагувати базу даних та додавати нові елементи (послуги). У клієнтській частині відсутня можливість редагування даних, тобто доступ до ресурсів, які можна редагувати, закритий для загального огляду.

Клієнтська частина дозволяє всім користувачам і відвідувачам інтернет-магазину переглядати, шукати, та обирати необхідні послуги, розміщені у різних категоріях. Це значно спрощує роботу користувача під час пошуку конкретної інформації. Клієнт може оформити замовлення на послугу, вибрати спосіб оплати та спосіб отримання послуги.

4.5 Висновки за розділом

У цьому розділі було розглянуто основне призначення й умови використання розроблюваного інтернет-магазину.

Описані основні логічні блоки що були розроблені у процесі розробки інтернет-магазину, та деякі логічні функції що наявні у логічних блоках.

Було розглянуто структуру бази даних первинні ключі та основні зв'язки у таблицях.

Також було розглянуто структурну схему компонентів інтернет-магазину, що демонструю логічну схему взаємодії компонентів, розроблених у магазині.

Створено структуру бази даних інтернет-магазину.

Описано структурну схему розробленої системи.

5 КЕРІВНИЦТВО СИСТЕМНОГО АДМІНІСТРАТОРА

5.1 Розгортання інтернет-магазину

Інтернет-магазин розміщено у архівному файлі під назвою «SquadHelp.zip».

Файл може бути переданий за допомогою флеш-накопичувача.

Для розміщення сайту в Інтернеті необхідно виконати такі кроки:

- створити віртуальний хост на сервері;
- розпакувати вміст архіву «squadhelp.zip» у кореневий каталог віртуального хоста;
- налаштувати базу даних сайту за допомогою системи керування базами даних mongodb;
- перейти на головну сторінку сайту для перевірки коректності його роботи.

5.2 Створення віртуального хоста на сервері

Спершу потрібно створити конфігураційний файл для віртуального хоста. Для цього, наприклад, у випадку з вебсервером Apache, конфігураційний файл може бути розташований у директорії /etc/apache2/sites-available/your_site.conf. Вміст конфігураційного файлу може виглядати наступним чином:

```
<VirtualHost 80:80>
    ServerAdmin webmaster@ squadhelp.com
    ServerName squadhelp.com
    ServerAlias www. squadhelp.com
    DocumentRoot /var/www/ squadhelp

    <Directory /var/www/ squadhelp >
        Options Indexes FollowSymLinks
        AllowOverride All
```

```

        Require all granted
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/squadhelp_error.log
    CustomLog ${APACHE_LOG_DIR}/squadhelp_access.log combined
</VirtualHost>

```

Після створення конфігураційного файлу необхідно активувати новий віртуальний хост за допомогою команди:

```
sudo a2ensite your_site.conf
```

Завершивши активацію віртуального хоста, перезапустіть вебсервер Apache для застосування змін:

```
sudo systemctl reload apache2
```

Основні директиви віртуального хоста:

«DocumentRoot» - визначає каталог, в якому розташовані файли (сторінки) цього віртуального хоста;

«ServerName» - вказує доменне ім'я віртуального хоста, за яким користувачі можуть отримати доступ до сайту;

«ErrorLog» – файл, у який записуються помилки, що виникають при роботі сайту;

«CustomLog» – файл журналу, в якому реєструються всі події та запити до сайту.

5.3 Розгортання бази даних вебсайту.

Перш ніж розгортати базу даних для вебсайту, важливо впевнитися, що на сервері запущений сервер MongoDB. Для цього використовуйте відповідну команду: «mongod».. Переконайтеся, що сервер працює стабільно і готовий приймати з'єднання.

Визначте базу даних MongoDB з ім'ям "shm-chat". Якщо така база даних ще не існує, створіть її через адміністративний інтерфейс MongoDB або використовуючи команди оболонки.

У файлі конфігурації(рис. 5.1) визначені параметри підключення до бази даних MongoDB для середовища продакшн. Переконайтеся, що налаштування бази даних дозволяють з'єднання з вебсайтом.

```
1  {
2    "development": {
3      "database": "shm-chat",
4      "host": "mongo-dev",
5      "port": 27017
6    },
7    "production": {
8      "database": "shm-chat",
9      "host": "localhost",
10     "port": 27017
11   }
12 }
```

Рисунок 5.1 - Файл конфігурації MongoDB.

Вказаний хост "localhost" показує, що база даних розгорнута на тому самому сервері, де і ваш вебсайт, а порт 27017 є типовим портом для MongoDB.

Для налаштування з'єднання вебсайту з базою даних MongoDB використовується модуль mongoose.js(рис 5.2). При запуску вебсервера виконується зчитування конфігураційних даних з файлу mongoConfig.json, що містить параметри підключення до бази даних MongoDB.

Зчитані дані використовуються для встановлення з'єднання з MongoDB через метод mongoose.connect(). Використання параметрів { useNewUrlParser: true, useUnifiedTopology: true } забезпечує сумісність з останніми версіями MongoDB та уникнення застарілих методів підключення. У випадку невдачі підключення буде виведено відповідне повідомлення про помилку на консоль, після чого процес вебсервера буде припинено.

Для полегшення відладки у режимі розробки встановлюється вивід додаткової інформації про операції бази даних за допомогою `mongoose.set('debug', true)`. Такий підхід забезпечує надійність та стабільність з'єднання вебсайту з базою даних MongoDB.

```
const mongoose = require('mongoose');
const path = require('path');
const env = process.env.NODE_ENV || 'development';
const configPath = path.join(__dirname, '..', 'config/mongoConfig.json');
const config = require(configPath)[ env ];

mongoose.connect(
  `mongodb://${ config.host }:${ config.port }/${ config.database }`,
  { useNewUrlParser: true, useUnifiedTopology: true }, (err) => {
    if (err) {
      console.log(err);
      process.exit(1);
    }
  });

mongoose.set('debug', env === 'development');

module.exports = mongoose;
```

Рисунок 5.2 - Модуль mongoose.js.

Після підключення вебсайту до бази даних MongoDB впевніться, що з'єднання встановлено коректно.

Виконайте тестові запити для перевірки правильності роботи вебсайту з базою даних MongoDB у середовищі продакшн.

Важливо провести ретельне тестування, щоб виявити та виправити всі можливі проблеми.

5.4 Висновки за розділом

В даному розділі було викладено детальний аналіз мети та умов функціонування вебсайту.

Описано процедуру розгортання електронного магазину та створення віртуального хоста на сервері.

У розділі також викладено процедуру розгортання бази даних MongoDB та описано роботу її модулів.

6 КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Призначення сайту

Вебсайт призначений для користувачів, які мають намір придбати послуги у розробці логотипів.

