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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення для оптимізації роботи інтернет-ресторану

Виконав(ла): студент(ка) 4 курсу, групи СПс-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

	(підпис)	Довбань В. І. (прізвище та ініціали)
Керівник	(підпис)	Тимків П. О. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю. М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М. Р. (прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

Тернопіль 2026

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

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра Інженерії програмного забезпечення
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
студенту Довбань Володимир Іванович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для оптимізації роботи інтернет-ресторану.

Керівник роботи к.т.н., ст. викл. Тимків Павло Олександрович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від « 06 » *квітня* 2026 року № *4/9-172*
2. Термін подання студентом завершеної роботи
3. Вихідні дані до роботи Технічне завдання на розробку, набір вхідних даних для тестування, вимоги до управління каталогом страв, вимоги для оформлення та оплати замовлення.

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз предметної області, аналіз конкурентів, визначення вимог, визначення інструментів, методологій та технологій, моделювання системи та створення діаграм, розробка веб застосунку, тестування, підготовка до перенесення на хостинг

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
UML діаграми варіантів використання для акторів, UML діаграми класів, UML діаграма послідовності, рисунки дизайну сайту, рисунки програмного коду, слайди презентації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Мариненко С. Ю. к.т.н., доц. каф. МТ		
Нормоконтроль	Стоянов Ю. М. к.т.н. доц. каф. ПІ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір та затвердження теми роботи	06.04.2026	
2	Опис предметної області	21.04.2026	
3	Аналіз конкурентів	21.04.2026	
4	Проєктування програмного забезпечення	08.05.2026	
5	Опис та реалізація сторінок системи	18.05.2026	
6	Тестування системи	27.05.2026	
7	Безпека життєдіяльності, основи охорони праці	02.06.2026	
8	Перевірка на плагіат	10.06.2026	
9	Попередній захист кваліфікаційної роботи	10.06.2026	
10	Нормоконтроль	15.06.2026	
11	Підготовка презентації та доповіді	15.06.2026	
12	Захист кваліфікаційної роботи		

Студент _____ ___ Довбань В. І.
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2026, Сторінок 71, таблиць 2, рисунків 24, лістингів 2, джерел 29, додатків 4, презентація.

Тема: Розробка програмного забезпечення для оптимізації роботи інтернет-ресторану.

Мета дипломної роботи полягає в аналізі бізнес-процесів ресторану та розробці сучасного, швидкого і зручного вебсайту для автоматизації прийому замовлень. На сьогодні швидкість завантаження сайту та зручність користувацького інтерфейсу є критичними факторами у сфері доставки їжі, оскільки вони безпосередньо впливають на конверсію та рівень задоволеності клієнтів.

При розробці програмного забезпечення було приділено значну увагу оптимізації швидкодії вебзастосунку та створенню інтуїтивно зрозумілого каталогу страв. Програмну реалізацію виконано на базі системи управління контентом WordPress із використанням модуля електронної комерції WooCommerce. Для автоматизації рутинних задач розробки, мініфікації файлів стилів та скриптів, а також оптимізації зображень було інтегровано інструментарій Gulp, що дозволило суттєво підвищити продуктивність роботи сайту.

Ця дипломна робота робить практичний внесок у вирішення проблеми повільних та неоптимізованих сервісів доставки. Розроблена система пропонує бізнесу готове та швидке рішення для електронної комерції, яке забезпечує високі показники продуктивності та покращує користувацький досвід.

Ключові слова: інтернет-ресторан, WordPress, WooCommerce, Gulp, онлайн замовлення, доставка.

ABSTRACT

Bachelor’s Thesis. Ivan Pul’uj Ternopil National Technical University, Department of Software Engineering, Major 121 “Software Engineering.” TNTU, 2026, The work consists of 71 pages, 2 tables, 24 figures, 2 code listings, 29 references, 4 appendices, presentation.

Topic: Software Development to Optimize the Operations of an Online Restaurant.

The aim of this thesis is to analyze the restaurant’s business processes and develop a modern, fast, and user-friendly website for automating order processing. Today, website loading speed and user interface convenience are critical factors in the food delivery industry, as they directly impact conversion rates and customer satisfaction.

During the development of the software, significant attention was paid to optimizing the web application's performance and creating an intuitive menu. The software implementation was based on the WordPress content management system using the WooCommerce e-commerce module. To automate routine development tasks, minify style sheets and scripts, and optimize images, the Gulp toolkit was integrated, which significantly improved the website’s performance.

This thesis makes a practical contribution to solving the problem of slow and unoptimized delivery services. The developed system offers businesses a ready-made and fast e-commerce solution that delivers high performance and improves the user experience.

Keywords: online restaurant, WordPress, WooCommerce, Gulp, online ordering, delivery.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – База даних

ІС – Інформаційна система

ПЗ – Програмне забезпечення

СУБД – Система управління базами даних

ТЗ – Технічне завдання

API – Application Programming Interface

CMS – Content Management System

CSS – Cascading Style Sheets

SASS – Syntactically Awesome Style Sheets

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

JS – JavaScript

JSON – JavaScript Object Notation

PHP – Hypertext Preprocessor

SEO – Search Engine Optimization

UI – User Interface

UX – User Experience

WP – WordPress

ВСТУП

Сучасна індустрія громадського харчування переживає фундаментальну трансформацію, де цифровий простір перестав бути просто доповненням до фізичного залу, а став основним каналом продажів. Сьогодні клієнт очікує від ресторану не лише якісної їжі, а й бездоганного сервісу в один клік: інтуїтивного меню, швидкого оформлення замовлення та прозорої системи доставки. В умовах глобальної цифровізації наявність власної, незалежної інформаційної системи є для закладу питанням не просто іміджу, а життєздатності та масштабування бізнесу.

Попри велику кількість готових рішень, існуючі на ринку варіанти часто мають суттєві недоліки. З одного боку знаходяться агрегатори доставки, які стягують високі комісійні відсотки та позбавляють ресторан прямого контакту з власною аудиторією. З іншого існують хмарні конструктори, які пропонують швидкий запуск, але суттєво обмежують власника у гнучкості налаштувань бізнес-логіки та прив'язують до постійної абонентської плати. Крім того, сайти, створені на базі універсальних шаблонів, часто страждають від надмірної перевантаженості кодом, що робить їх повільними та погіршує користувацький досвід.

Відповіддю на ці виклики є дана кваліфікаційна робота, яка несе в собі комплексне дослідження та практичну реалізацію повного циклу розробки програмного продукту. Вона охоплює всі етапи створення інформаційної системи: від детального аналізу бізнес-процесів інтернет-ресторану та проектування логічної архітектури бази даних до розробки індивідуального користувацького інтерфейсу і фінального розгортання проєкту. Це не просто розробка чергового вебсайту, а створення повноцінної екосистеми, яка об’єднує клієнта, кухню та адміністрацію в єдиному цифровому середовищі.

Окрім безпосередньої взаємодії з клієнтом, розроблена система покликана суттєво оптимізувати внутрішні операційні процеси закладу. Перехід від традиційного ручного прийому замовлень по телефону до повністю автоматизованої системи мінімізує людський фактор та кількість критичних помилок. Зручна панель адміністратора дозволяє персоналу в режимі реального

часу відслідковувати нові надходження, змінювати статуси приготування страв та миттєво оновлювати асортимент. Додатково такий підхід створює надійну базу для збору аналітики: власник отримує точні статистичні дані про найпопулярніші позиції меню, пікові години навантаження та загальну ефективність продажів, що є необхідним фундаментом для подальшого стратегічного планування та маркетингу.

Для реалізації поставлених завдань було обрано сучасний та збалансований стек вебтехнологій. Фундаментом серверної частини системи виступає платформа управління контентом WordPress, написана мовою PHP. Її використання на відміну від закритих конструкторів дає повний контроль над даними клієнтів та серверною архітектурою. Логіка електронної комерції побудована на базі потужного розширення WooCommerce, яке забезпечує надійну роботу каталогу страв, кошика, розрахунку вартості доставки та управління статусами замовлень.

Окрему увагу в роботі приділено клієнтській частині та автоматизації процесів розробки. Для забезпечення динамічної взаємодії користувача з інтерфейсом (наприклад, AJAX додавання страв у кошик) активно застосовується JavaScript. А для того, щоб фінальний продукт працював максимально швидко та стабільно, у проєкт інтегровано інструментарій Gulp. Завдяки йому налаштовано автоматизовану збірку проєкту: Gulp бере на себе компіляцію препроцесорів, мініфікацію JavaScript-файлів та оптимізацію графічних ресурсів. Це дозволяє писати чистий, структурований код у процесі розробки, отримуючи на виході максимально легкий та швидкий вебзастосунок.

Таким чином, результатом даної роботи є готова до впровадження, швидка та незалежна інформаційна система. Вона вирішує ключові проблеми сучасного ресторанного бізнесу, надаючи закладу гнучкий інструмент для автоматизації онлайн-продажів та забезпечення високого рівня сервісу для своїх клієнтів.

РОЗДІЛ 1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ УПРАВЛІННЯ ІНТЕРНЕТ-РЕСТОРАНОМ

У першому розділі кваліфікаційної роботи проводиться аналіз предметної області у сфері інтернет ресторанів. Метою етапу є дослідження ринку та оцінка існуючих конкурентних рішень. Зібрані дані дозволяють сформувати концепцію застосунку визначити цільову аудиторію та скласти детальний перелік вимог до системи.

1.1 Аналіз конкурентів

Розробка власного програмного забезпечення для інтернет-ресторану потребує детального вивчення існуючих рішень на ринку України та локально у місті Тернопіль. Сьогодні заклади громадського харчування мають кілька альтернативних шляхів для організації онлайн-продажів. Для визначення переваг розроблюваної системи було проаналізовано три основні типи конкурентних рішень: глобальний агрегатор доставки, спеціалізовану хмарну платформу для ресторанів та успішний локальний інтернет-ресторан.

1.1.1 Платформа Glovo

Glovo є одним із найпопулярніших міжнародних агрегаторів доставки їжі в Україні, який активно працює в Тернополі [1]. Цей сервіс надає ресторанам можливість швидко вийти на ринок онлайн-замовлень без необхідності розробляти власний вебзастосунок. Достатньо лише укласти договір та передати своє меню для розміщення у загальному каталозі додатку.

До переваг цієї платформи належить величезна база активних користувачів, наявність власних кур'єрів та відсутність витрат на розробку технічної складової.

Проте система має критичні недоліки для бізнесу. Найбільшим з них є дуже висока комісія за кожне замовлення (до 30 відсотків від чека). Також заклад не має доступу до бази даних своїх клієнтів, позбавлений можливості повноцінного

брендування інтерфейсу та змушений конкурувати з десятками інших ресторанів безпосередньо на одному екрані пристрою користувача. На рисунку 1.1 зображено логотип застосунку Glovo.

Рисунок 1.1 – Логотип застосунку Glovo [1]

1.1.2 Хмарна система ChoiceQR

ChoiceQR є популярним українським IT-продуктом, який дозволяє закладам харчування створювати електронні меню та сайти для доставки за моделлю підписки (SaaS) [2]. Це готове програмне рішення, яке потребує лише базового налаштування візуальної теми та наповнення контентом.

Перевагами системи є швидкий запуск, сучасний інтерфейс користувача та наявність інтеграцій з популярними системами обліку.

Головний недолік полягає у закритій архітектурі. Ресторан змушений сплачувати регулярну абонентську плату, а власник не володіє вихідним кодом платформи. Це суттєво обмежує можливості для глибокої технічної оптимізації. Розробники закладу не можуть застосовувати специфічні інструменти автоматизації або модифікувати архітектуру бази даних під унікальні потреби свого бізнесу. На рисунку 1.2 зображено логотип платформи ChoiceQR.

Рисунок 1.2 – Логотип птатформи ChoiceQR [2]

1.1.3 Ресторан мережі «Файне Місто» (Тернопіль)

Цей конкурент є прикладом успішного локального рішення. Мережа ресторанів «Файне Місто» має власний повноцінний сайт для замовлення їжі та обслуговування жителів Тернополя [3]. Цей ресурс представляє індивідуальну розробку, яка функціонально найбільш наближена до теми даної дипломної роботи.

Переваги такого підходу очевидні: повний контроль над брендом, формування власної бази клієнтів, відсутність комісій агрегаторам та висока лояльність локальної аудиторії.

До недоліків існуючих індивідуальних рішень часто можна віднести складність управління контентом для адміністраторів (якщо бекенд не побудований на зручній системі управління контентом) та проблеми зі швидкодією. Без правильної оптимізації фронтенду сайти з великим каталогом страв та зображень можуть завантажуватися повільно, що негативно впливає на показники конверсії. На рисунку 1.3 зображено головну сторінку сайту Файного міста.

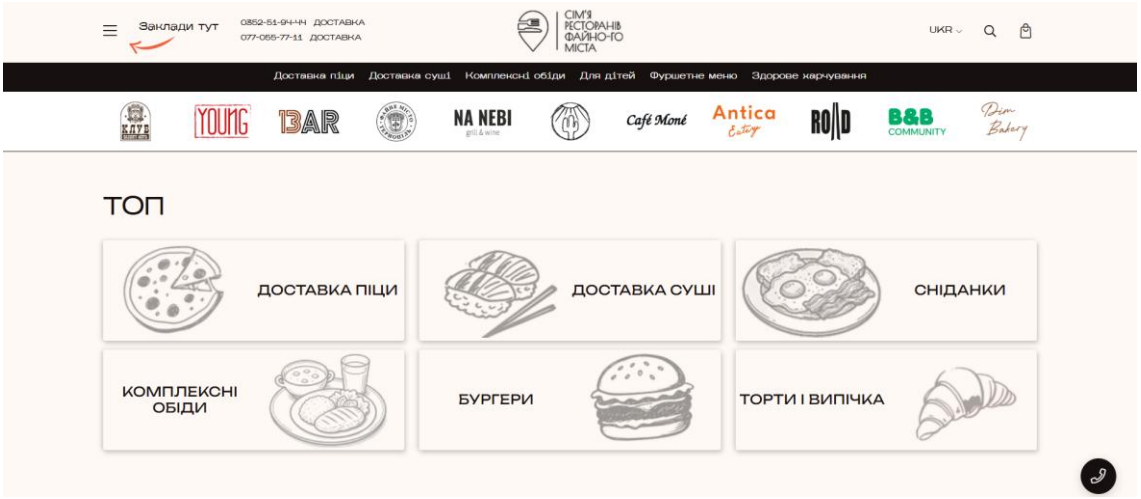


Рисунок 1.3 – Головна сторінка сайту Файне місто [3]

Проведений аналіз існуючих рішень показав, що використання агрегаторів забирає надто велику частку прибутку компанії, а готові хмарні конструктори обмежують технічну свободу закладу. Розробка власного незалежного інтернет-ресторану є найбільш вигідним вектором розвитку, проте потребує грамотного

вибору технологій. Створення інформаційної системи на базі гнучкої платформи WordPress із розширенням WooCommerce та оптимізацією фронтенд-ресурсів через Gulp є найбільш збалансованим рішенням. Такий підхід дозволить бізнесу уникнути комісій, отримати зручну панель адміністрування та забезпечити високу швидкість роботи застосунку, перевершуючи існуючі локальні аналоги.

1.2 Визначення вимог та специфікації ПЗ

Визначення вимог є критично важливим етапом створення програмного забезпечення [4]. Детальний аналіз та специфікація дозволяють узгодити очікування замовника з технічними можливостями розробників. Практика показує що саме на етапі планування закладається фундамент успішного продукту. Недостатнє розуміння потреб бізнесу часто призводить до перевитрат бюджету та зриву термінів здачі проекту. Завдяки детальному формуванню вимог створюється чітке бачення готового ресурсу та мінімізуються ризики виникнення архітектурних помилок під час подальшого написання коду.

Цільова аудиторія нашого вебзастосунку є доволі широкою та включає кілька ключових груп користувачів. Основний сегмент складають цифрові клієнти які замовляють страви з доставкою додому або обирають опцію самовивозу. Їм потрібен інтуїтивно зрозумілий інтерфейс для швидкого пошуку улюблених страв та зручного оформлення покупки. Окремою важливою групою є персонал закладу. Менеджерам та адміністраторам потрібен зручний інструмент для керування системою та миттєвої обробки нових заявок. Також розроблена система слугуватиме зручним цифровим меню для відвідувачів фізичного ресторану які зможуть ознайомитися з асортиментом без очікування вільного офіціанта.

Функціональні вимоги безпосередньо регламентують поведінку системи та визначають які саме завдання вона повинна виконувати у відповідь на дії користувача [5]. Вони формують основний перелік можливостей що надаються клієнту в процесі взаємодії з платформою [4]. Для нашого проекту ці вимоги формують основу електронної комерції та діяльності закладу.