Серверна частина сайту була розроблена за допомогою мови програмування JavaScript, мови гіпертекстової розмітки HTML, каскадної таблиці стилів CSS. У якості бази даних використовується MongoDB.

Умовою надійної експлуатації вебсайту є наявність надійного з'єднання з мережею інтернет, а також наявність всіх модулів та бази даних на вебсервері.

Для роботи вебсайту необхідний комп'ютер з клавіатурою і мишею, ноутбук або інший пристрій, здатний відображати вебсторінки та підключатися до інтернету. Пристрій повинен мати встановлений веббраузер: Google Chrome, Opera, Mozilla Firefox, Internet Explorer або будь-який інший.

6.2 Вхід до вебсайту та процес авторизації й замовлення товару

Для відкриття даного вебсайту необхідно в полі пошуку веббраузера ввести вебадресу сайту.

Вікно вхід на сайт на рисунку 6.1.

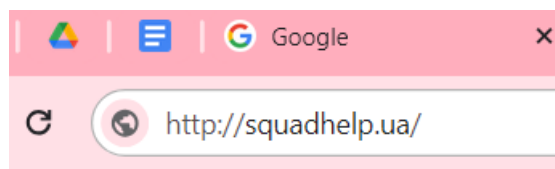


Рисунок 6.1 –Вхід до вебсайту з браузера

Після переходу за посиланням на вебсайт, користувач потрапляє на його головну сторінку. Тут розміщено основну інформацію про сайт, наведено приклади вже проданих робіт і надано вказівки щодо процесу замовлення товару. Головна сторінка вебсайту продемонстрована на рисунку 6.2.

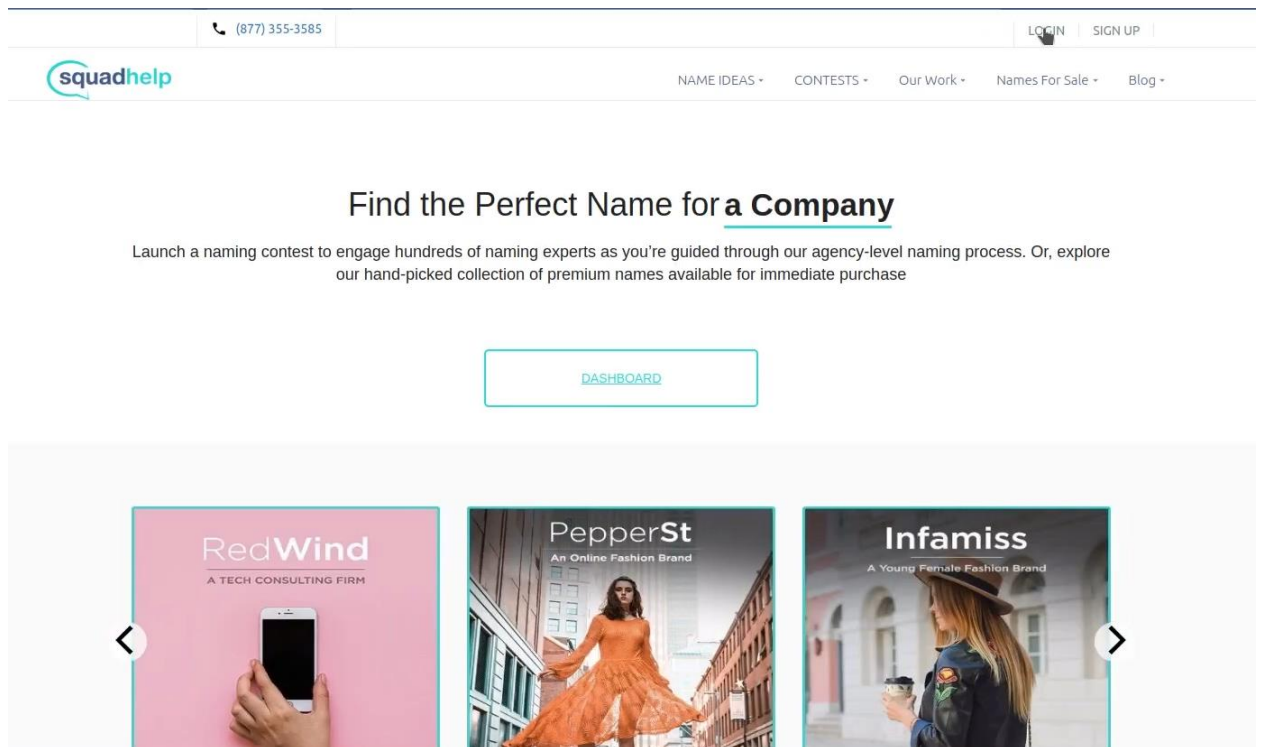


Рисунок 6.2 – Головна сторінка вебсайту

Щоб здійснити покупку, потрібно авторизуватися в необхідному вікні. Вікно авторизації вебсайту зображено на рисунку 6.3

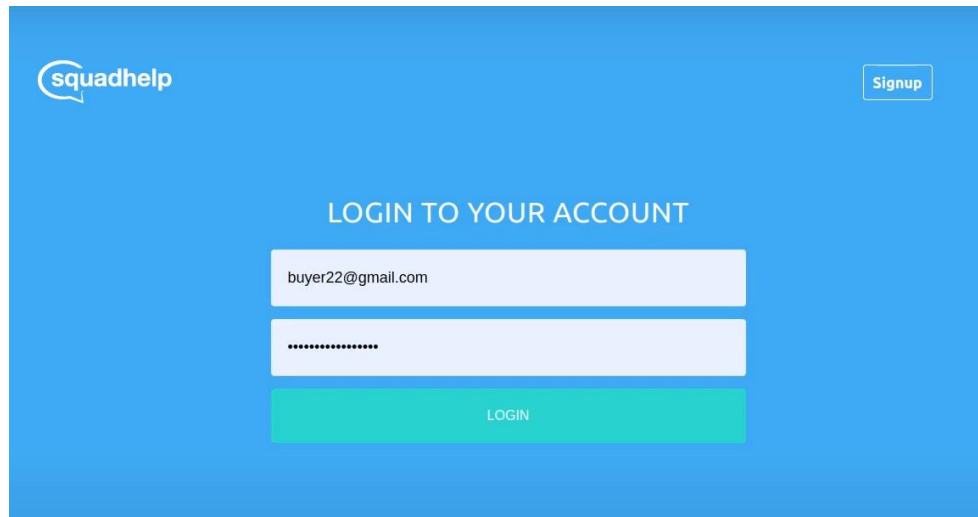
The image shows the login page of the Squadhelp website. At the top left is the Squadhelp logo, and at the top right is a 'Signup' button. The main heading is 'LOGIN TO YOUR ACCOUNT'. Below it are two input fields: the first contains the email 'buyer22@gmail.com' and the second contains a masked password '.....'. A green 'LOGIN' button is positioned below the password field.