Перелік базового функціоналу розробленої системи включає:

– Надання доступу до інтерактивного каталогу страв з актуальними цінами детальними описами та якісними фотографіями кожної страви;

– Створення повноцінного кошика для формування замовлень на винос або адресну доставку з автоматичним розрахунком загальної вартості;

– Впровадження надійної системи реєстрації та авторизації клієнтів для створення персональних облікових записів на платформі;

– Забезпечення роботи особистого кабінету де зберігатиметься історія попередніх покупок поточний статус активного замовлення та список улюблених страв користувача;

– Розробка захищеної адміністративної панелі для управління асортиментом формування ціноутворення та відслідковування нових заявок менеджерами закладу в режимі реального часу;

– Підтримка швидкого доступу до електронного каталогу безпосередньо в залі закладу за допомогою зручного сканування графічних QR кодів розміщених на столиках.

Нефункціональні вимоги регламентують внутрішні і зовнішні умови або загальні атрибути якості функціонування розроблюваної системи [5]. Оскільки платформа буде найактивніше використовуватися з мобільних пристроїв, питання технічної оптимізації та безперебійної роботи виходить на перший план. Сучасний користувач не буде чекати довгого завантаження сторінки тому технічні показники впливають на кількість успішних продажів.

Ключові параметри якості роботи вебзастосунку:

– Висока продуктивність: архітектура сайту код та медіафайли мають бути максимально оптимізовані для миттєвого завантаження сторінок навіть при слабкому сигналі мережі;

– Адаптивний дизайн та кросплатформність: користувацький інтерфейс повинен коректно і красиво відображатися на смартфонах планшетах та стаціонарних комп'ютерах з різною роздільною здатністю екрану;

– Ергономічність та зручність: процес пошуку вибору страви та оформлення

покупки має бути максимально логічним і вимагати мінімальної кількості кліків від гостя;

– Масштабованість та стабільність: серверна частина системи повинна легко витримувати великі навантаження та наплив трафіку у вечірні години вихідні чи святкові дні без падіння швидкості відгуку;

– Захист інформації та безпека: персональні дані покупців паролі та історія їхніх транзакцій мають зберігатися у зашифрованому вигляді з суворим дотриманням сучасних стандартів конфіденційності.

Зібрані специфікації та вимоги утворюють надійний фундамент для переходу до наступного етапу життєвого циклу розробки. Вони дають змогу сформувати правильне бачення кінцевого продукту розпочати детальне архітектурне проектування бази даних та приступити безпосередньо до написання програмного коду на базі обраного технологічного стеку.

1.3 Методологія та технології розробки

Створення сучасної інформаційної системи вимагає не лише глибокого розуміння бізнес процесів ресторану але й правильного підходу до самої розробки. Фундаментом якісного цифрового продукту є оптимально підібраний технологічний стек. Вдала комбінація інструментів для написання коду автоматизації збірки та локального тестування здатна кардинально змінити швидкість реалізації проекту. Завдяки правильним налаштуванням робочого середовища можна зекономити десятки годин на рутинних завданнях спрямовуючи всі зусилля на створення чистої архітектури.

1.3.1 Методологія розробки та організація робочого середовища

Процес створення інформаційної системи інтернет-ресторану базується на ітеративній методології розробки. Цей підхід передбачає поступове нарощування функціоналу системи починаючи від базового налаштування каталогу і закінчуючи інтеграцією складних механізмів електронної комерції та оптимізацією роботи.

Для забезпечення безпечного та ізольованого процесу створення програмного продукту використовується локальне середовище розробки. З цією метою обрано спеціалізоване програмне забезпечення Local. Даний інструмент дозволяє автоматизувати процес розгортання локального сервера, конфігурацію бази даних MySQL та налаштування вебсервера для коректної роботи PHP скриптів [6]. Використання програми Local суттєво пришвидшує початковий етап підготовки до роботи та дозволяє розробнику тестувати функціонал без необхідності використання віддаленого хостингу.

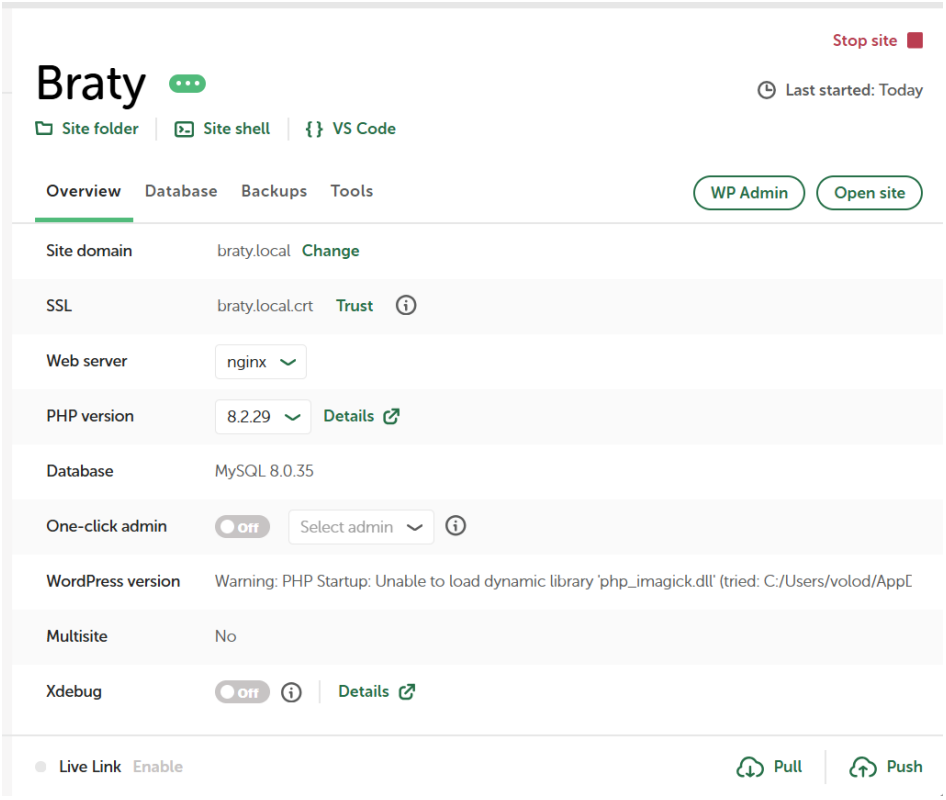


Рисунок 1.4 – Вікно налаштувань локального сервера у програмі Local

Безпосереднє написання програмного коду, редагування конфігураційних файлів та робота з терміналом здійснюється у середовищі Visual Studio Code. Цей редактор забезпечує зручну підсвітку синтаксису, швидку навігацію по файлах проекту та легкий запуск консольних команд через вбудований термінал.

1.3.2 Технології серверної частини та електронної комерції

Серверна архітектура розроблюваної системи базується на платформі WordPress. Ця система управління контентом написана мовою програмування PHP та є визнаним світовим лідером серед рішень для створення вебзастосунків [7]. Платформа WordPress забезпечує надійну основу для збереження інформації, надає готові механізми маршрутизації та безпеки, а також дозволяє гнучко розширювати базовий функціонал за допомогою власного програмного інтерфейсу (API). Для адміністраторів ресторану ця система пропонує інтуїтивно зрозумілу панель управління вмістом, яка не вимагає глибоких технічних знань від персоналу закладу.

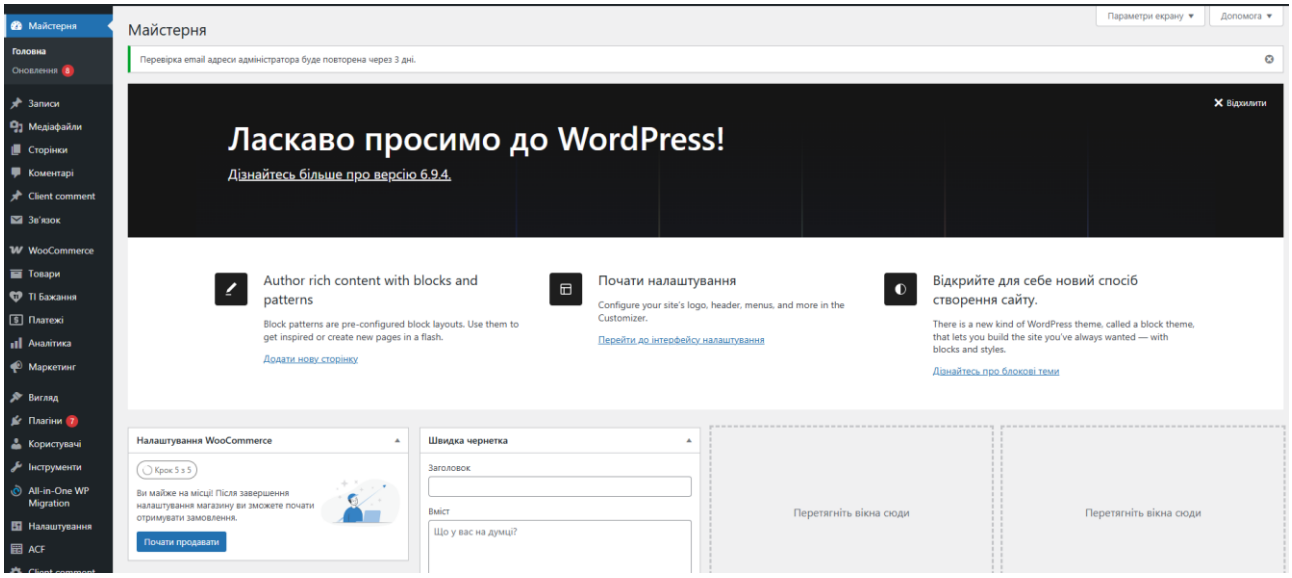


Рисунок 1.5 – Панель управління системою WordPress

Перетворення базового сайту на повноцінний інтернет-ресторан здійснюється за допомогою розширення WooCommerce. Це спеціалізоване програмне рішення для електронної комерції, яке глибоко інтегрується в архітектуру бази даних WordPress [8]. Розширення WooCommerce бере на себе всю бізнес-логіку роботи закладу. Воно автоматизує процеси додавання страв до кошика, розрахунку фінальної вартості з урахуванням доставки, оформлення замовлення клієнтом та зміни статусів заявок під час їх обробки менеджерами.

1.3.3 Технології клієнтської частини та стилізації

Графічний інтерфейс користувача розробляється з урахуванням концепції орієнтованості на мобільні пристрої (mobile-first). Для побудови адаптивного каркасу сторінок застосовується популярний фреймворк Bootstrap [9]. Його використання дозволяє створити гнучку систему сіток, яка автоматично підлаштовує відображення каталогу страв та кошика під розмір екрану пристрою клієнта, будь то смартфон, планшет чи настільний комп'ютер.

Рисунок 1.6 – Логотип фреймворку Bootstrap [9]

Для написання каскадних таблиць стилів застосовується препроцесор SASS [10]. Використання класичного CSS у масштабних проектах часто призводить до дублювання коду та складнощів у його підтримці. Натомість SASS надає розробнику можливість використовувати змінні для збереження корпоративних кольорів ресторану, застосовувати вкладені правила стилізації та створювати модульні блоки для повторного використання. Це робить візуальну частину проекту структурованою та легкою для подальшої модифікації.

Рисунок 1.7 – Логотип препроцесора Sass [10]

1.3.4 Автоматизація збірки та оптимізація продуктивності

Висока швидкість завантаження сторінок є однією з найважливіших нефункціональних вимог для інтернет-ресторану. Для досягнення максимальних показників продуктивності у проєкт впроваджено інструментарій Gulp [11]. Це система автоматизації збірки, яка працює на базі платформи Node.js і виконує ряд критично важливих завдань у фоновому режимі розробки.

У рамках даного проекту інструмент Gulp налаштовано на виконання таких процесів:

– Автоматична компіляція файлів препроцесора SASS у зрозумілий для браузера формат CSS;

– Об'єднання та мініфікація таблиць стилів і JavaScript файлів для зменшення їхнього загального розміру;

– Оптимізація та стиснення графічних матеріалів (фотографій страв) без помітної втрати візуальної якості;

– Автоматичне оновлення сторінки у браузері при збереженні змін у вихідному коді, що значно пришвидшує темп розробки.

Рисунок 1.8 – Процес виконання збірки інструментом Gulp

Використання автоматизованої збірки гарантує, що кінцевий користувач отримуватиме максимально оптимізовані ресурси. Це знижує навантаження на сервер ресторану та пришвидшує відображення електронного меню на мобільних пристроях клієнтів навіть при низькій швидкості інтернет-з'єднання.

У першому розділі було проаналізовано предметну область та конкурентні рішення на ринку що підтвердило необхідність розробки власної незалежної

інформаційної системи для інтернет ресторану. Завдяки детальному дослідженню сформовано чіткі функціональні вимоги, а також нефункціональні вимоги орієнтовані на високу продуктивність і безпеку даних. Для ефективної реалізації проєкту обрано збалансований стек технологій де серверна частина будується на базі платформи WordPress із розширенням WooCommerce клієнтська частина використовує адаптивну сітку Bootstrap та препроцесор SASS, а за оптимізацію ресурсів відповідає система автоматизованої збірки Gulp. Визначені специфікації та підібрані інструменти закладають міцний фундамент для переходу до етапу архітектурного проектування та безпосереднього написання програмного коду.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ ІНТЕРНЕТ-РЕСТОРАНОМ

Процес проектування програмного забезпечення є критично важливою стадією життєвого циклу розробки будь якої складної інформаційної системи. Після етапу ретельного збору вимог та аналізу потреб бізнесу настає момент переходу до архітектурного моделювання. Саме на цьому етапі розробник закладає фундамент майбутньої платформи визначаючи логіку взаємодії компонентів структуру бази даних та користувацькі сценарії. Якісно виконане проектування дозволяє значно знизити витрати на етапі безпосереднього написання коду оскільки більшість складних бізнес процесів та архітектурних рішень візуалізуються та перевіряються ще до моменту реалізації.

Для проектування інтернет ресторану обрано об'єктно орієнтований підхід із використанням уніфікованої мови моделювання UML. Це дозволяє створити цілісну несуперечливу та наочну модель системи яка буде зрозумілою для всіх учасників проекту. Основна увага в даному розділі приділяється функціональному моделюванню логічному структуруванню даних та опису динамічної поведінки застосунку при взаємодії з клієнтами та адміністраторами закладу. Такий системний підхід гарантує що фінальний продукт на базі WordPress та WooCommerce буде стабільним масштабованим та відповідатиме всім заявленим специфікаціям.

2.1 Моделювання діаграм варіантів використання

Побудова функціональної моделі майбутньої системи починається з детального визначення того як саме зовнішні об'єкти будуть взаємодіяти з програмним продуктом. Найбільш ефективним та загальноприйнятим інструментом для такого опису виступає діаграма варіантів використання або прецедентів. Дана модель дозволяє візуалізувати функціональність системи через набір конкретних цілей які переслідують різні категорії користувачів [12].

Використання стандартизованих засобів моделювання та сучасних CASE засобів на цьому етапі проектування забезпечує високу точність відображення всіх операційних процесів майбутнього інтернет ресторану [13].

Діаграма варіантів використання дозволяє розробнику ще до початку безпосереднього програмування побачити повну картину функціональних можливостей проекту. Вона виступає своєрідним містком між технічним описом системи та реальними потребами ресторанного бізнесу. Головними компонентами такої діаграми є актори та варіанти використання. Актори представляють собою зовнішні сутності які взаємодіють із застосунком тоді як варіанти використання описують конкретні сервіси що надаються цим акторам [14]. Для нашої інформаційної системи було ідентифіковано двох ключових акторів а саме зовнішній Клієнт та внутрішній Адміністратор.

2.1.1 Моделювання можливостей для актора Клієнт

Діаграма варіантів використання для клієнта відображає розширений спектр можливостей які надає інформаційна система для забезпечення високої якості обслуговування. Клієнт виступає ключовим зовнішнім актором чия взаємодія з платформою охоплює весь цикл від ознайомлення з асортиментом до управління власним профілем та здійснення повторних покупок. Саму діаграму варіантів використання для даного актора зображено на рисунку 2.1.