Рисунок 6.3 – Вікно авторизації вебсайту

Якщо користувач ще не зареєстрований, він може перейти на сторінку реєстрації, яка наведена на рисунку 6.4.

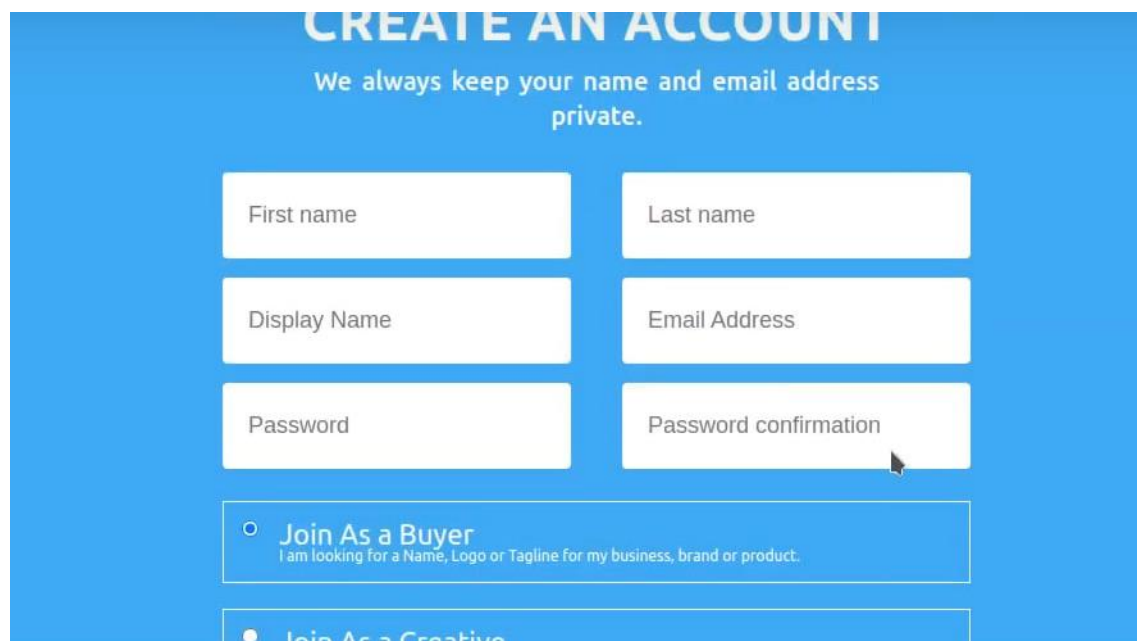
The image shows the registration page of the Squadhelp website. The heading is 'CREATE AN ACCOUNT'. Below it is a sub-heading: 'We always keep your name and email address private.' There are six input fields arranged in two columns: 'First name', 'Last name', 'Display Name', 'Email Address', 'Password', and 'Password confirmation'. Below the input fields are two radio button options: 'Join As a Buyer' (selected) and 'Join As a Creative'. The 'Join As a Buyer' option has a sub-text: 'I am looking for a Name, Logo or Tagline for my business, brand or product.'

Рисунок 6.4 – Сторінка реєстрації

Далі користувач обирає послугу у відповідній категорії. Після вибору пакета послуг йому необхідно заповнити форму оформлення замовлення, в

якій слід вказати тип послуги, тип підприємства, для якого вона призначена, сферу діяльності підприємства або компанії, короткий опис цільової аудиторії, назву підприємства, бажаний стиль кінцевого продукту та додати відповідний файл стилю. Після заповнення форми користувач натискає кнопку "Далі".

Форму оформлення замовлення зображена на рисунку 6.5.

The screenshot shows the 'LOGO' contest creation interface on the Squadhelp website. The header includes the Squadhelp logo, contact information (877) 355-3585, and user details (Hi, buyer33). Navigation links for NAME IDEAS, CONTESTS, Our Work, Names For Sale, and Blog are present, along with a 'START CONTEST' button.

The main section is titled 'LOGO' and includes a progress indicator with a green checkmark. Below this, the form consists of several sections:

- Title of contest:** A text input field with the placeholder 'Title'.
- Describe industry associated with your venture:** A dropdown menu currently showing 'Creative Agency'.
- What does your company / business do?:** A text input field with the placeholder 'e.g. We're an online lifestyle brand that'.
- What does your company / business do?:** A larger text input field containing the example text: 'e.g. We're an online lifestyle brand that provides stylish and high quality apparel to the expert eco-conscious shopper'.
- Tell us about your customers:** A text input field with the placeholder 'customers'.
- What name of your venture?:** A text input field with the placeholder 'name of venture'.
- Brand Style:** A dropdown menu currently showing 'Techy'.
- Choose File:** A button to upload a style file.

At the bottom of the form, there are 'Back' and 'Next' buttons for navigation.

Рисунок 6.5 – Вікно оформлення замовлення

Після цього користувач переходить на сторінку оплати, де йому потрібно ввести банківські дані своєї картки у спеціально розроблену форму, яка відображає введені дані на зображенні картки.

Форму оплати замовлення зображено на рисунку 6.6.

The screenshot displays a payment interface. On the left, under the heading "Payment Information", there is a visual representation of a credit card. Below the card image are input fields for "Name", "Card Number", "* Expires" (with a sub-label "expiry"), and "* Security Code" (with a sub-label "cvc"). A "Total: \$100.00" label is positioned above the "Pay Now" button. On the right side of the interface, the "Package Name: Standard" is listed with a price of "\$100 USD". Below this, the "Total: \$100.00 USD" is confirmed, followed by a link that says "Have a promo code?". At the bottom, there are two buttons: "Pay Now" (highlighted in orange) and "Back".

Рисунок 6.6 – Форма оплати замовлення

У результаті користувач має можливість переглянути всі свої придбані та активні послуги у своєму особистому кабінеті з метою перевірки та підтвердження своїх покупок.

6.3 Реалізація особистого кабінету користувача

Під час реєстрації дані користувача автоматично вносяться до особистого кабінету. У випадку потреби зміни цих даних або їх невідповідності дійсним, користувач може вносити зміни. Для цього потрібно увійти в особистий кабінет та натиснути кнопку "Редагувати", яка розташована праворуч від фотографії користувача. Після внесення необхідних змін користувач може зберегти їх, натиснувши відповідну

кнопку.

Вигляд сторінки особистого кабінету зображений на малюнку 6.7.

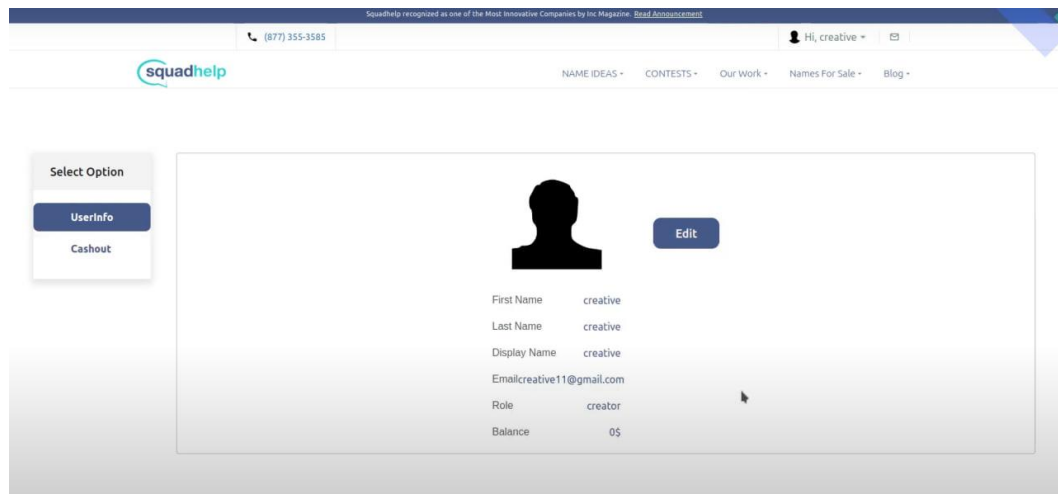


Рисунок 6.7 – Вікно особистого кабінету користувача

Також користувач може в будь-який момент переглянути всі свої придбані послуги у розділі особистого кабінету(рис 6.8) У ньому відображаються назва замовлення, його вартість, кількість залучених осіб та час, що минув з моменту придбання послуги.



Рисунок 6.8 – Вікно придбаних послуг у особистому кабінеті користувача

6.4 Висновки за розділом

В розділі було описано процес взаємодії користувача з інтернет-магазином, способи реєстрації та авторизації, заповнення форми замовлення та оплати послуги.

Інтерфейс вебдодатка розроблений з урахуванням зручності для всіх користувачів і має на меті уникнути будь-яких непередбачених складнощів під час його використання відповідно до основного призначення.

ВИСНОВКИ

У процесі виконання дипломної кваліфікаційної роботи було розроблено інтернет-магазин з продажу послуг розробки логотипу. Для цього було виконано ряд завдань:

- виконано аналіз потреб та проблемної області в сфері розробки брендингу та створення логотипів;
- проведено дослідження існуючих підходів та програмних рішень;
- обрано оптимальні технічні засоби для реалізації інтернет-магазину з продажу послуг розробки логотипу та обґрунтовано цей вибір;
- виконано проєктування архітектури інтернет-магазину з урахуванням функціональних та нефункціональних вимог;
- програмно реалізовано інтернет-магазин з використанням обраних технічних засобів та з урахуванням проєктованої архітектури.

Для розробки проєкту було обрано мову JavaScript для розробки серверної частини сайту, HTML та CSS для клієнтської частини.

Як базу даних було обрано базу даних MongoDB. Створено структуру бази даних інтернет-магазину.

Також проаналізоване середовище розробки, що відповідає зазначеним у постановці задачі критеріям, та необхідний набір інструментів для вирішення проєктної задачі.

Обґрунтовано вибір мови програмування та середовища функціонування системи.

В результаті випробувань розроблений інтернет-магазин дозволяє відображати інформацію про послуги на сайті замовляти та продавати їх коректно.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Пономаренко Л. А. Електронна комерція [Текст] : підруч. для студ. екон. спец. / Л. А. Пономаренко, В. О. Філатов. – К. : КНТЕУ, 2002. – 443 с.
2. JavaScript Variables [Electronic resource]. – Access mode : www.w3schools.com/js/js_variables.asp.
3. Краус К. М. Електронна комерція та інтернет-торгівля [Текст] : навч.-метод. Посіб. / К. М. Краус, Н. М. Краус, О. В. Манжура. – Київ : Аграр Медіа Груп, 2021. – 454 с.
4. PHP Tutorial [Electronic resource]. – Access mode : falconbyte.net/php-einrichten.php.
5. MySQL Documentation [Electronic resource]. – Access mode: www.mysql.com.
6. Закон України Про електронну комерцію [Електронний ресурс]. – Режим доступу: zakon.rada.gov.ua/laws/show/675-19#Text.
7. Електронна комерція у 2024 році [Електронний ресурс]. – Режим доступу: wezom.com.ua/ua/blog/10-golovnih-chelendzhiv-dlya-elektronnoyi-komertsiyi-u-2024-rotsi
8. PHP [Electronic resource]. – Access mode : www.php.de/.
9. Бардзелл, Джеффри Macromedia Dreamweaver MX 2004 с ASP, ColdFussion и PHP. Из первых рук (+CD – ROM) [Текст] : учебник для вузов / Джеффри Бардзелл. – М.:Эком, 2017. – 560 с.
10. Бенкен, Е. PHP, MySQL, XML. Программирование для Интернета [Текст] : учебник для вузов / Е. Бенкен. – М.: БХВ-Петербург, 2016. – 352 с.
11. Logo Design Service Market Size Research Report [2023-2030] [Electronic resource]. – Access mode: www.linkedin.com/pulse/logo-design-service-market-size-research-hvv1f

12. Дунаев.В. Сценарии для Web-сайта. PHP и JavaScript [Текст] : учебник для вузов / В. Дунаев. – М.: БХВ-Петербург, 2017. – 576 с.
13. Кузнецов, М. PHP. Практика создания Web-сайтов [Текст] : учебник для вузов / М. Кузнецов, И. Симдянов. – М.: БХВ-Петербург, 2012. – 577 с.
14. Ленгисторф, Джейсон PHP и jQuery для профессионалов [Текст] : учебник для вузов / Джейсон Ленгсторф. – М.: Вильямс, 2017. – 362 с.
15. Hedengren T. D. Smashing WordPress [Text] : beyond the blog / Thord Daniel Hedengren. – Chichester, West Sussex : Wiley, 2010. – 308 p.