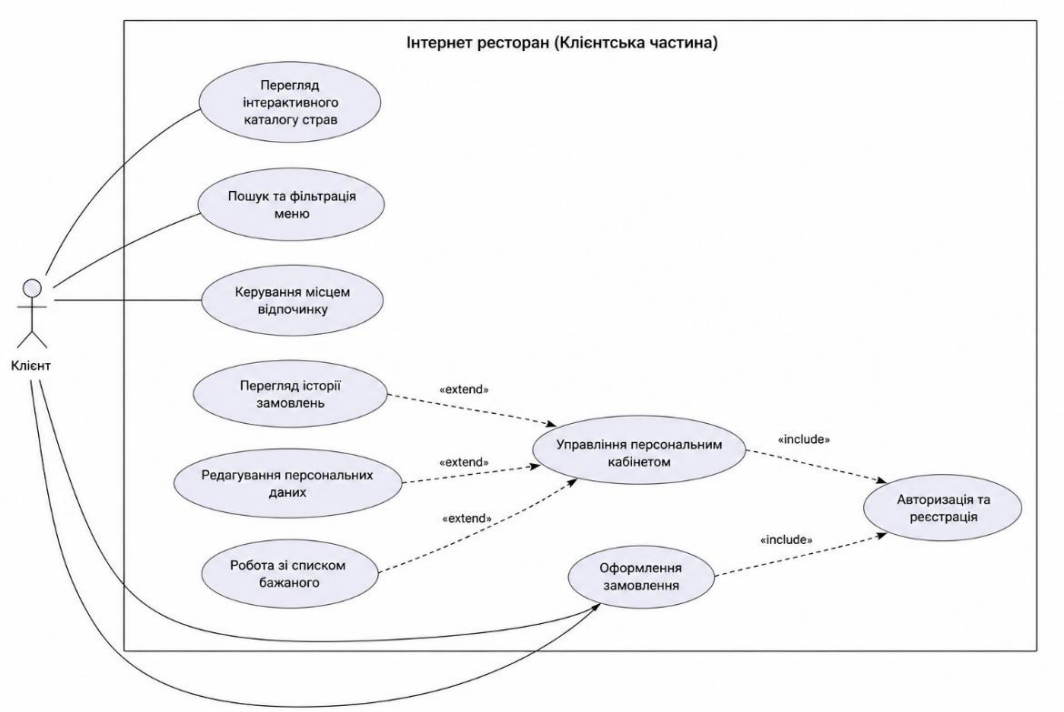


Рисунок 2.1 – Діаграма варіантів використання для користувача

Проектування функціоналу для клієнтської частини базується на створенні безперервного та інтуїтивно зрозумілого шляху користувача. Кожен прецедент на діаграмі відповідає конкретній бізнес цілі яку клієнт може досягти за допомогою системи [12].

Детальний аналіз основних функціональних блоків включає такі аспекти:

– Інформаційна взаємодія та пошук страв. Користувач переглядає інтерактивний каталог та використовує механізми швидкого пошуку і фільтрації. Це дозволяє миттєво знаходити потрібні позиції що особливо зручно при скануванні QR кодів безпосередньо у закладі;

– Комерційний цикл та обробка замовлень. Клієнт керує вмістом кошика для гнучкого формування списку покупок. Головним етапом є оформлення онлайн замовлення яке згідно зі стандартами UML обов'язково включає прецедент авторизації через відношення включення (include) для захисту даних;

– Персоналізація сервісу. У персональному кабінеті клієнт може працювати зі списком бажаного зберігаючи улюблені страви для майбутніх візитів. Цей функціонал реалізовано через відношення розширення (extend);

– Управління історією та даними. Кабінет надає доступ до історії транзакцій дозволяючи швидко повторювати покупки одним кліком. Також користувач має змогу самостійно редагувати контактну інформацію та адреси доставки.

Використання такої деталізованої моделі дозволяє наочно продемонструвати як технічні компоненти платформи інтегруються для вирішення завдань користувача [13].

2.1.2 Моделювання можливостей для актора Адміністратор

Діаграма варіантів використання для адміністратора фокусується на внутрішніх управлінських процесах які забезпечують безперебійну життєдіяльність інформаційної системи інтернет ресторану. Адміністратор виступає головним модератором та оператором платформи який володіє найвищим рівнем доступу до бази даних. Його головна мета полягає у підтримці актуальності контенту швидкій обробці вхідних заявок та аналізі комерційної ефективності закладу. Діаграму варіантів використання для адміністратора зображено на рисунку 2.2.

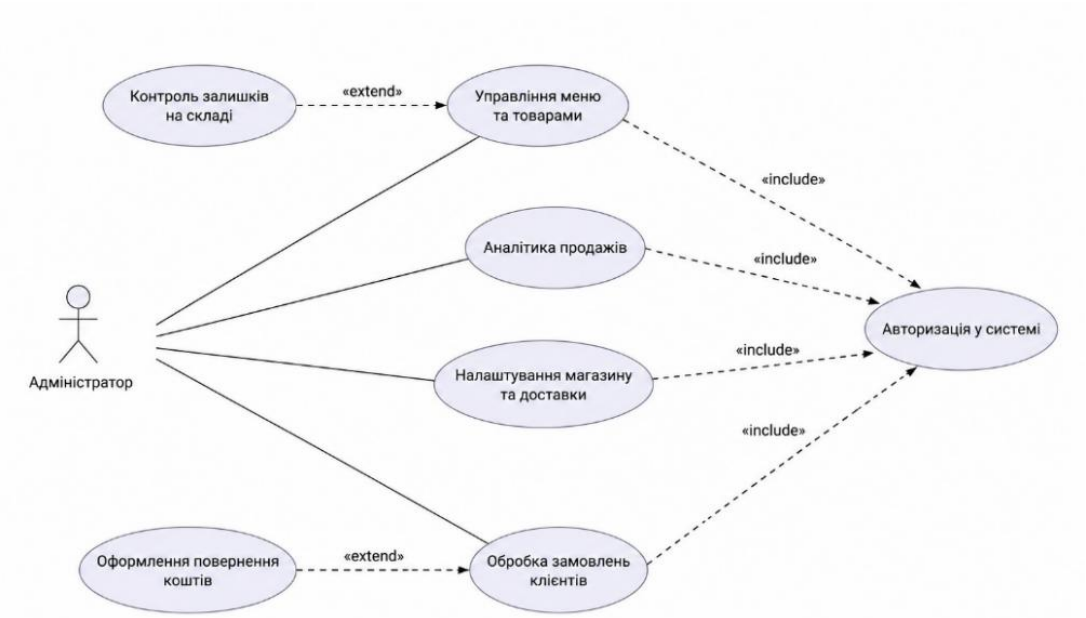


Рисунок 2.2 – Діаграма варіантів використання для адміністратора

Проектування адміністративної панелі базується на чіткій ієрархії де головні процеси розширюються додатковими специфічними сценаріями.

Детальний аналіз спрощеної моделі включає такі аспекти:

– Фундаментальна безпека. Абсолютно всі управлінські дії вимагають попередньої ідентифікації особи. Саме тому базові процеси мають відношення включення (include) до прецеденту авторизації. Це гарантує захист комерційної таємниці та персональних даних покупців;

– Управління каталогом та запасами. Адміністратор здійснює повний контроль над електронним меню. Цей базовий варіант використання розширюється (extend) прецедентом контролю залишків на складі. Таке розгалуження ілюструє автоматизовану функцію WooCommerce яка самостійно приховує страви якщо на кухні закінчилися необхідні інгредієнти;

– Обробка замовлень клієнтів. Даний блок відповідає за головний операційний цикл. Адміністратор перевіряє нові заявки та змінює їхні статуси. Процес має важливе розширення для електронної комерції а саме можливість оформлення швидкого повернення коштів у разі скасування замовлення гостем;

– Налаштування магазину та доставка. Прецедент налаштування акумулює в собі управління зонами доставки та конфігурацію платіжних систем. Окремим незалежним блоком виступає аналітика продажів яка дозволяє керівництву закладу формувати звіти та приймати стратегічні управлінські рішення на основі зібраної статистики.

Отже розроблені діаграми варіантів використання дозволили чітко зафіксувати функціональні межі інформаційної системи та визначити ролі основних категорій користувачів. Детальна візуалізація сценаріїв взаємодії для клієнта та адміністратора забезпечила глибоке розуміння бізнес процесів інтернет ресторану що виступає надійним фундаментом для подальшого проектування структури бази даних та об'єктної моделі.

2.2 Моделювання діаграми класів

Після успішного визначення функціональних вимог та побудови моделей варіантів використання процес проектування переходить до етапу розробки логічної структури системи. Головним інструментом для візуалізації статичної архітектури програмного забезпечення виступає діаграма класів. Вона є центральним компонентом уніфікованої мови моделювання UML оскільки саме на її основі здійснюється подальша розробка реляційної бази даних та безпосереднє написання програмного коду.

Діаграма класів демонструє з яких саме структурних елементів складається вебзастосунок які властивості вони мають та яким чином взаємодіють між собою для виконання бізнес логіки. Згідно з концепцією об'єктно орієнтованого підходу кожна фізична чи логічна сутність предметної області представляється у вигляді окремого класу [13]. У контексті інтернет ресторану такими сутностями є зареєстровані користувачі страви з меню відгуки або сформовані замовлення.

Графічно кожен клас на діаграмі зображується у вигляді прямокутника який розділений на три секції. У верхній секції вказується унікальна назва класу яка має чітко відображати його суть. Середня секція містить перелік атрибутів тобто характеристик які описують стан об'єкта. Для страви це може бути назва ціна категорія та грамаж. У нижній секції записуються методи які визначають поведінку класу та операції які він може виконувати з власними даними [15].

Не менш важливим етапом моделювання є встановлення зв'язків між виділеними класами. Інформаційна система інтернет ресторану містить багато взаємопов'язаних сутностей тому для їх логічного об'єднання використовуються різні типи відношень:

– Асоціація: показує базовий структурний зв'язок між об'єктами наприклад коли клієнт здійснює оплату;

– Агрегація та композиція показують зв'язок між головним об'єктом та його частинами. Наприклад кошик покупця містить у собі окремі вибрані страви. Це є яскравим прикладом композиції оскільки додані товари не мають сенсу і не можуть

існувати в системі самі по собі якщо видалити сам кошик;

– Узагальнення або успадкування: дозволяє виділити спільні атрибути в базовий клас та передати їх дочірнім класам. У нашій системі адміністратор та клієнт можуть успадковувати спільні властивості від базового класу Користувач;

Побудова правильної діаграми класів дозволяє розробнику уникнути дублювання коду оптимізувати майбутню структуру таблиць у базі даних та закласти міцний фундамент для легкого масштабування платформи інтернет ресторану.

Для проектування архітектури інтернет ресторану та підготовки структури бази даних розроблено об'єктну модель платформи яка відображає ключові сутності системи, їхні атрибути та методи взаємодії. Розширену діаграму класів із відображенням управлінських модулів зображено на рисунку 2.3.

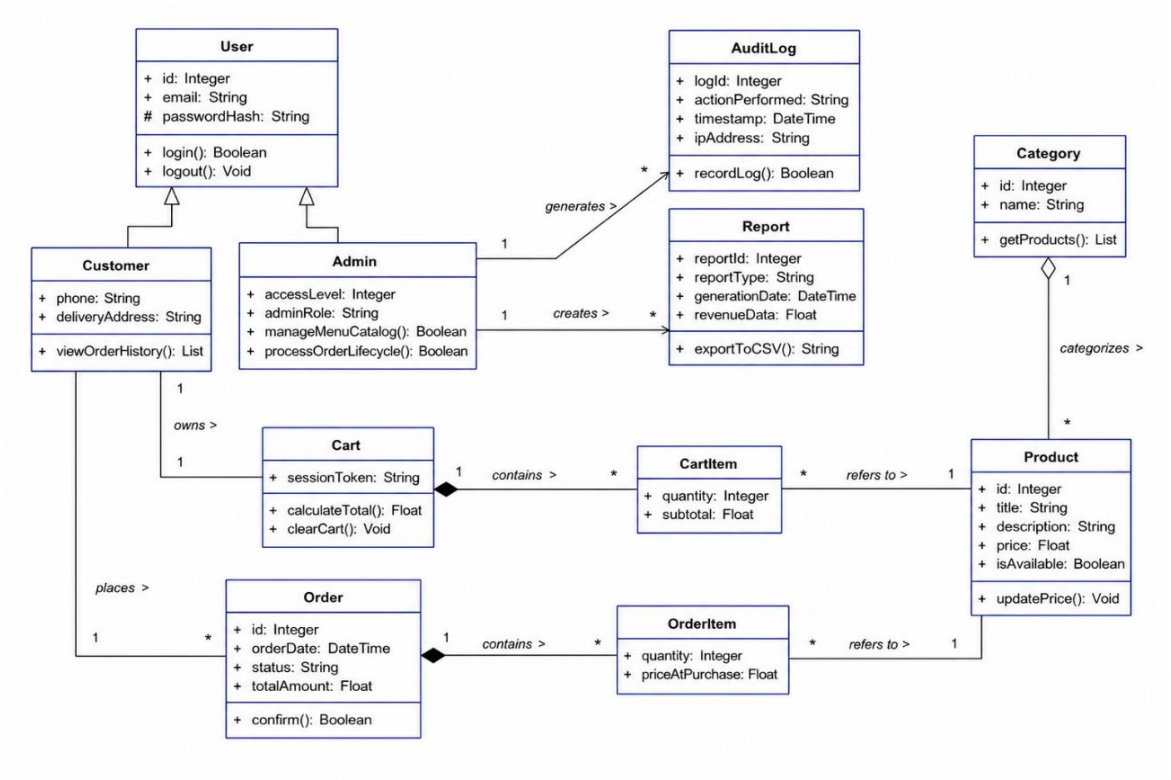


Рисунок 2.3 – Діаграма класів системи

На представленій діаграмі чітко виділено ключові логічні блоки інформаційної системи інтернет ресторану. Базовим компонентом виступає клас User який зберігає загальні облікові дані. Від нього за допомогою відношення

узагальнення успадковують свої властивості класи Customer та Admin що дозволяє ефективно розділити права доступу та обов'язки всередині платформи.

Функціональний простір адміністратора розширено додатковими сутностями для забезпечення контролю та аналітики. Клас Admin генерує записи у класі AuditLog для відстеження історії дій персоналу та створює об'єкти класу Report для формування фінансових звітів.

Ядро електронної комерції складається з каталогу та модулів обробки покупок. Клас Category класифікує об'єкти класу Product утворюючи структуроване меню закладу. Процес вибору страв клієнтом реалізовано через класи Cart та CartItem де гість володіє кошиком який містить обрані позиції. При остаточному оформленні покупки клієнт розміщує об'єкт Order який зберігає фінальні позиції чека у класі OrderItem. Такий архітектурний поділ гарантує що зміна ціни на страву в майбутньому ніяк не вплине на суми в історії старих замовлень.

Для більш глибокого розуміння спроектованої архітектури та призначення кожного окремого елемента в таблиці 2.1 наведено таблицю з детальним описом усіх класів системи.

Таблиця 2.1 – Детальний опис всіх класів системи

№	**Назва класу**	**Опис класу**
1	User	Базовий клас який представляє будь якого зареєстрованого користувача платформи та містить загальні атрибути такі як електронна пошта та пароль для авторизації в системі.
2	Customer	Похідний від користувача клас який представляє клієнта інтернет ресторану та зберігає його контактний номер телефону і адресу для доставки замовлень.
3	Admin	Похідний від користувача клас який представляє менеджера або адміністратора закладу. Він надає розширені права для управління каталогом та обробки вхідних заявок.

Продовження таблиці 2.1

1	**2**	**3**
4	AuditLog	Клас який відповідає за фіксацію всіх дій адміністратора у системі управління контентом. Він зберігає час виконання операції та айпі адресу для забезпечення безпеки.
5	Report	Клас який автоматично генерує фінансову аналітику та звіти про доходи ресторану дозволяючи експортувати дані для бухгалтерського обліку.
6	Category	Клас який використовується для логічного групування страв у меню наприклад десерти чи напої що полегшує пошук для клієнта.
7	Product	Клас який описує конкретну страву з каталогу. Він містить інформацію про її назву склад актуальну ціну та статус наявності на кухні.
8	Cart	Клас який представляє поточний кошик покупця під час його сесії на сайті. Він відповідає за накопичення обраних страв та розрахунок їхньої загальної вартості.
9	CartItem	Клас який представляє окрему позицію всередині кошика. Він зберігає кількість обраних порцій конкретної страви та проміжну суму.
10	Order	Клас який акумулює всю інформацію про фіналізоване замовлення клієнта включаючи час створення загальну суму та поточний статус виконання.
11	OrderItem	Клас який фіксує деталі конкретної страви в уже оформленому замовленні. Він зберігає ціну товару саме на момент покупки щоб уникнути помилок при майбутній зміні цін у меню.

Окрім загального огляду необхідно більш детально проаналізувати ті компоненти системи які відіграють найважливішу роль у забезпеченні життєдіяльності платформи. Такий поглиблений розгляд дає змогу краще зрозуміти логіку збереження інформації та функціонування ядра електронної комерції.

Клас Product є основою всього каталогу та відповідає за відображення кожної

окремої страви яку ресторан пропонує своїм гостям. Його внутрішня структура складається з низки важливих полів та методів:

– Поле id виконує роль первинного ключа та гарантує унікальність запису в базі даних;

– Поле title містить текстову назву яка безпосередньо виводиться в інтерфейсі користувача на сторінці меню;

– Поле description зберігає детальну інформацію про складові страви та її харчову цінність;

– Поле price має тип числа з рухомою комою і фіксує поточну вартість продажу;

– Індикатор isAvailable автоматично приховує товар з публічного доступу якщо на кухні закінчилися необхідні інгредієнти;

– Метод updatePrice дає змогу персоналу оперативно коригувати цінову політику безпосередньо в каталозі.