ДОДАТОК А
Текст програми

A.1 Компонент OfferBox.jsx. Функція додавання товару до кошика.

```

import React from 'react';
import { connect } from 'react-redux';
import Rating from 'react-rating';
import { withRouter } from 'react-router-dom';
import isEqual from 'lodash/isEqual';
import classNames from 'classnames';
import { confirmAlert } from 'react-confirm-alert';
import { goToExpandedDialog } from '../../store/slices/chatSlice';
import {
  changeMark,
  clearChangeMarkError,
  changeShowImage,
} from '../../store/slices/contestByIdSlice';
import CONSTANTS from '../../constants';
import styles from './OfferBox.module.sass';
import 'react-confirm-alert/src/react-confirm-alert.css';
import './confirmStyle.css';

const OfferBox = (props) => {
  const findConversationInfo = () => {
    const { messagesPreview, id } = props;
    const participants = [id, props.data.User.id];
    participants.sort(
      (participant1, participant2) => participant1 - participant2
    );
    for (let i = 0; i < messagesPreview.length; i++) {
      if (isEqual(participants, messagesPreview[i].participants)) {
        return {
          participants: messagesPreview[i].participants,
          _id: messagesPreview[i]._id,
          blackList: messagesPreview[i].blackList,
          favoriteList: messagesPreview[i].favoriteList,
        };
      }
    }
    return null;
  };
};

```

```

const resolveOffer = () => {
  confirmAlert({
    title: 'confirm',
    message: 'Are u sure?',
    buttons: [
      {
        label: 'Yes',
        onClick: () =>
          props.setOfferStatus(props.data.User.id, props.data.id, 'resolve'),
      },
      {
        label: 'No',
      },
    ],
  });
};

```

```

const rejectOffer = () => {
  confirmAlert({
    title: 'confirm',
    message: 'Are u sure?',
    buttons: [
      {
        label: 'Yes',
        onClick: () =>
          props.setOfferStatus(props.data.User.id, props.data.id, 'reject'),
      },
      {
        label: 'No',
      },
    ],
  });
};

```

```

const changeMark = (value) => {
  props.clearError();
  props.changeMark({
    mark: value,
    offerId: props.data.id,
    isFirst: !props.data.mark,
    creatorId: props.data.User.id,
  });
};

```

```

    });
};

const offerStatus = () => {
  const { status } = props.data;
  if (status === CONSTANTS.OFFER_STATUS_REJECTED) {
    return (
      <i
        className={classNames('fas fa-times-circle reject', styles.reject)}
      />
    );
  }
  if (status === CONSTANTS.OFFER_STATUS_WON) {
    return (
      <i
        className={classNames('fas fa-check-circle resolve', styles.resolve)}
      />
    );
  }
  return null;
};

const goChat = () => {
  props.goToExpandedDialog({
    interlocutor: props.data.User,
    conversationData: findConversationInfo(),
  });
};

const { data, role, id, contestType } = props;
const { avatar, firstName, lastName, email, rating } = props.data.User;
return (
  <div className={styles.offerContainer}>
    {offerStatus()}
    <div className={styles.mainInfoContainer}>
      <div className={styles.userInfo}>
        <div className={styles.creativeInfoContainer}>
          <img
            src={
              avatar === 'anon.png'
                ? CONSTANTS.ANONYM_IMAGE_PATH
                : `${CONSTANTS.publicURL}${avatar}`
            }
          />
        </div>
      </div>
    </div>
  </div>
);

```

```

    }
    alt=»user»
  />
  <div className={styles.nameAndEmail}>
    <span>`${firstName} ${lastName}`</span>
    <span>{email}</span>
  </div>
</div>
<div className={styles.creativeRating}>
  <span className={styles.userScoreLabel}>Creative Rating </span>
  <Rating
    initialRating={rating}
    fractions={2}
    fullSymbol={
      <img
        src={` ${CONSTANTS.STATIC_IMAGES_PATH}star.png`}
        alt=»star»
      />
    }
    placeholderSymbol={
      <img
        src={` ${CONSTANTS.STATIC_IMAGES_PATH}star.png`}
        alt=»star»
      />
    }
    emptySymbol={
      <img
        src={` ${CONSTANTS.STATIC_IMAGES_PATH}star-outline.png`}
        alt=»star-outline»
      />
    }
    readonly
  />
</div>
</div>
<div className={styles.responseConainer}>
  {contestType === CONSTANTS.LOGO_CONTEST ? (
    <img
      onClick={() =>
        props.changeShowImage({
          imagePath: data.fileName,
          isShowOnFull: true,

```

```

    })
  }
  className={styles.responseLogo}
  src={` ${CONSTANTS.publicURL} ${data.fileName} `}
  alt=»logo»
/>
):(
  <span className={styles.response}>{data.text}</span>
)}
{data.User.id !== id && (
  <Rating
    fractions={2}
    fullSymbol={
      <img
        src={` ${CONSTANTS.STATIC_IMAGES_PATH}star.png`}
        alt=»star»
      />
    }
    placeholderSymbol={
      <img
        src={` ${CONSTANTS.STATIC_IMAGES_PATH}star.png`}
        alt=»star»
      />
    }
    emptySymbol={
      <img
        src={` ${CONSTANTS.STATIC_IMAGES_PATH}star-outline.png`}
        alt=»star»
      />
    }
    onClick={changeMark}
    placeholderRating={data.mark}
  />
)}
</div>
{role !== CONSTANTS.CREATOR && (
  <i onClick={goChat} className=»fas fa-comments» />
)}
</div>
{props.needButtons(data.status) && (
  <div className={styles.btnsContainer}>
    <div onClick={resolveOffer} className={styles.resolveBtn}>

```

```

        Resolve
      </div>
      <div onClick={rejectOffer} className={styles.rejectBtn}>
        Reject
      </div>
    </div>
  )}
</div>
);
};

```

A.2 Компонент `mongoose.js`. Функція підключення до бази даних.

```

Const mongoose = require('mongoose');
const path = require('path');
const env = process.env.NODE_ENV || 'development';
const configPath = path.join(__dirname, '..', 'config/mongoConfig.json');
const config = require(configPath)[ env ];

mongoose.connect(
  `mongodb://${ config.host }:${ config.port }/${ config.database }`,
  { useNewUrlParser: true, useUnifiedTopology: true }, (err) => {
    if (err) {
      console.log(err);
      process.exit(1);
    }
  });
mongoose.set('debug', env === 'development');

module.exports = mongoose;

```

A.3 Компонент `mongoose.js`. Функція авторизації користувача.

```

Import React from 'react';
import { connect } from 'react-redux';
import { Field, Form, Formik } from 'formik';
import { checkAuth, clearAuth } from '../store/slices/authSlice';
import styles from './LoginForm.module.sass';
import FormInput from '../FormInput/FormInput';
import Schemas from '../utils/validators/validationSchemas';
import Error from '../Error/Error';
import CONSTANTS from '../constants';