Клас Order відіграє роль головного вузла у процесі комерційної взаємодії акумулюючи всі дані після того як користувач підтвердив свій вибір у кошику. Ця сутність ідентифікується та обробляється за допомогою наступних складових:

– Поле id є однозначним номером кожної створеної транзакції для її відстеження;

– Точний час генерації заявки записується автоматично у спеціальне поле orderDate;

– Текстовий маркер status використовується для інформування клієнта та персоналу про етап приготування;

– Підсумкова сума до сплати з урахуванням усіх можливих нарахувань зберігається в полі totalAmount;

– Функціональний метод confirm запускає ланцюжок подій для валідації оплати та передачі інформації кухарям;

– Прямий зв'язок із допоміжною сутністю OrderItem дозволяє робити історичний зріз ціни на момент транзакції щоб убезпечити звітність від майбутніх змін у прайсі.

Клас Admin моделює дії співробітників закладу які наділені спеціальними правами для адміністрування платформи та контролюють ключові бізнес процеси. Для реалізації цих повноважень клас використовує такі характеристики та операції:

– Числове поле accessLevel визначає глибину доступу співробітника до налаштувань бази даних платформи;

– Текстове поле adminRole зберігає назву посади для правильного розподілу обов'язків у команді;

– Метод manageMenuCatalog призначений для повноцінної роботи з асортиментом та редагування товарів;

– Метод processOrderLifecycle є необхідним для переведення клієнтських замовлень між різними стадіями виконання;

– Взаємодія з класом AuditLog гарантує створення записів безпеки про всі виконані дії менеджера;

– Взаємодія з класом Report дозволяє ініціювати формування фінансових виписок для бухгалтерського обліку.

Під час проектування архітектури програмного забезпечення надзвичайно важливу роль відіграє використання шаблонів проектування. Вони являють собою перевірені часом та стандартизовані рішення для типових проблем які виникають у процесі розробки. Замість того щоб щоразу винаходити нові концепції розробники застосовують готові патерни які гарантують високу надійність та продуктивність вебзастосунку [16].

У рамках створення інформаційної системи інтернет ресторану на базі WordPress та WooCommerce було використано кілька ключових архітектурних шаблонів. Основним патерном який забезпечує ефективну взаємодію між компонентами платформи є шаблон Спостерігач. В екосистемі обраної системи управління контентом він реалізований через вбудовану глобальну систему подій та фільтрів. Цей підхід ідеально підходить для електронної комерції оскільки дозволяє гнучко реагувати на зміни стану об'єктів [17].

Зазначений патерн дозволяє різним класам системи виконувати свої дії без створення жорсткої залежності один від одного. Наприклад при зміні статусу в

класі Order система автоматично сповіщає інші модулі про необхідність виконання їхніх функцій. Таким чином модуль відправки листів самостійно генерує повідомлення клієнту а клас Report одразу отримує тригер для оновлення фінансової статистики закладу.

Додатково в архітектурі платформи активно застосовується архітектурний шаблон Одинак. Його головна мета це щоб певний клас мав лише один єдиний екземпляр з глобальною точкою доступу до нього. У нашій системі цей патерн ідеально підходить для класу Cart оскільки під час однієї сесії покупця на сайті може існувати лише один кошик що повністю виключає можливість дублювання товарів [18].

Для наочної демонстрації роботи архітектурних патернів у системі було спроектовано відповідні об'єктні моделі. На рисунку 2.4 який ілюструє шаблон «Спостерігач» показано взаємодію між головним класом OrderSubject та залежними компонентами. Клас замовлення містить методи для додавання та видалення спостерігачів а також функцію notifyObservers яка автоматично розсилає оновлення. Класи EmailNotifier та ReportUpdater реалізують спільний інтерфейс Observer що дозволяє їм приймати дані про зміну статусу замовлення.

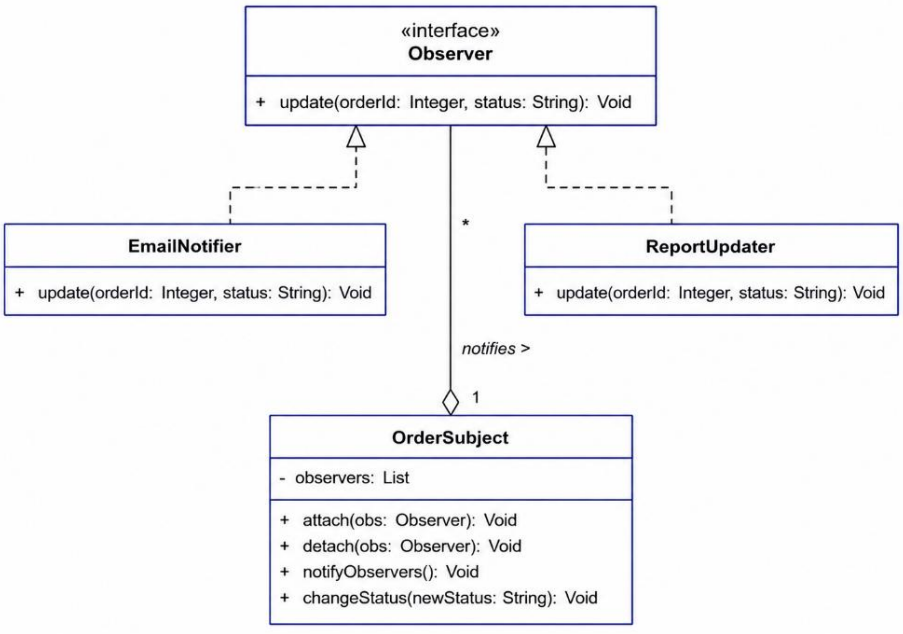


Рисунок 2.4 – Використання патерну «Спостерігач»

Функціонування шаблону «Одинак» та його взаємодію з іншими класами показано на рисунку 2.5. Клас клієнта має пряму залежність від сутності CartSingleton оскільки для додавання позицій меню та розрахунку фінальної вартості замовлення покупець повинен звертатися до активної сесії. При цьому сам кошик продовжує суворо контролювати свій життєвий цикл через внутрішні обмеження. Будь-який запит перенаправляється на публічний статичний метод який надає доступ до вже існуючого кошика в системній пам'яті що виключає ризик створення дублікатів під час роботи.

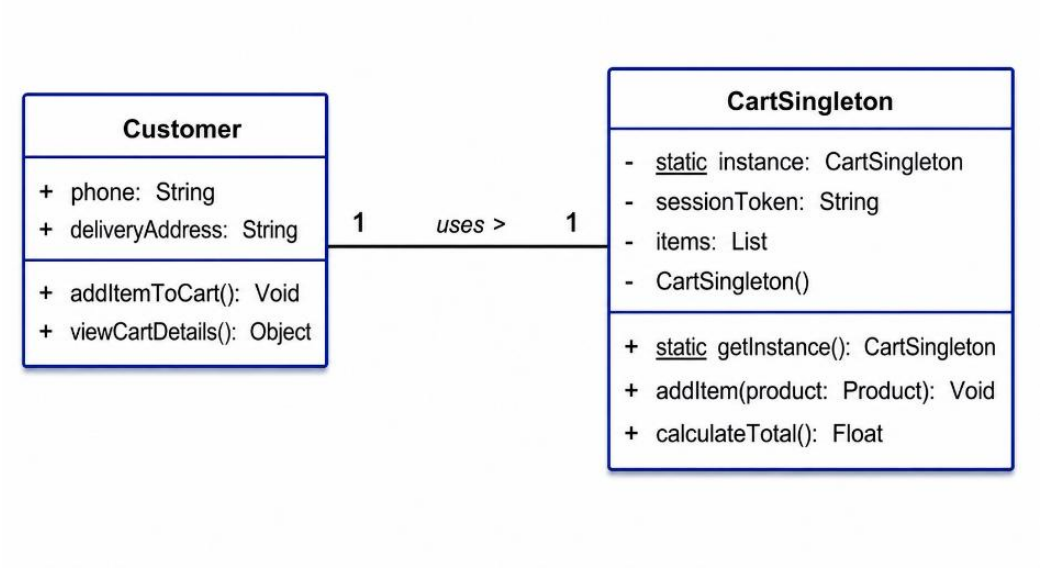


Рисунок 2.5 – Використання патерну «Одинак

Використання обраних архітектурних шаблонів робить систему інтернет ресторану стабільною та зручною для подальшого розширення. Ключовим рішенням у цій архітектурі є саме патерн Одинак який повністю захищає процес покупок від помилок та гарантує існування лише одного кошика для кожного гостя. У поєднанні з незалежною роботою інших модулів це дозволяє створити надійний вебзастосунок який дуже легко підтримувати та покращувати у майбутньому.

2.3 Моделювання діаграми послідовності

Динамічний аспект архітектури програмного забезпечення вимагає детального опису взаємодії об'єктів у часі під час виконання конкретних сценаріїв. Головним інструментом для візуалізації процесу обміну повідомленнями між різними компонентами системи є діаграма послідовності. Вона дозволяє розробнику чітко побачити порядок подій у межах певного прецеденту, відображаючи життєвий цикл об'єктів та логіку передачі керування всередині платформи [19].

Для детального аналізу динамічної поведінки розроблюваної платформи інтернет ресторану було обрано найважливіший бізнес процес а саме оформлення замовлення клієнтом. Цей сценарій охоплює взаємодію користувача з інтерфейсом сайту, перевірку даних у ядрі електронної комерції WooCommerce та збереження інформації. Повну модель взаємодії об'єктів у часі для цього процесу зображено на рисунку 2.6 де представлено детальну діаграму послідовності інформаційної системи.

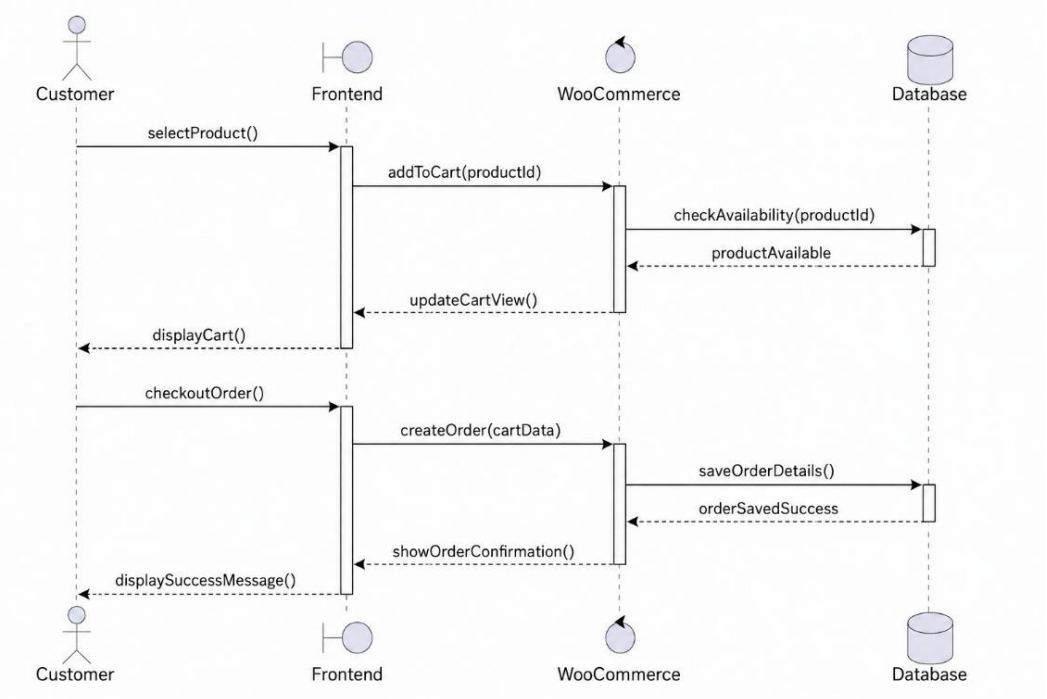


Рисунок 2.6 – Діаграма послідовності оформлення замовлення

На представленій схемі детально візуалізовано хронологію взаємодії та вертикальні лінії життєвого циклу об'єктів які беруть участь у процесі купівлі страв. Початковий імпульс виходить від актора Клієнт який ініціює вибір продукту через граничний елемент Інтерфейс. Подальша логіка обробки повністю делегується керуючому модулю WooCommerce який виконує синхронний запит до бази даних для миттєвої перевірки наявності обраної позиції на кухні закладу.

Отримавши позитивне підтвердження система оновлює графічне відображення кошика та повертає актуальний стан екрана користувачу що завершує перший етап комунікації.

Друга фаза динамічної моделі описує безпосередню фіналізацію та збереження комерційної транзакції. Після ініціації операції оформлення з боку покупця модуль WooCommerce приймає сформований масив даних кошика, проводить фінансові розрахунки та надсилає команду на збереження деталей транзакції у сховище даних. База даних виконує низькорівневу операцію запису, реєструє унікальний номер чека та повертає сигнал про успішне завершення процедури. На основі цієї відповіді керуючий компонент формує екран підтвердження, а інтерфейс виводить для клієнта фінальне повідомлення про успішне прийняття замовлення в обробку.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ ІНТЕРНЕТ-РЕСТОРАНОМ

Практична частина присвячена розробці та комплексному тестуванню програмного забезпечення. Матеріал розділу описує кроки створення графічного інтерфейсу та інтеграції інструментів електронної комерції. Окремо розглядається перевірка якості функціоналу за допомогою ручного й автоматизованого тестування а також підготовка програмного продукту до розгортання на сервері.

3.1 Структура та ініціалізація програмного коду

Процес розробки програмного забезпечення інтернет ресторану розпочався зі створення власної унікальної теми для WordPress. Відмова від використання готових шаблонних рішень дозволила забезпечити максимальну продуктивність та уникнути наявності зайвого коду. Нова програмна розробка отримала назву «Braty» та була розміщена у відповідній директорії локального сервера.

Для забезпечення правильної архітектури та зручності подальшої підтримки проєкту весь програмний код було розділено на логічні модулі. Ієрархію файлів та тек створеної теми зображено на рисунку 3.1 де відкрито робоче середовище.

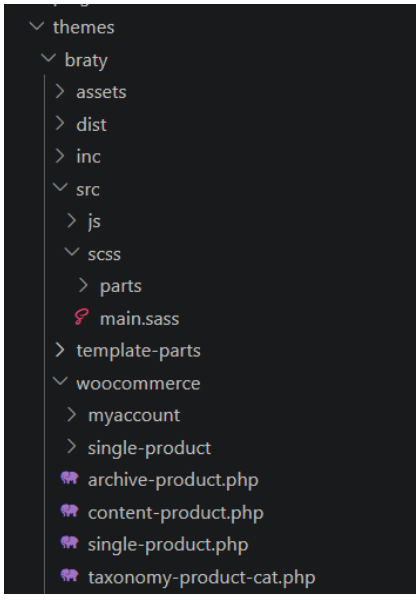


Рисунок 3.1 – Структура каталогів та файлів розробленої теми

Як видно з структури проєкт має чіткий поділ на вихідні ресурси та скомпільовані файли:

– Тека src містить нестиснені вихідні файли скриптів та стилів які використовуються під час написання програмного коду.

– Тека dist є кінцевою точкою куди система автоматизації зберігає готові мініфіковані файли стилів та скриптів які підключаються на сторінках.

– Тека template parts містить файли для відображення окремих блоків сайту що дозволяє ефективно повторно використовувати їх у різних місцях.

Спеціальна директорія woocommerce призначена виключно для перевизначення стандартних візуальних шаблонів електронної комерції під індивідуальний дизайн ресторану.

Для автоматизації процесів розробки було налаштовано інструментарій Gulp у поєднанні з модулем Webpack. Конфігураційний файл системи збірки містить інструкції для компіляції препроцесорів оптимізації коду та підтримки сучасного синтаксису за допомогою компілятора Babel. На лістингу 3.1 наведено фрагмент коду який відповідає за створення та мінімізацію Sass.

Лістинг 3.1 – Фрагмент коду для збірки Sass

```
const compileCss = () => {
  return gulp.src(`${template}/src/scss/main.scss`)
    .pipe(scss().on('error', scss.logError))
    .pipe(autoprefixer('last 2 versions'))
    .pipe(cssnano())
    .pipe(concat('main.css'))
    .pipe(rename({ suffix: '.min' }))
    .pipe(gulp.dest(`${template}/dist`));
};
```

Наведений алгоритм збирає всі стилі з робочої папки та стискає їх видаляючи всі зайві пробіли і зберігає готовий файл у фінальну директорію. Аналогічним чином в файлі працює функція збірки скриптів яка додатково розділяє загальний код та бібліотеки сторонніх розробників у окремий файл vendors для оптимізації кешування браузером клієнта.

3.2 Створення дизайну та його реалізація

Створений вебзастосунок включає масштабний каталог страв, особистий кабінет клієнта з усіма необхідними опціями управління профілем та окрему секцію з реальними відгуками де гості мають можливість залишати власні коментарі щодо якості обслуговування. Додатково в інтерфейс інтегровано блоки для демонстрації акційних та найбільш популярних пропозицій закладу, контактні форми для зворотного зв'язку з адміністрацією а також усі класичні функціональні сторінки ядра WooCommerce. Нижче детально розглянуто та представлено зображення лише ключових екранів які найкраще демонструють основний шлях користувача під час здійснення покупки.

Першим кроком у реалізації візуальної частини стало проєктування адаптивного та зрозумілого інтерфейсу. Головна мета полягала у створенні зручного середовища для відвідувачів яке б дозволяло швидко знаходити необхідні позиції меню та оформлювати замовлення з будь якого пристрою.

Головна сторінка вебзастосунку слугує візитною карткою закладу та містить ключову інформацію про найактуальніші пропозиції. Її дизайн спроєктовано таким чином щоб одразу привертати увагу відвідувача до асортименту. Загальний вигляд цього головного екрана представлено на рисунку 3.2.

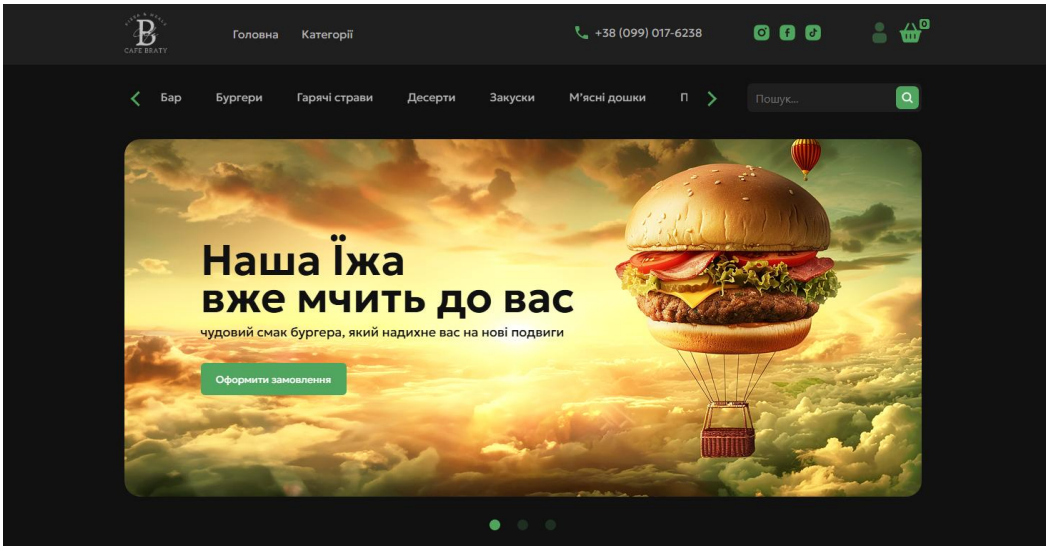


Рисунок 3.2 – Головний екран головної сторінки

Для зручної навігації по меню розроблено окрему сторінку категорій яку детально зображено на рисунку 3.3. На ній всі страви логічно згруповано за типом що дозволяє клієнту миттєво переходити до потрібного розділу без необхідності довгого гортання загального списку товарів.

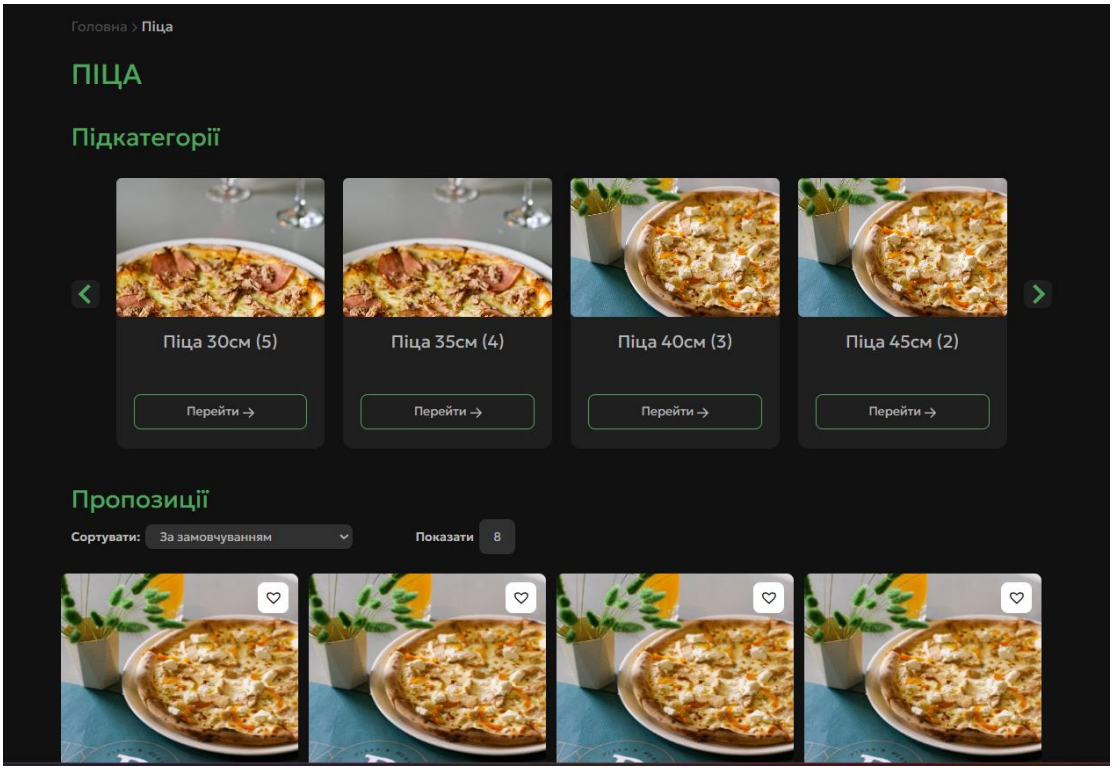


Рисунок 3.3 – Сторінка категорій

Кожна позиція асортименту має власну детальну сторінку зовнішній вигляд якої можна побачити на рисунку 3.4. На ній відображається високоякісне зображення актуальна ціна детальний опис інгредієнтів та помітна кнопка для миттєвого додавання обраної порції у кошик.

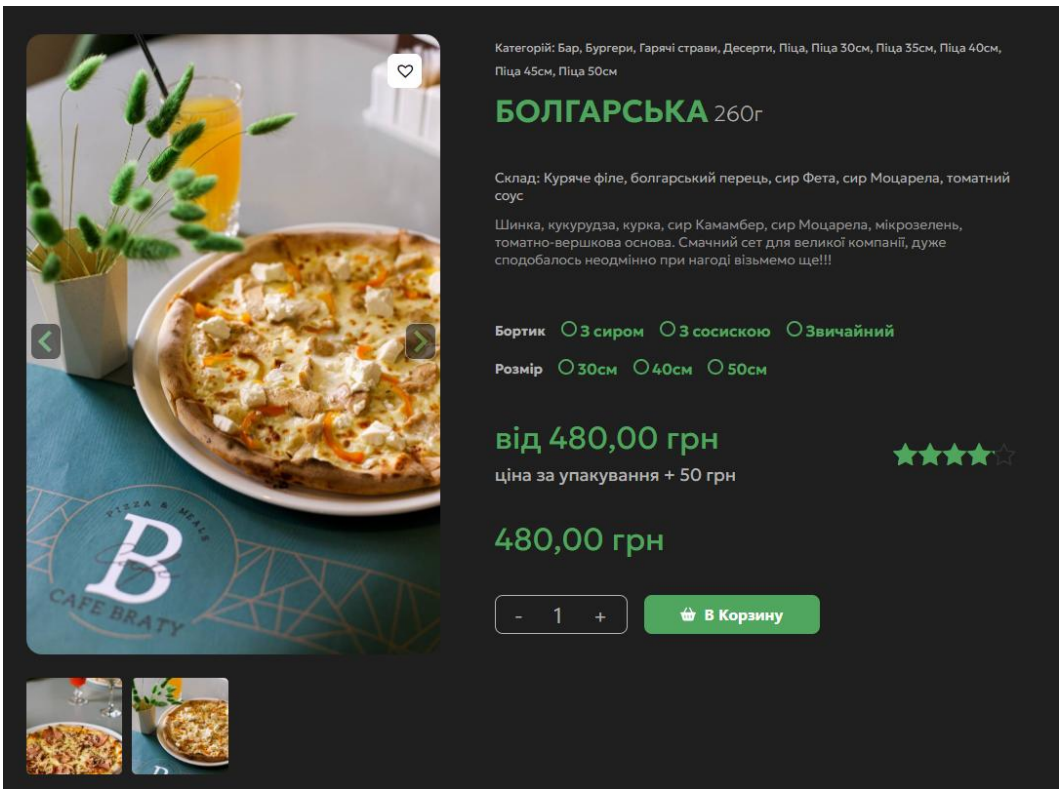


Рисунок 3.4 – Сторінка страви

Особистий кабінет користувача розширює базовий функціонал надаючи можливість зберігати вподобані позиції для майбутніх візитів. Для цього реалізовано окрему сторінку списку бажань як показано на рисунку 3.5 яка значно підвищує лояльність постійних клієнтів закладу.

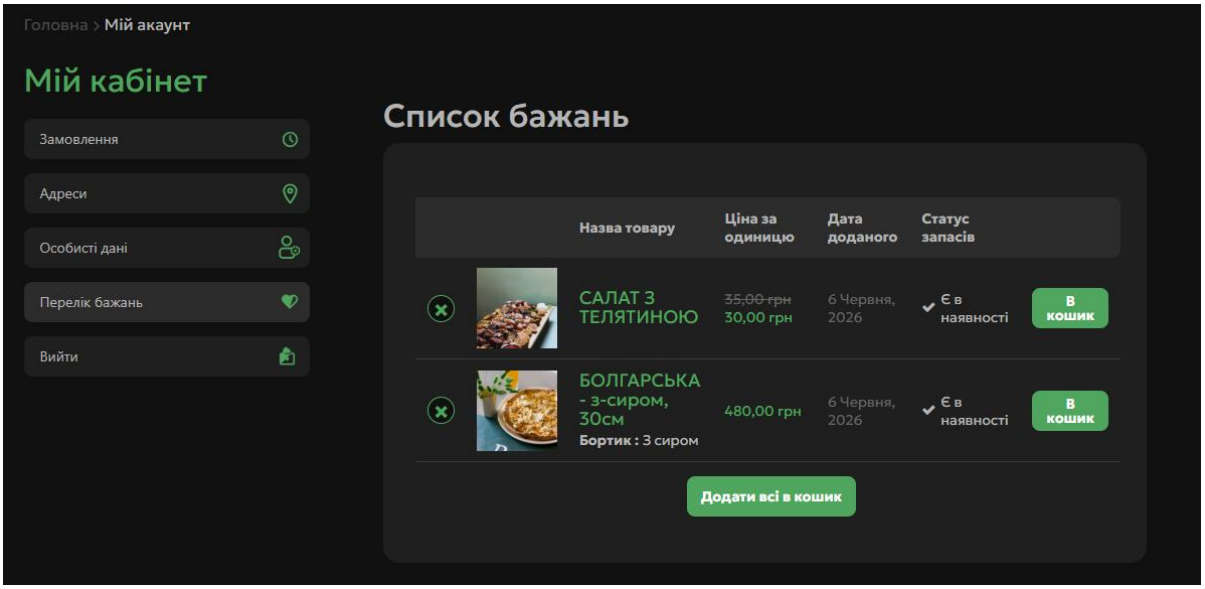


Рисунок 3.5 – Сторінка списку бажань в кабінеті

Процес купівлі починається зі сторінки кошика, що показано на рисунку 3.6, де покупець має змогу перевірити всі обрані страви змінити їхню кількість видалити зайве та одразу побачити попередню загальну вартість свого замовлення.

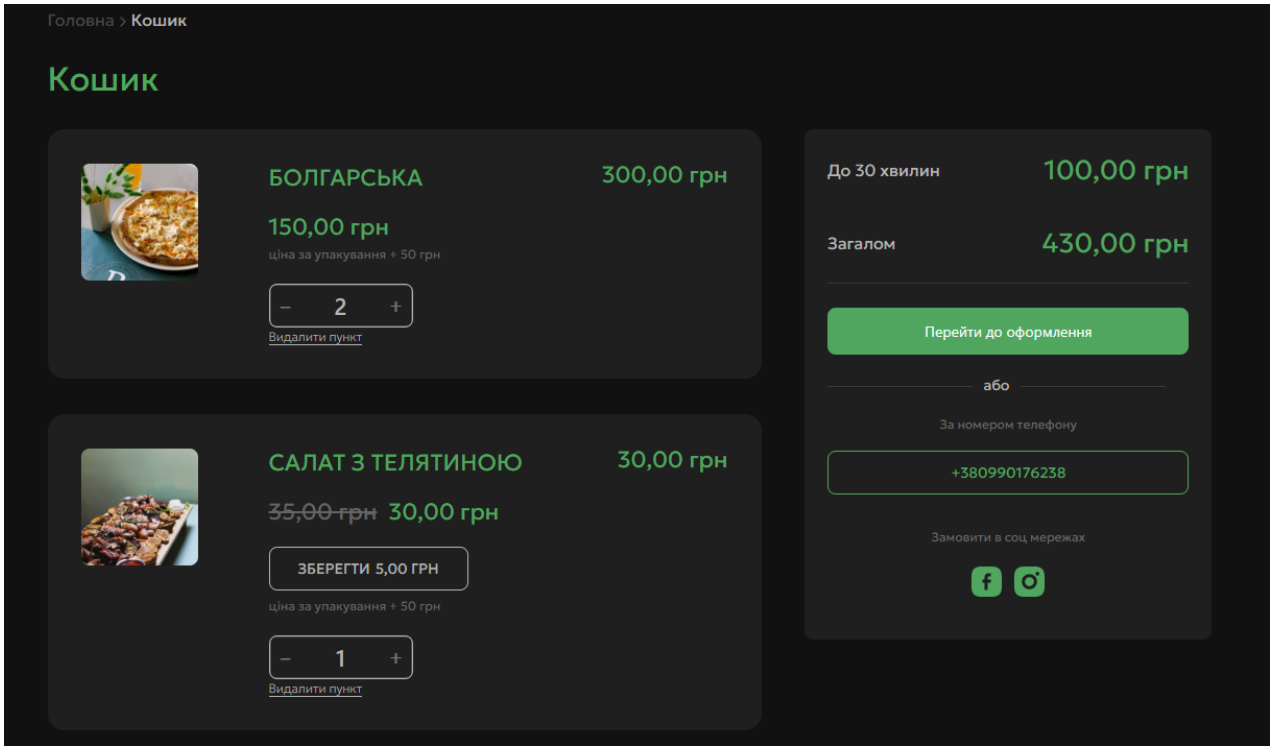


Рисунок 3.6 – Сторінка кошика

Далі користувач переходить до найважливішого етапу а саме безпосереднього оформлення покупки. Через велику кількість необхідної інформації цей процес візуально розділено на зручні логічні блоки. На першій частині екрана яку представлено на рисунку 3.7 гість заповнює контактні дані та вказує точну адресу для кур'єрської доставки.

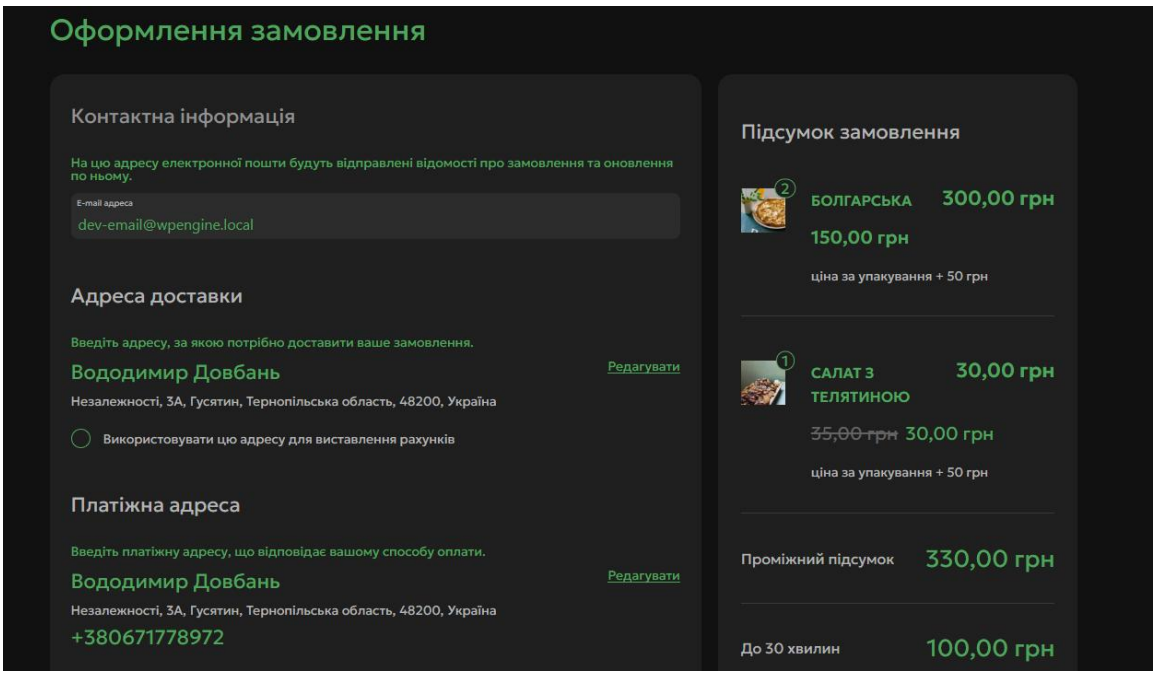


Рисунок 3.7 – Перша частина сторінки оформлення замовлення

На другій частині сторінки оформлення див рис 3.8 покупець обирає оптимальний метод оплати залишає коментарі для кухаря та остаточно підтверджує транзакцію. Інтерфейс цієї форми максимально оптимізовано для уникнення помилок при введенні платіжних даних.