```

```

class LoginForm extends React.Component {
  componentWillMount() {
    this.props.authClear();
  }

  clicked = (values) => {
    this.props.loginRequest({ data: values, history: this.props.history });
  };

  render() {
    const { error, isFetching } = this.props.auth;
    const { submitting, authClear } = this.props;

    const formInputClasses = {
      container: styles.inputContainer,
      input: styles.input,
      warning: styles.fieldWarning,
      notValid: styles.notValid,
      valid: styles.valid,
    };

    return (
      <div className={styles.loginForm}>
        {error && (
          <Error
            data={error.data}
            status={error.status}
            clearError={authClear}
          />
        )}
        <h2>LOGIN TO YOUR ACCOUNT</h2>
        <Formik
          initialValues={{
            email: '',
            password: '',
          }}
          onSubmit={this.clicked}
          validationSchema={Schemas.LoginSchema}
        >
          <Form>
            <FormInput

```

```

        classes={formInputClasses}
        name=»email«
        type=»text«
        label=»Email Address«
      />
      <FormInput
        classes={formInputClasses}
        name=»password«
        type=»password«
        label=»Password«
      />
      <button
        type=»submit«
        disabled={submitting}
        className={styles.submitContainer}
      >
        <span className={styles.inscription}>
          {isFetching ? 'Submitting...' : 'LOGIN'}
        </span>
      </button>
    </Form>
  </Formik>
</div>
);
}
}

const mapStateToProps = (state) => {
  const { auth } = state;
  return { auth };
};

const mapDispatchToProps = (dispatch) => ({
  loginRequest: ({ data, history }) =>
    dispatch(checkAuth({ data, history, authMode: CONSTANTS.AUTH_MODE.LOGIN })),
  authClear: () => dispatch(clearAuth()),
});

export default connect(mapStateToProps, mapDispatchToProps)(LoginForm);

import React from 'react';
import { Link } from 'react-router-dom';

```

```

import { connect } from 'react-redux';
import LoginForm from '../../components/LoginForm/LoginForm';
import Logo from '../../components/Logo';
import styles from './LoginPage.module.sass';
import { clearAuthError } from '../../store/slices/authSlice';
import CONSTANTS from '../../constants';

const LoginPage = props => (
  <div className={styles.mainContainer}>
    <div className={styles.loginContainer}>
      <div className={styles.headerSignUpPage}>
        <Logo src={`/${CONSTANTS.STATIC_IMAGES_PATH}logo.png`} alt='logo' />
        <div className={styles.linkLoginContainer}>
          <Link to='/registration' style={{ textDecoration: 'none' }}>
            <span>Signup</span>
          </Link>
        </div>
      </div>
      <div className={styles.loginFormContainer}>
        <LoginForm history={props.history} />
      </div>
    </div>
  </div>
);

const mapDispatchToProps = dispatch => ({
  clearError: () => dispatch(clearAuthError()),
});

export default connect(null, mapDispatchToProps)(LoginPage);

```

A.4 Компонент AgreeTermOfServiceInput.jsx. Функція отримання згоди з правилами надання послуг.

```

// get Agree Term Of Service
const AgreeTermOfServiceInput = ({ id, type, classes, label, ...rest }) => (
  <Field {...rest}>
    {props => {
      const {
        meta: { touched, error },
        field,

```

```

    } = props;

    return (
      <div>
        <div className={classes.container}>
          <input {...field} placeholder={label} id={id} type={type} />
          <label htmlFor={id}>
            By clicking this checkbox, you agree to our{ ' ' }
            <a href='https://www.google.com' target='_blank' rel='noreferrer'>
              Terms of Service.
            </a>
          </label>
        </div>
        {touched && error && <span className={classes.warning}>{error}</span>}
      </div>
    );
  }
}
</Field>
);

```

A.5 Компонент RegistrationForm.jsx. Функція реєстрації користувача.

```

Class RegistrationForm extends React.Component {
  componentWillUnmount () {
    this.props.authClear();
  }

  clicked = values => {
    this.props.register({
      data: {
        firstName: values.firstName,
        lastName: values.lastName,
        displayName: values.displayName,
        email: values.email,
        password: values.password,
        role: values.role,
      },
      history: this.props.history,
    });
  }
}

```

```

};

render () {
  const { submitting, auth, authClear } = this.props;
  const { error } = auth;
  const formInputClasses = {
    container: styles.inputContainer,
    input: styles.input,
    warning: styles.fieldWarning,
    notValid: styles.notValid,
    valid: styles.valid,
  };
  return (
    <div className={styles.signUpFormContainer}>
      {error && (
        <Error
          data={error.data}
          status={error.status}
          clearError={authClear}
        />
      )}
      <div className={styles.headerFormContainer}>
        <h2>CREATE AN ACCOUNT</h2>
        <h4>We always keep your name and email address private.</h4>
      </div>
      <Formik
        initialValues={{
          firstName: '',
          lastName: '',
          displayName: '',
          email: '',
          password: '',
          confirmPassword: '',
          role: CONSTANTS.CUSTOMER,
          agreeOfTerms: false,
        }}
        onSubmit={this.clicked}
        validationSchema={Schemas.RegistrationSchem}
      >
        <Form>
          <div className={styles.row}>
            <FormInput

```

```

        name='firstName'
        classes={formInputClasses}
        type='text'
        label='First name'
    />
    <FormInput
        name='lastName'
        classes={formInputClasses}
        type='text'
        label='Last name'
    />
</div>
<div className={styles.row}>
    <FormInput
        name='displayName'
        classes={formInputClasses}
        type='text'
        label='Display Name' />
    <FormInput
        name='email'
        classes={formInputClasses}
        type='text'
        label='Email Address' />
</div>
<div className={styles.row}>
    <FormInput
        name='password'
        classes={formInputClasses}
        type='password'
        label='Password'>
    <FormInput
        name='confirmPassword'
        classes={formInputClasses}
        type='password'
        label='Password confirmation'>
</div>
<div className={styles.choseRoleContainer}>
    <Field
        name='role'
        type='radio'
        value={CONSTANTS.CUSTOMER}
        strRole='Join As a Buyer'

```

```

    infoRole='I am looking for a Name, Logo or Tagline for my business, brand or product.'
    Component={RoleInput}
    id={CONSTANTS.CUSTOMER}
  />
  <Field
    name='role'
    type='radio'
    value={CONSTANTS.CREATOR}
    strRole='Join As a Creative'
    infoRole='I plan to submit name ideas, Logo designs or sell names in Domain Marketplace.'
    Component={RoleInput}
    id={CONSTANTS.CREATOR}
  />
</div>
<div className={styles.termsOfService}>
  <AgreeTermOfServiceInput
    name='agreeOfTerms'
    classes={{
      container: styles.termsOfService,
      warning: styles.fieldWarning,
    }}
    id='termsOfService'
    type='checkbox'
  />
</div>
<button
  type='submit'
  disabled={submitting}
  className={styles.submitContainer}
>
  <span className={styles.inscription}>Create Account</span>
</button>
</Form>
</Formik>
</div>);}}
const mapStateToProps = state => ({
  auth: state.auth,
  initialValues: {role: CONSTANTS.CUSTOMER,},
});