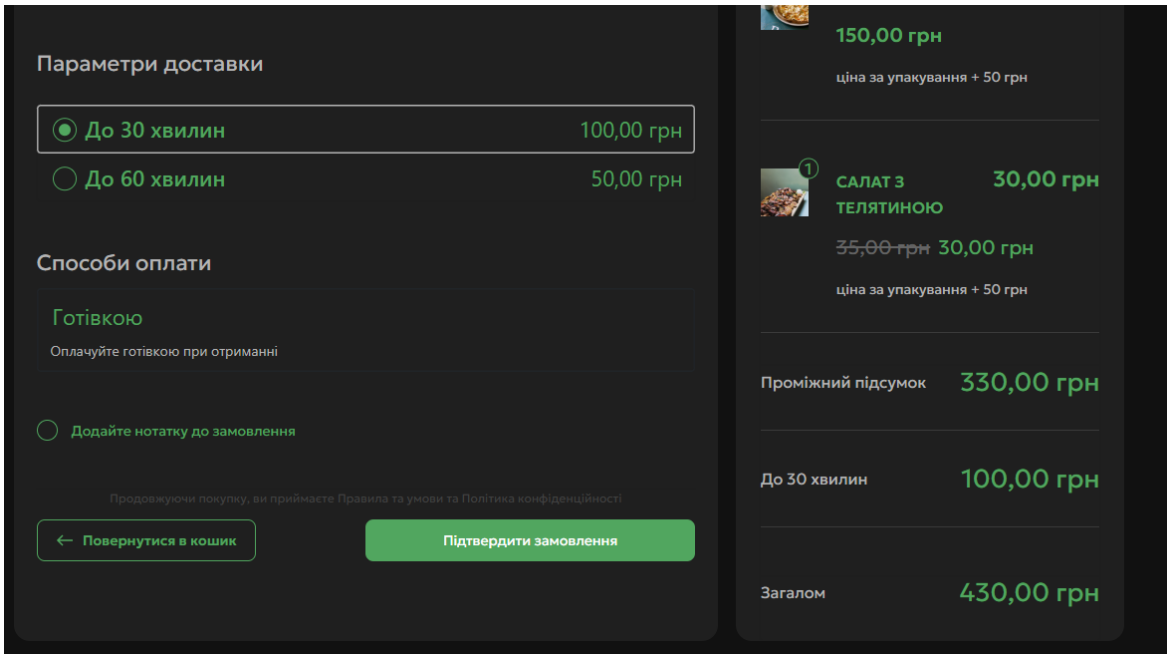


Рисунок 3.8 – Друга частина сторінки оформлення замовлення

Після успішного завершення всіх попередніх кроків система автоматично перенаправляє клієнта на сторінку очікування підтвердження яку зображено на рисунку 3.9. Тут виводиться унікальний номер успішної транзакції та орієнтовний час доставки що забезпечує якісний зворотний зв'язок та спокій для покупця. Окрім цього розроблений функціонал дозволяє користувачу в будь який момент відстежувати поточний статус свого замовлення безпосередньо в особистому кабінеті де автоматично зберігається повна історія покупок та детальна інформація про етапи приготування кожної страви.

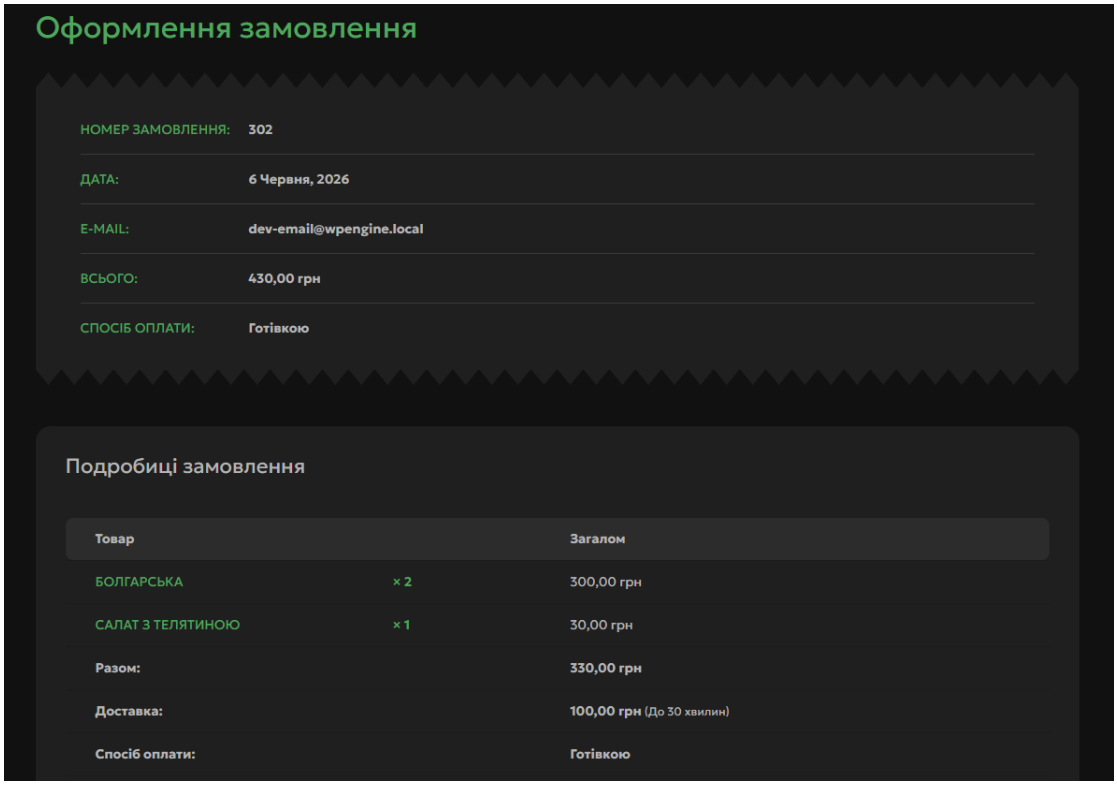


Рисунок 3.9 – Сторінка очікування підтвердження замовлення

Загалом візуальна реалізація проєкту повністю відповідає сучасним стандартам електронної комерції та гарантує високий рівень комфорту під час взаємодії користувача з платформою. Повний набір ілюстрацій усіх розроблених екранів детально представлено у додатку Б. Важливою технічною перевагою дизайну є гнучка організація коду де файл variables.scss має всі базові налаштування стилів. Завдяки цьому можна миттєво змінити загальний візуальний стиль сайту просто відредагувавши кілька значень у цьому документі.

Продумана модульна структура стилів дозволяє успішно застосовувати одні й ті самі класи для оформлення різних секцій платформи що повністю виключає шкідливе дублювання програмного коду. Це безпосередньо запобігає надмірному навантаженню на браузер клієнта забезпечує високу швидкість відображення каталогу та значно полегшує подальшу технічну підтримку ресурсу.

3.3 Інтеграція електронної комерції

Головним завданням на етапі програмування бізнес логіки стала кастомізація ядра електронної комерції. Стандартний функціонал базового модуля не міг повністю задовольнити вимоги інтернет-ресторану тому систему було суттєво модифіковано на рівні програмного коду. Під час проєктування архітектури та розробки власних алгоритмів взаємодії з базою даних застосовувалися офіційні стандарти програмування які регламентують безпечну та правильну інтеграцію з ядром платформи [22].

Для досягнення унікального дизайну каталогу було застосовано механізм перевизначення шаблонів. Перенесення ключових файлів платформи до директорії розробленої теми дозволило повністю контролювати виведення даних та успішно інтегрувати власну верстку. Яскравим прикладом такого підходу є кастомна сторінка окремої страви з інтерактивною фотогалереєю інформаційними блоками про алергени та системою відгуків. Повний програмний код цієї реалізації детально представлено у додатку В кваліфікаційної роботи.

Окрім візуальної частини значну увагу було приділено модифікації базової логіки роботи платформи. Для цього активно застосовувалися спеціальні функції перехоплення подій а саме екшени та фільтри. Цей інструментарій дозволяє втручатися у процеси системи без прямого редагування файлів самого плагіна що гарантує стабільну та безперебійну роботу сайту після майбутніх глобальних оновлень ядра.

Як приклад створення складнішої бізнес логіки було розроблено алгоритм динамічної зміни інтерфейсу каталогу залежно від попередніх дій користувача. За

допомогою звернення до глобального об'єкта кошика система перевіряє наявність кожної страви у списку покупок клієнта. Якщо обрана позиція вже присутня в замовленні стандартна дія кнопки каталогу відхиляється а замість неї генерується посилання для швидкого переходу до оформлення покупки. Програмну реалізацію цієї функції наведено у лістингу 3.2.

Лістинг 3.2 – Програмна реалізація динамічної зміни кнопки

```
function is_product_in_cart($product_id) {
    $cart = WC()->cart->get_cart();
    foreach ($cart as $cart_item) {
        if ($cart_item['product_id'] == $product_id) {
            return true;
        }
    }
    return false;
}

function custom_woocommerce_after_shop_loop_item() {
    global $product;
    $product_id = $product->get_id();

    if (is_product_in_cart($product_id)) {
        echo '<a href="' . wc_get_cart_url() . '" class="button
added-to-cart">Переглянути кошик </a>';
    } else {
        woocommerce_template_loop_add_to_cart();
    }
}
remove_action('woocommerce_after_shop_loop_item',
'woocommerce_template_loop_add_to_cart', 10);
add_action('woocommerce_after_shop_loop_item',
'custom_woocommerce_after_shop_loop_item', 10);
```

Такий комплексний підхід до програмування забезпечив створення максимально гнучкої та відмовостійкої платформи яка ідеально відповідає всім поставленим вимогам комерційного проєкту.

3.4 Тестування розробленої системи

Заключним етапом розробки програмного продукту стало його комплексне тестування. Оскільки створена платформа є комерційним інструментом інтернет-ресторану наявність помилок у логіці роботи може призвести до погіршення користувацького досвіду та втрати клієнтів. Тому головним завданням цього етапу стала перевірка відповідності розробленого функціоналу початковим вимогам системи.

Основним методом було обрано ручне функціональне тестування за принципом чорної скриньки. Функціональне тестування це один із видів тестування спрямованого на перевірку відповідностей функціональних вимог програмного забезпечення його реальним характеристикам [23]. Цей підхід дозволяє імітувати дії реального покупця перевіряючи систему виключно через її графічний інтерфейс без втручання у внутрішню архітектуру коду. Для систематизації процесу перевірки було використано набір тестових сценаріїв які охоплюють ключові алгоритми роботи платформи. Результати ключових тестів виконання наведено у таблиці 3.1.

Таблиця 3.1 – Результати ручного тестування

Назва	**Вимога**	**Статус**	**Примітки**
Рейтинг страви	F1	Зауваження	На сторінці страви варто виводити рейтинг числами також а не тільки зірочками
Зміна кількості страв в кошику	F2	Зауваження	При зміні кількості страв сума пропадає та оновлюється через пару секунд
Прокрут слайдеру категорій	F6	Не пройшов	Слайдер не крутиться в праву сторону тільки в ліву
Пошук страви з пустим полем	F6	Не пройшов	Якщо лишити поле пошуку пустим то виводяться всі страви без повідомлення

Продовження таблиці 3.1

1	**2**	**3**	**4**
Відображення всіх рецептів в меню	F1	Пройшов	Відповідає очікуваному результату
Перехід на наступну сторінку меню цифрою	F1	Пройшов	Відповідає очікуваному результату
Перехід на наступну сторінку меню стрілкою	F1	Пройшов	Відповідає очікуваному результату
Відображення інформації страв	F1	Пройшов	Відповідає очікуваному результату
Коректність кількості страв на сторінці	F1	Пройшов	Відповідає очікуваному результату
Зміна кількості страв на сторінці	F1	Пройшов	Відповідає очікуваному результату
Індикатор списку бажаного	F1	Пройшов	Відповідає очікуваному результату
Індикатор знижки	F1	Пройшов	Відповідає очікуваному результату
Перехід на сторінку страви	F1	Пройшов	Відповідає очікуваному результату

У наведеній вище таблиці використано спеціальні ідентифікатори функціональних вимог які були повністю сформовані на початковому етапі проектування. Процес ручного тестування відбувався шляхом послідовної перевірки кожної специфікації через графічний інтерфейс користувача:

– Вимога F1 – Перегляд меню ресторану з фото, описом і цінами;

– Вимога F2 – Додавання страв до кошика та оформлення замовлення онлайн;

– Вимога F3 – Вибір способу оплати: онлайн або при отриманні;

– Вимога F4 – Персональний кабінет з переліком всіх замовлень та списком бажаного;

– Вимога F5 – Зворотній зв’язок з власниками ресторану ;

– Вимога F6 – Пошук з фільтром і фільтрація страв за категоріями;

– Вимога F7 – Адаптивний дизайн для мобільних пристроїв;

– Вимога F8 – Головна сторінка, яка містить гарячі пропозиції, категорії, пошук страв, популярні позиції меню, відгуки, опис закладу та сайту;

– Вимога F9 – Наявність сторінки рецепту, в якій зібрана інформація про конкретну страву з можливістю замовити її та залишити відгук.

Розподіл тестових сценаріїв за цими категоріями дозволив системно підійти до оцінки готовності кожного модуля та зафіксувати проблемні місця для подальшого доопрацювання програмного коду.

Окремим та надзвичайно важливим етапом перевірки якості стало автоматизоване тестування критичного функціоналу. Для реалізації цього завдання було обрано інструмент Selenium. Програмний продукт Selenium це фреймворк призначений для автоматизації дій веббраузера який дозволяє імітувати поведінку та кліки реального користувача [24]. Використання цього інструментарію дозволяє виключити людський фактор та значно економити час розробника при тестуванні цілісності системи після внесення нових змін у програмний код.

Замість багаторазового ручного введення даних було розроблено спеціальний скрипт який самостійно тестує головний бізнес процес платформи, а саме повний сценарій оформлення замовлення. Процес успішного виконання цього тесту у середовищі розробки наочно представлено на рисунку 3.10.

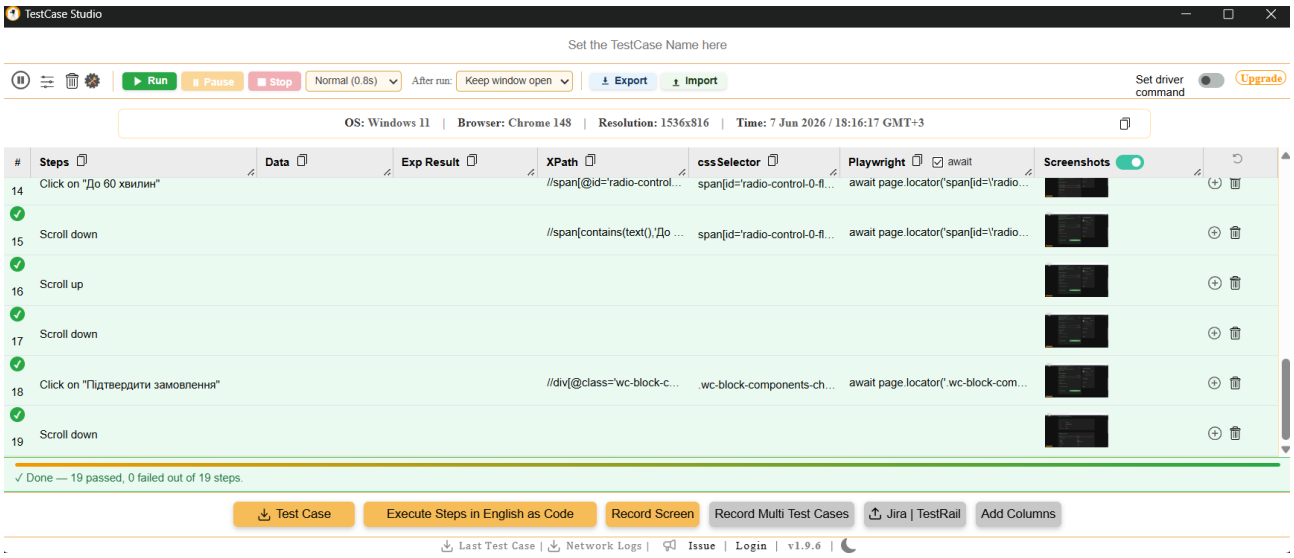


Рисунок 3.10 – Виконання автоматизованого тестування

Під час запуску тесту алгоритм автоматично відкриває сторінку інтернет ресторану знаходить потрібну страву та ініціює її додавання до кошика. Після цього скрипт програмно переходить на сторінку оформлення покупок і самостійно заповнює всі необхідні текстові поля включаючи ім'я клієнта прізвище номер телефону адресу електронної пошти та точну адресу доставки. На фінальному етапі автоматизований тест імітує затримку часу для повного завантаження графічного інтерфейсу та успішно натискає кнопку підтвердження транзакції. Такий професійний підхід забезпечив надійну перевірку найважливішого модуля платформи гарантуючи його стабільну роботу в реальних умовах експлуатації.