```

A.6 Компонент PayForm.jsx. Функція оплати товару.

```

Import React from 'react';
import Cards from 'react-credit-cards';
import { Form, Formik } from 'formik';
import 'react-credit-cards/es/styles-compiled.css';
import { connect } from 'react-redux';
import styles from './PayForm.module.sass';
import { changeFocusOnCard } from '../store/slices/paymentSlice';
import PayInput from '../InputComponents/PayInput/PayInput';
import Schems from '../utils/validators/validationSchems';

const PayForm = (props) => {
  const changeFocusOnCard = (name) => {
    props.changeFocusOnCard(name);
  };

  const pay = (values) => {
    props.sendRequest(values);
  };

  const { focusOnElement, isPayForOrder } = props;
  return (
    <div className={styles.payFormContainer}>
      <span className={styles.headerInfo}>Payment Information</span>
      <Formik
        initialValues={{
          focusOnElement: '',
          name: '',
          number: '',
          cvc: '',
          expiry: '',
        }}
        onSubmit={pay}
        validationSchema={Schems.PaymentSchema}
      >
        {({ values }) => {
          const { name, number, expiry, cvc } = values;

          return (
            <>

```

```

<div className={styles.cardContainer}>
  <Cards
    number={number || ''}
    name={name || ''}
    expiry={expiry || ''}
    cvc={cvc || ''}
    focused={focusOnElement}
  />
</div>
<Form id={»myForm»} className={styles.formContainer}>
  <div className={styles.bigInput}>
    <span>Name</span>
    <PayInput
      name={»name»}
      classes={ {
        container: styles.inputContainer,
        input: styles.input,
        notValid: styles.notValid,
        error: styles.error,
      } }
      type={»text»}
      label={»name»}
      changeFocus={changeFocusOnCard}
    />
  </div>
  {!isPayForOrder && (
    <div className={styles.bigInput}>
      <span>Sum</span>
      <PayInput
        name={»sum»}
        classes={ {
          container: styles.inputContainer,
          input: styles.input,
          notValid: styles.notValid,
          error: styles.error,
        } }
        type={»text»}
        label={»sum»}
      />
    </div>
  )}
  <div className={styles.bigInput}>

```

```

<span>Card Number</span>
<PayInput
  isInputMask
  mask=»9999 9999 9999 9999 999»
  name=»number»
  classes={ {
    container: styles.inputContainer,
    input: styles.input,
    notValid: styles.notValid,
    error: styles.error,
  } }
  type=»text»
  label=»card number»
  changeFocus={ changeFocusOnCard }
/>
</div>
<div className={styles.smallInputContainer}>
  <div className={styles.smallInput}>
    <span>* Expires</span>
    <PayInput
      isInputMask
      mask=»99/99»
      name=»expiry»
      classes={ {
        container: styles.inputContainer,
        input: styles.input,
        notValid: styles.notValid,
        error: styles.error,
      } }
      type=»text»
      label=»expiry»
      changeFocus={ changeFocusOnCard }
    />
  </div>
  <div className={styles.smallInput}>
    <span>* Security Code</span>
    <PayInput
      isInputMask
      mask=»9999»
      name=»cvc»
      classes={ {
        container: styles.inputContainer,

```

```

        input: styles.input,
        notValid: styles.notValid,
        error: styles.error,
      }}
      type=»text«
      label=»cvc«
      changeFocus={changeFocusOnCard}
    />
  </div>
</div>
</Form>
</>
);
}}
</Formik>
{isPayForOrder && (
  <div className={styles.totalSum}>
    <span>Total: $100.00</span>
  </div>
)}
<div className={styles.buttonsContainer}>
  <button form=»myForm« className={styles.payButton} type=»submit«>
    <span>{isPayForOrder ? 'Pay Now' : 'CashOut'}</span>
  </button>
  {isPayForOrder && (
    <div onClick={() => props.back()} className={styles.backButton}>
      <span>Back</span>
    </div>
  )}
</div>
</div>
);
};

const mapDispatchToProps = (dispatch) => ({
  changeFocusOnCard: (data) => dispatch(changeFocusOnCard(data)),
});

export default connect(null, mapDispatchToProps)(PayForm);

```

ДОДАТОК Б
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ З
ПРОДАЖУ ПОСЛУГ РОЗРОБКИ ЛОГОТИПУ**

Виконав: Марія БАТАНОВА
ст. групи КНТ-111сп

Керівник: Олена ШИТІКОВА
к.т.н., доцент каф.ПЗ

Рисунок Б.1 – Слайд 1

Об'єкт, предмет та мета дослідження

- **Об'єкт дослідження** – процеси електронної комерції.
- **Предмет дослідження** – інтернет-магазин з продажу послуг розробки логотипу.
- **Мета роботи** – розробка програмного продукту, який реалізує функції інтернет-магазину з індивідуальним наповненням.

Рисунок Б.2 – Слайд 2

Для досягнення поставленої мети необхідно виконати наступні завдання:

- провести аналіз сучасних тенденцій і технологій у сфері електронної комерції;
- вивчити вимоги користувачів до функціоналу та дизайну інтернет-магазинів;
- розробити технічне завдання для створення онлайн-магазину з продажу послуг розробки логотипів;
- обрати та обґрунтувати вибір технологій та інструментів для розробки вебдодатку;
- провести проектування архітектури інтернет-магазину з урахуванням функціональних та нефункціональних вимог;
- реалізувати структуру бази даних для зберігання інформації про послуги та замовлення;
- розробити інтерфейс користувача, який забезпечить зручну навігацію та замовлення послуг;
- підготувати документацію та інструкції для користувачів і адміністраторів системи.

Рисунок Б.3 – Слайд 3

Огляд існуючих аналогів онлайн-платформ з продажу послуг розробки логотипу

Критерій	Freelance.ua	Kabanchik.ua	Weblancer.net	Designer.ua
Комісія за послуги	10-20%	5-15%	7-10%	5-10%
Підтримка користувачів	Обмежена	Цілодобова	Робочі години	Обмежена
Кількість проектів	Середнє	Високе	Середнє	Низьке
Перевірка виконавців	Є, але обмежена	Ретельна	Є, але обмежена	Є, але обмежена
Інтерфейс і зручність	Застарілий	Зручний	Сучасний, але складний	Простий, але незручний
Безпека угод	Гарантується, але з обмеженнями	Повна гарантія	Обмежена гарантія	Гарантується, але з обмеженнями

Рисунок Б.4 – Слайд 4

Постановка завдання

Програмна реалізація інтернет-магазину з продажу послуг розробки логотипу є актуальним завданням у зв'язку зі зростаючою потребою бізнесів у професійному брендингу. Логотип відіграє ключову роль у формуванні бренду та допомагає виділитися на ринку.