3.5 Підготовка до розгортання системи

Важливим кроком після завершення етапу розробки та успішного тестування є перенесення інформаційної системи з локального комп'ютера розробника на віддалений хостинг для повноцінного запуску. Цей процес вимагає чіткого узгодження системних вимог обох середовищ для забезпечення стабільної роботи платформи електронної комерції в реальних умовах.

Першим етапом підготовки став аналіз та фіксація параметрів локального середовища де проводилася безпосередня розробка інтернет ресторану. Для функціонування платформи було розгорнуто технологічний стек який включає вебсервер для обробки запитів, інтерпретатор мови PHP актуальної версії 8.2.29 та систему управління базами даних MySQL версії 8.0.35. Сама ж система керування контентом базується на версії WordPress версії 7.0 яка разом із зазначеним серверним програмним забезпеченням забезпечує високий рівень безпеки оптимізацію навантаження та повну сумісність із усіма інтегрованими модулями платформи.

Для забезпечення повного спектра бізнес функцій ресторану в локальному середовищі було активовано та налаштовано набір ключових плагінів. Серед них базовий модуль електронної комерції WooCommerce який відповідає за кошик каталог та транзакції інструмент Advanced Custom Fields Pro для управління

додатковими метаданими страв а також спеціалізовані розширення для забезпечення безпеки платформи. Усі ці компоненти утворюють цілісну екосистему вебресурсу.

Для безпосереднього здійснення міграції та розгортання було обрано інструмент автоматизованого перенесення даних All in One WP Migration. Цей плагін дозволяє створити повний зліпок інформаційної системи в один клік утворюючи єдиний пакетний файл конфігурації. До складу цього архіву автоматично включаються ядро системи вся база даних із користувачами та замовленнями медіафайли завантажених страв й активні плагіни разом із розробленою унікальною темою оформлення. Такий підхід гарантує що при розгортанні на віддаленому сервері хостингу структура сайту та його працездатність будуть повністю збережені без ризику втрати важливої інформації.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У сучасних умовах розвитку цифрових технологій особливої актуальності набувають питання забезпечення безпеки життєдіяльності та охорони праці. Це стосується і розробки програмного забезпечення для оптимізації роботи інтернет-ресторану, оскільки створення та супровід такої системи передбачає тривалу роботу фахівців за персональним комп'ютером, а безперервна діяльність закладу харчування потребує готовності персоналу до дій у надзвичайних ситуаціях.

Безпека життєдіяльності формує світогляд майбутнього фахівця, який у своїй повсякденній праці повинен будувати передумови запобігання нещасним випадкам, захворюванням та усувати негативний вплив шкідливостей на здоров'я людини в умовах виконання виробничих завдань, її існування в життєвому середовищі та в надзвичайних ситуаціях [25].

4.1 Аварії з викидом радіоактивних речовин

Аварії з викидом радіоактивних речовин належать до техногенних надзвичайних ситуацій. Вони супроводжуються порушенням нормального функціонування об'єктів, радіоактивним зараженням місцевості та можливим ураженням населення [26].

Причини таких аварій умовно поділяють на мирного та воєнного характеру. До першої групи належать вихід з ладу технічних споруд, пожежі на об'єктах атомної енергетики та промисловості, аварії під час транспортування радіоактивних матеріалів. Друга група пов'язана з надзвичайними ситуаціями воєнного характеру, коли застосування звичайних засобів ураження спричиняє вторинні наслідки: руйнування атомних станцій, промислових об'єктів із викидом радіоактивних речовин у довкілля [26].

Радіаційне ураження людини відбувається внаслідок зовнішнього опромінення та потрапляння радіонуклідів у організм. Найтиповіші наслідки: гостра променева хвороба, підвищений ризик онкологічних захворювань, тривале

забруднення територій.

Заходи захисту населення при аваріях з викидом радіоактивних речовин включають [26]:

– Укриття в приміщеннях з мінімальною проникністю випромінювання (підвали, центральні частини будівель, споруди цивільного захисту);

– Обмеження перебування на відкритій місцевості, особливо під час радіоактивного опаду;

– Захист органів дихання засобами індивідуального захисту;

– Прийом стабільного йоду за призначенням органів охорони здоров’я;

– Дезактивація одягу, взуття, шкіри, приміщень;

– Контроль радіаційної обстановки за допомогою дозиметрів і радіометрів.

На об’єктах господарювання для зниження ризику надзвичайних ситуацій передбачається підготовка фонду захисних споруд, створення запасів засобів індивідуального захисту, розробка планів ліквідації аварійних ситуацій, перевірка готовності системи оповіщення. Захист населення організовується відповідно до чинного законодавства у сфері цивільного захисту [27].

У сучасних умовах найбільш актуальною причиною аварій воєнного характеру є ракетні та дронові удари по критичній інфраструктурі. Сигнал повітряної тривоги попереджає не лише про небезпеку безпосереднього ураження, а й про можливість радіаційної загрози, якщо уражено об’єкти, де зберігаються або використовуються радіоактивні речовини.

Під час розробки програмного забезпечення для інтернет-ресторану працівники тривалий час перебувають у приміщенні за комп’ютером. Отримавши сигнал повітряної тривоги, необхідно негайно припинити роботу, зберегти дані, та знеструмити обладнання і прямувати до найближчого укриття. Укриття одночасно захищає від ураження уламками та зменшує дозу опромінення. Після відбою тривоги слід дочекатися офіційної інформації і лише тоді повертатися до роботи.

Якщо викид радіоактивних речовин уже відбувся, застосовують наведені вище заходи захисту. Окремо слід врахувати, що ураження може пошкодити електромережі, системи зв’язку та інтернет-підключення, через що порушується

робота серверної інфраструктури інтернет-ресторану. Тому доцільно передбачити резервне живлення та збереження критичних даних поза основним офісом.

Таким чином, розробники програмного забезпечення повинні знати причини та наслідки радіаційних аварій, вміти діяти під час повітряної тривоги та дотримуватися заходів захисту у разі викиду радіоактивних речовин.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Розробка програмного забезпечення для оптимізації роботи інтернет-ресторану належить до розумової праці. При цьому основне навантаження несе нервова система, що проявляється у зоровому та нервово-емоційному напруженні [28]. Програмісти, тестувальники та системні адміністратори керують процесами дистанційно, сприймаючи інформацію через екранні пристрої, тобто оцінюють стан системи за сигналами на індикаторах, а не безпосередньо в полі зору. Тому виникає завдання узгодження робочого місця та програмно-апаратних засобів з фізіологічними й психологічними можливостями людини [29].

У підручнику В.Ц. Жидецького розглядається система «людина, комп’ютер та середовище», яка формує основу ергономічного підходу до організації роботи користувачів ПК [29]. Робота з візуальними дисплейними терміналами пов’язана із зоровим і нервово-емоційним напруженням, виконується у вимушеній робочій позі при локальному напруженні верхніх кінцівок на фоні гіподинамії.

Для поліпшення умов праці ІТ-фахівців потрібно враховувати мікроклімат, склад повітря робочої зони, шум і освітлення. Стабільні мікроклімату підтримує терморегуляцію організму. Їхні різкі коливання знижують працездатність і можуть спричинити професійні захворювання [28].

У приміщеннях для роботи з ВДТ заборонено розміщувати робочі місця в підвалах і на цокольних поверхах. Площа на одне місце має бути не менше 6,0 м², об’єм не менше 20,0 м³. Приміщення оснащують аптечками, щоденно проводять вологе прибирання, обладнують побутові кімнати для відпочинку та кімнату психологічного розвантаження. Принтери й інше шумне устаткування

розташовують поза приміщенням для роботи з ВДТ.

Виробниче освітлення забезпечує сприятливу роботу зорового аналізатора. Для роботи з документами та екраном важливі рівномірний розподіл яскравості, відсутність блиску й засліпленості, сталість освітленості в часі. Для робочих місць з ВДТ освітленість у зоні документів має становити 300-500 лк, коефіцієнт пульсації не перевищує 5 %, загальне освітлення розташовують збоку, переважно ліворуч, яскравість бліків на екрані не перевищує 40 кд/м².

Організація робочого місця оператора ПК має відповідати ергономічним вимогам:

– Працівник має простір для зміни положення та рухів;

– Стіл і крісло регулюються по висоті, поверхня столу має низьку відбивну здатність;

– Екран, клавіатура і документи розміщують без зайвого напруження шиї, спини та очей;

– Зображення на екрані стабільне, яскравість і контрастність регулюються, екран не відблискує;

– Клавіатура відокремлена від екрана, екранні пристрої щодня очищують від пилу.

Режим праці та відпочинку передбачає регламентовані перерви. При 8-годинній зміні для розробників програм призначають перерву 15 хвилин через кожну годину роботи за ВДТ.

Роботодавець інформує працівників про умови праці, організовує навчання з охорони праці, медичні огляди та вживає заходів щодо зниження шуму і надлишкового тепловиділення від обладнання.

Працівникам ІТ-підрозділу рекомендується тримати екран на відстані 50-70 см від очей, чергувати роботу за комп’ютером з перервами для руху та своєчасно повідомляти про погіршення самопочуття [29].

У цьому розділі розглянуто два взаємодоповнюючі аспекти безпеки, безпосередньо пов’язані з розробкою програмного забезпечення для інтернет-ресторану. Перший стосується готовності ІТ-фахівців до дій у надзвичайних

ситуаціях: знання заходів захисту при радіаційних аваріях, правильна реакція на сигнали повітряної тривоги, збереження даних і технічної інфраструктури проєкту.

Другий охоплює організацію безпечної роботи за комп’ютером на основі вимог гігієни та ергономіки праці. Дотримання цих вимог знижує зорове та нервово-емоційне навантаження, запобігає професійним захворюванням і сприяє стабільній роботі колективу та надійному функціонуванню програмного продукту.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було проведено комплексне дослідження та реалізовано програмне забезпечення для оптимізації роботи інтернет-ресторану. На початковому етапі розробки було здійснено детальний аналіз предметної області та вивчено специфіку сучасного ресторанного бізнесу в умовах цифрової трансформації. Проведений аналіз конкурентних рішень на ринку довів що використання сторонніх сервісів доставки або хмарних конструкторів суттєво обмежує можливості закладу та знижує його прибутковість.

На етапі проектування було створено логічну архітектуру інформаційної системи яка забезпечує надійне збереження даних про клієнтів замовлення та асортимент страв. Для детальної візуалізації та чіткого структурування майбутнього програмного продукту було розроблено комплекс моделей та діаграм. Зокрема побудовано діаграми варіантів використання для визначення ролей та сценаріїв взаємодії різних акторів із системою.

Розроблений користувацький інтерфейс повністю враховує сучасні вимоги до ергономіки та забезпечує інтуїтивно зрозумілий процес вибору страв і здійснення покупки.

Програмну реалізацію серверної частини системи виконано на базі платформи управління контентом WordPress із використанням потужного розширення електронної комерції WooCommerce. Це забезпечило необхідну гнучкість налаштувань бізнес логіки та надало повний контроль над архітектурою бази даних.

Підсумовуючи результати можна стверджувати що мета дипломної роботи досягнута у повній мірі. Розроблене програмне забезпечення успішно вирішує ключові проблеми сучасного ресторанного бізнесу мінімізує людський фактор при ручному прийомі замовлень та надає закладу гнучкий незалежний інструмент для масштабування онлайн продажів. Готовий продукт відповідає всім поставленим вимогам і готовий до практичного впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Доставки Glovo в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://glovoapp.com/uk/ua;

2) ChoiceQR – онлайн QR меню для закладів [Електронний ресурс] – Режим доступу до ресурсу: https://choiceqr.com/uk/;

3) Сім'я ресторанів Файного Міста - fainemisto.com [Електронний ресурс] – Режим доступу до ресурсу: https://fainemisto.com/;

4) Вимоги до програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ Вимоги_до_програмного_забезпечення;

5) Курс «Аналіз вимог до програмного забезпечення (SE322)» [Електронний ресурс] – Режим доступу до ресурсу: https://dl.tntu.edu.ua/bounce.php?course=1559;

6) Local - Local WordPress development made simple [Електронний ресурс] – Режим доступу до ресурсу: https://localwp.com/;

7) WordPress Developer Resources [Електронний ресурс] – Режим доступу до ресурсу: https://developer.wordpress.org/;

8) WooCommerce Documentation [Електронний ресурс] – Режим доступу до ресурсу https://woocommerce.com/documentation/woocommerce/;

9) Bootstrap: Powerful, extensible, and feature-packed frontend toolkit [Електронний ресурс] – Режим доступу до ресурсу https://getbootstrap.com/docs/5.3/getting-started/introduction/;

10) Sass: Syntactically Awesome Style Sheets [Електронний ресурс] – Режим доступу до ресурсу https://sass-lang.com/documentation/;

11) Gulp.js: The streaming build system [Електронний ресурс] – Режим доступу до ресурсу https://gulpjs.com/docs/en/getting-started/quick-start/;

12) Процес ППЗ, керований варіантами використання [Електронний ресурс] – Режим доступу до ресурсу: https://dl.tntu.edu.ua/content.php?cid=213464;

13) М.Р. Петрик, І.Я. Мудрик Архітектура програмного забезпечення (на базі використання CASE-засобів IBM(sad)) навчальний посібник, Тернопіль: ТНТУ імені Івана Пулюя, 2017. 100с;

14) Unified Modeling Language (UML) основні поняття та принципи моделювання [Електронний ресурс] – Режим доступу до ресурсу https://uk.wikipedia.org/wiki/UML;

15) Діаграма класів UML основні поняття та зв'язки [Електронний ресурс] – Режим доступу до ресурсу https://uk.wikipedia.org/wiki/Діаграма_класів;

16) Патерни проектування каталог патернів [Електронний ресурс] – Режим доступу до ресурсу https://refactoring.guru/uk/design-patterns;

17) Патерн Спостерігач призначення та застосування [Електронний ресурс] – Режим доступу до ресурсу https://refactoring.guru/uk/design-patterns/observer;

18) Патерн Одинак призначення та застосування [Електронний ресурс] – Режим доступу до ресурсу https://refactoring.guru/uk/design-patterns/singleton;

19) Емблер С. Вступ до діаграм послідовності UML [Електронний ресурс] – Режим доступу до ресурсу https://agilemodeling.com/artifacts/sequenceDiagram.htm;

20) Дистанційний курс «Кваліфікаційна робота бакалавра» сайту дистанційного навчання ТНТУ [Електронний ресурс] – Режим доступу до ресурсу https://dl.tntu.edu.ua/bounce.php?course=6845;

21) Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М. – Тернопіль:ТНТУ імені Івана Пулюя, 2024. – 45 с.