Рисунок Б.5 – Слайд 5

Порівняння мов програмування

№ п/п	Можливості	JavaScript	PHP	Java
1	Використання фреймворків	JavaScript має доступ до таких фреймворків, як React, Angular, Vue	Користувачі PHP мають доступ до таких фреймворків як Laravel, Zend, Yii	Користувачі Java мають доступ до фреймворків, як Spring, Spring boot, Hibernate
2	Популярність використання	Вибір мови для веброзробки, особливо для фронтенду, але також і для бекенду з Node.js	Вибір мови для веброзробки	Широко використовується в додатках безпеки, додатках Android, вебдодатках AI
3	Підключення до бази даних	Можливий доступ до різних баз даних за допомогою бібліотек, таких як Sequelize, Mongoose	Можливий доступ до більш ніж 20 різних баз даних.	Можливий доступ майже до всіх баз даних.
4	Крива здатності освоєння мови	JavaScript відрізняється дуже низькою кривою навчання, легко почати.	PHP відрізняється дуже низькою кривою навчання, легко почати. Бивчати	Більше підходить для довгострокових проєктів
5	Майбутнє мови програмування	Довготривале, оскільки JavaScript широко використовується у веброзробці	Довготривале, понад 70% вебсайтів створено за допомогою PHP	Безперечно довготривале завдяки своїй безпеці та швидкості
6	Читабельність	Мова JavaScript не дуже типізована і дотримується класичного підходу	Мова PHP не дуже типізована і дотримується класичного підходу	Нелегко читати.

Рисунок Б.6 – Слайд 6

Порівняння БД

Критерій	MongoDB	MySQL	PostgreSQL
Гнучкість схеми	Гнучка схема	Статична схема	Гнучка схема
Горизонтальне масштабування	Підтримується	Обмежена підтримка	Підтримується
Підтримка JSON	Так	Ні	Ні
Мова запитів	JSON-подібні	SQL	SQL
Швидкість операцій	Висока	Залежить від операції	Залежить від операції
Підтримка транзакцій	Так	Так	Так
Строки або колонки	Колекції	Таблиці	Таблиці
Проміжний код	JavaScript	SQL	SQL
Підтримка ACID	Часткова	Повна	Повна
Реплікація та шардування	Підтримується	Обмежена підтримка	Підтримується

Рисунок Б.7 – Слайд 7

Функціональна схема роботи інтернет-магазину



Рисунок Б.8 – Слайд 8

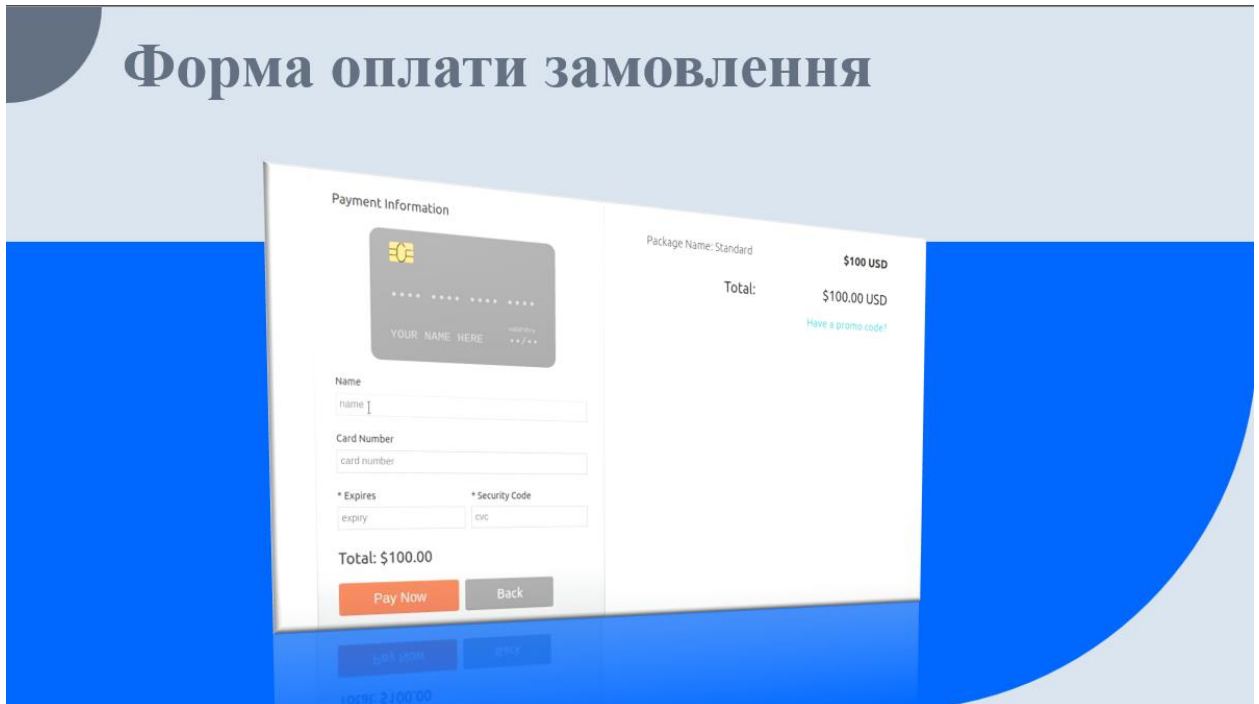


Рисунок Б.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію інтернет-магазину з продажу послуг розробки логотипу. Для цього було виконано ряд завдань:

- виконано аналіз потреб та проблемної області в сфері розробки брендінгу та створення логотипів;
- проведено дослідження існуючих підходів та програмних рішень;
- обрано оптимальні технічні засоби для реалізації інтернет-магазину з продажу послуг розробки логотипу та обґрунтовано цей вибір;
- виконано проектування архітектури інтернет-магазину з урахуванням функціональних та нефункціональних вимог;
- програмно реалізовано інтернет-магазин з використанням обраних технічних засобів та з урахуванням проекрованої архітектури;
- сформовано висновки та рекомендації на основі виконаної роботи з розробки інтернет-магазину з продажу послуг розробки логотипу.

В результаті виконання завдання було успішно створено систему, яка повністю відповідає вимогам, визначеним на етапі початку розробки і закріпленим у технічному завданні. Розроблена система відрізняється високою зручністю використання та надійно виконує всі необхідні функції, що визначено для задоволення потреб користувачів.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

Рисунок Б.12 – Слайд 12

Дякую за увагу

Рисунок Б.13 – Слайд 13