22) WooCommerce Developer Resources [Електронний ресурс] – Режим доступу до ресурсу https://developer.woocommerce.com/;

23) База знань QALight. Функціональне тестування [Електронний ресурс] – Режим доступу до ресурсу https://qalight.ua/baza-znaniy/funktsionalne-testuvannya/;

24) Selenium Browser Automation [Електронний ресурс] – Режим доступу до ресурсу https://www.selenium.dev/;

25) Желібо Є.П. Безпека життєдіяльності : підручник / В. В. Зацарний. Київ : Каравела, 2023. 344 с;

26) Атаманчук П.С. Безпека життєдіяльності : навч. посіб. Київ : Центр учбової літератури, 2020. 276 с;

27) Кодекс цивільного захисту України : Закон України від 01.07.2013 № 5403-VII (із змінами);

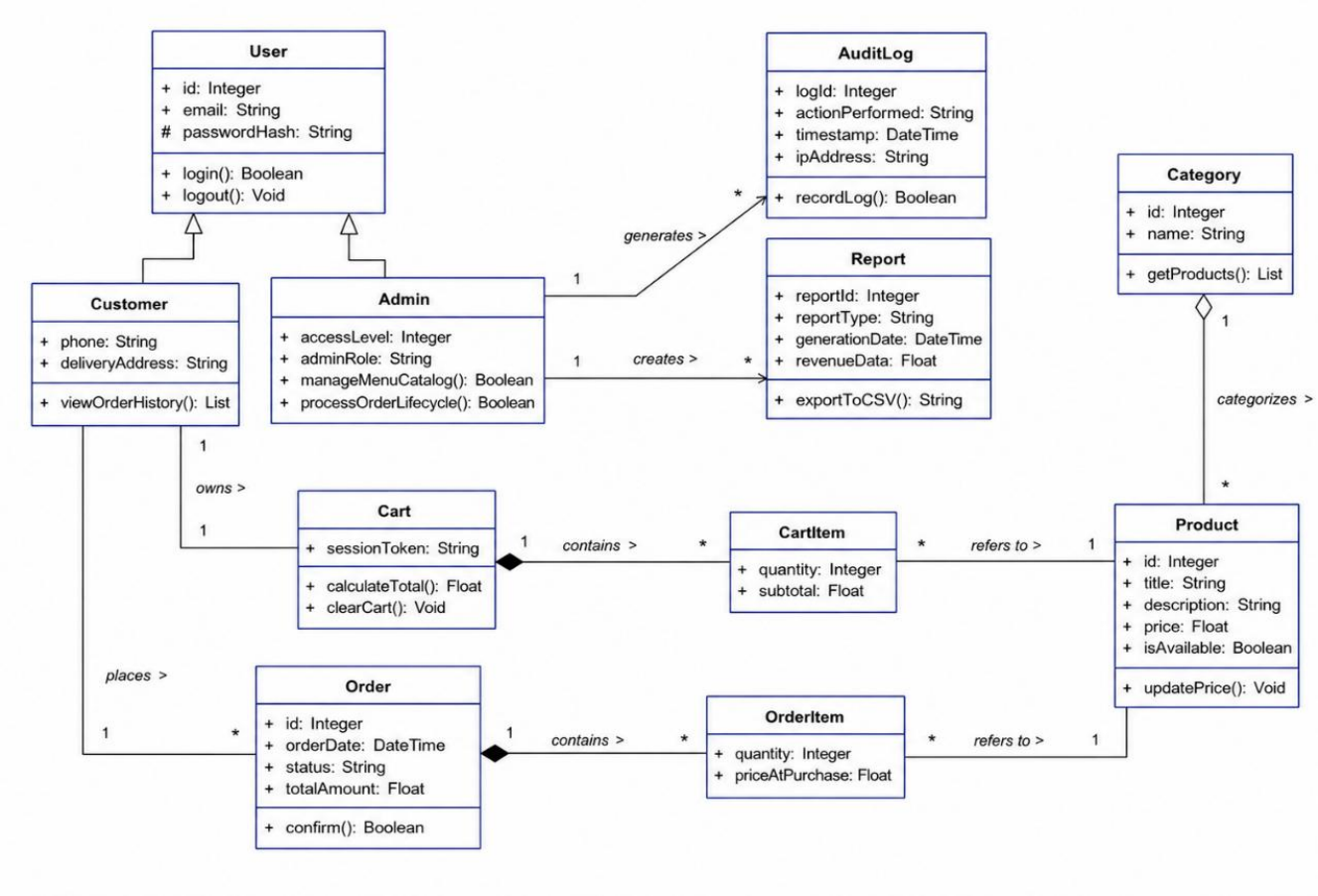
28) Фізіологія, гігієна та психологія праці [Електронний ресурс] – Режим доступу до ресурсу https://dl.tntu.edu.ua/content.php?cid=289144;

29) Жидецький В.Ц. Охорона праці користувачів комп’ютерів : підручник. Львів : Афіша, 2020. 176 с;

ДОДАТКИ

ДОДАТОК А

Діаграма класів програмного забезпечення



ДОДАТОК Б

Дизайн вебзастосунку

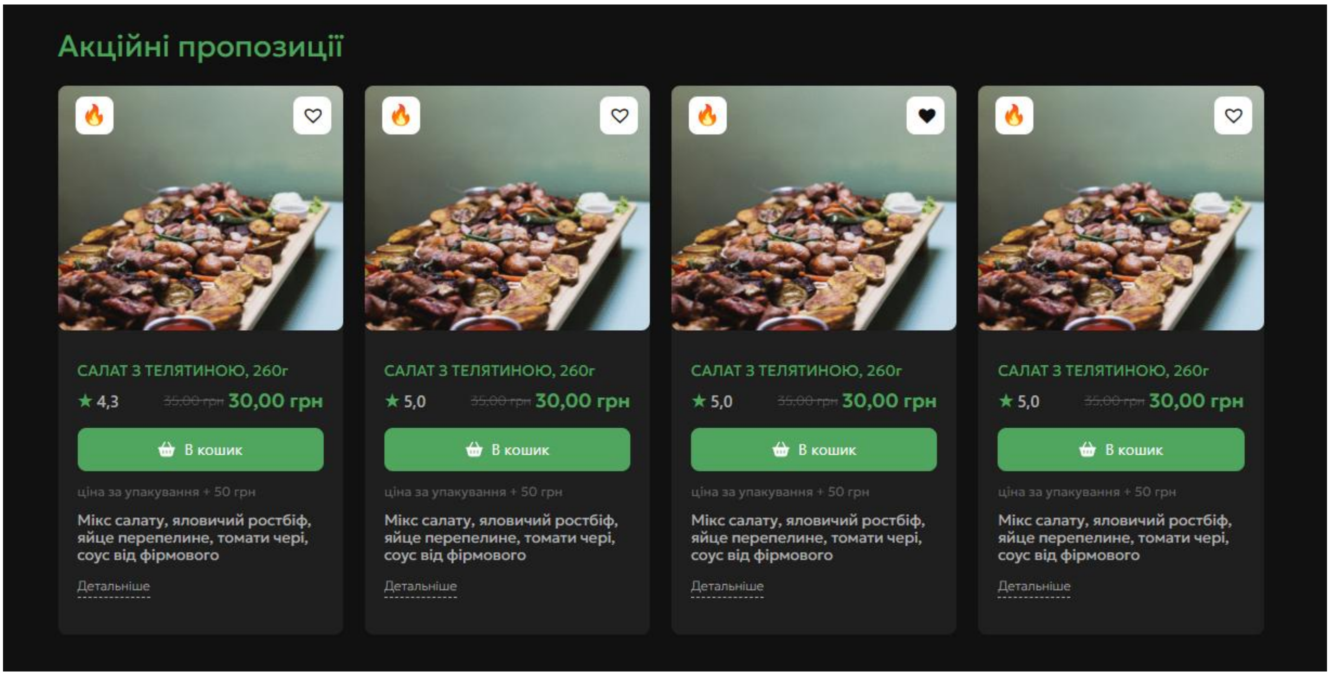


Рисунок В.1 – Секція акційних пропозицій

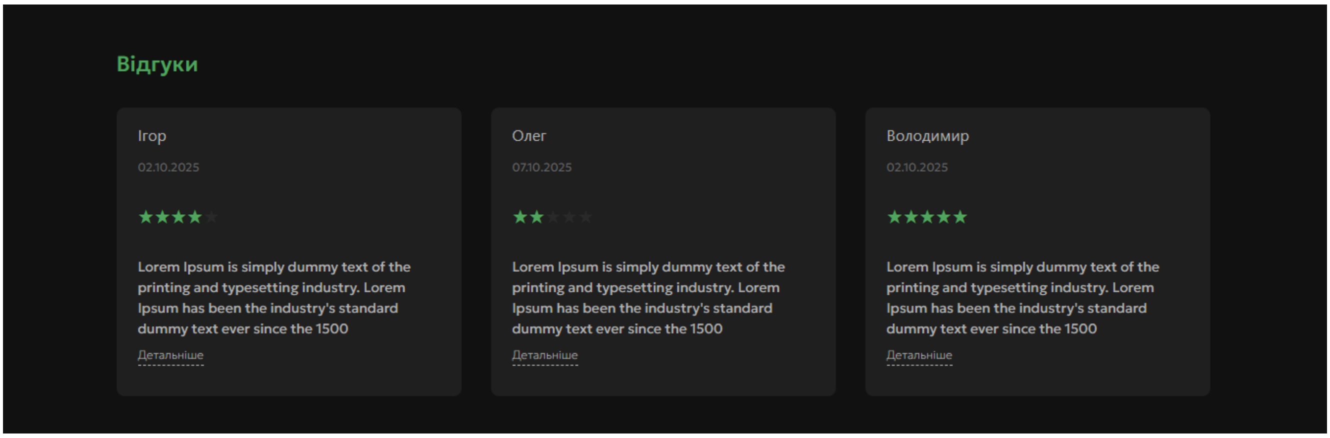


Рисунок В.2 – Секція відгуків клієнтів

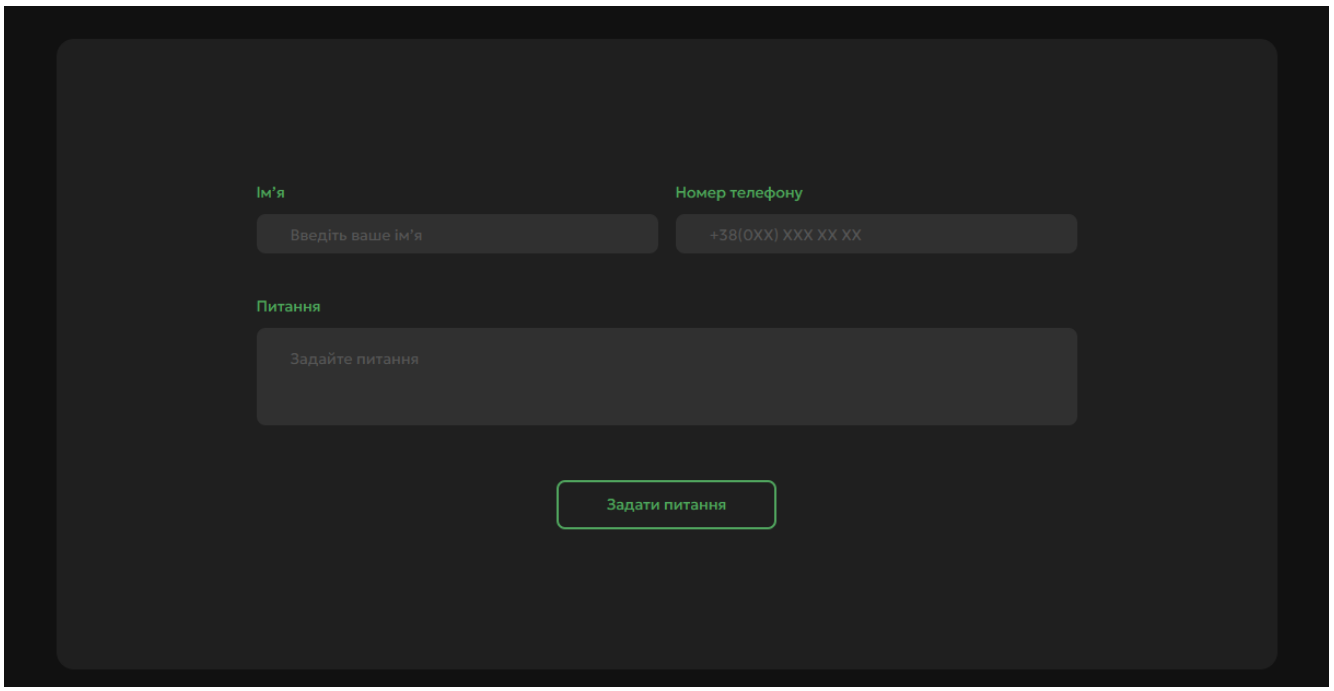


Рисунок В.3 – Секція контактної форми

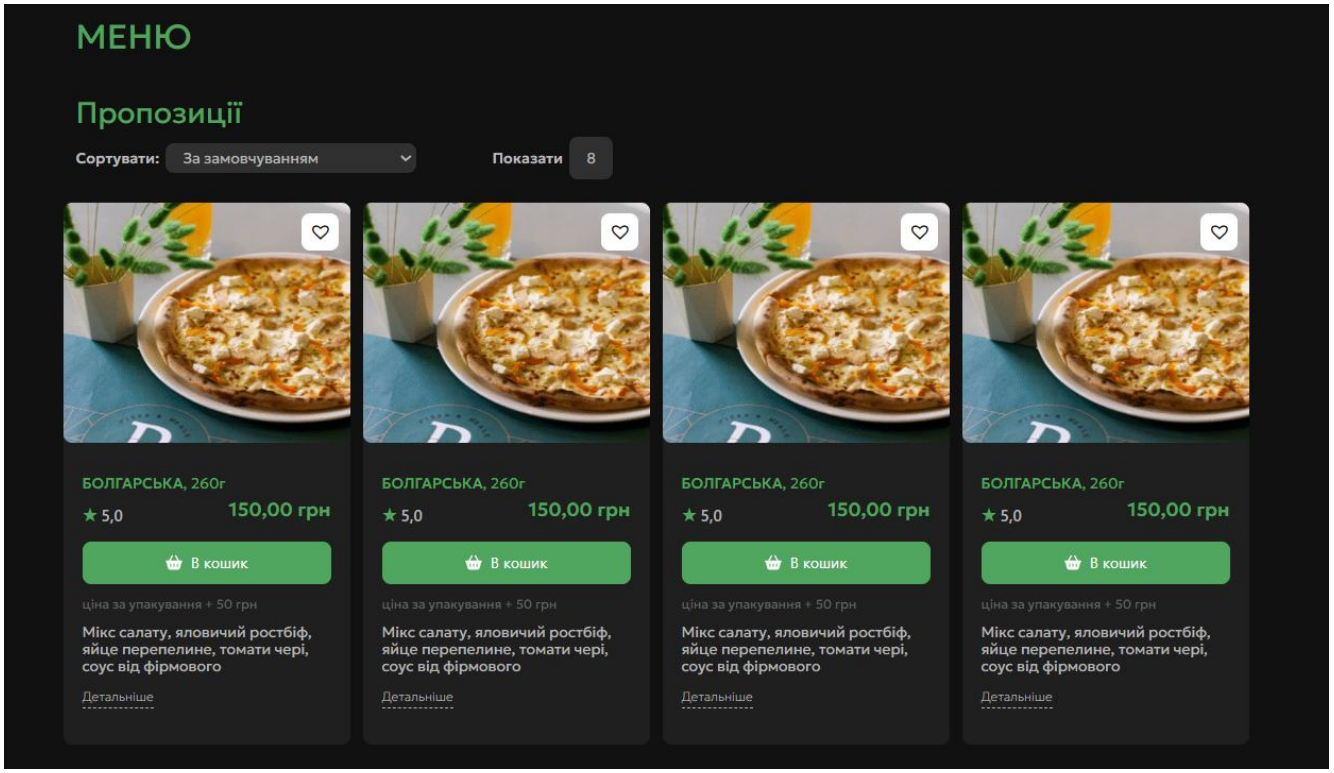


Рисунок В.4 – Сторінка меню

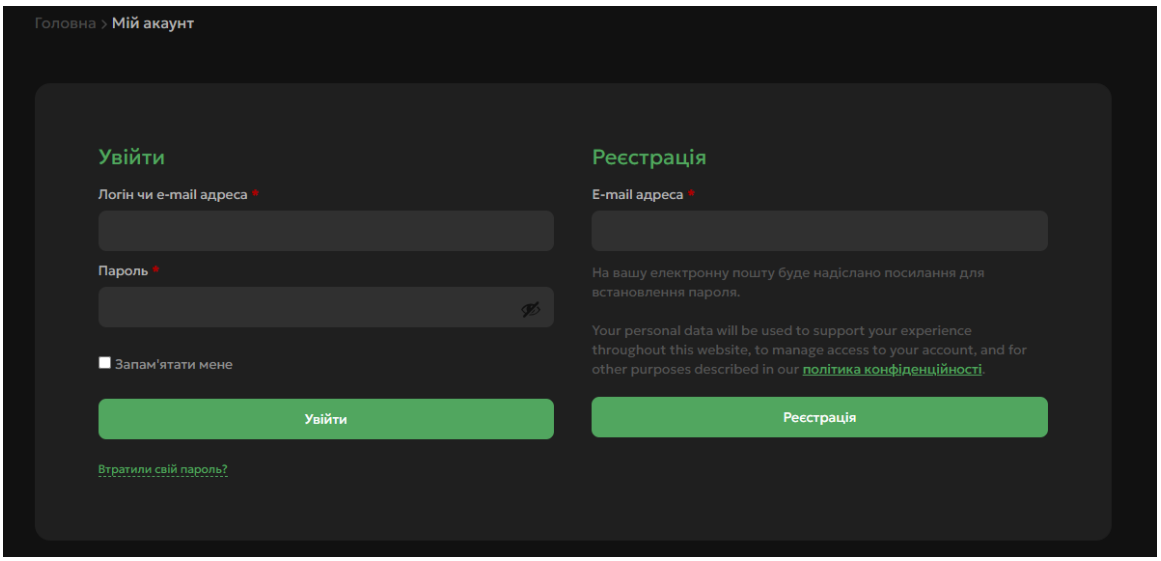


Рисунок В.5 – Сторінка входу в кабінет

Рисунок В.6 – Вкладка замовлень в кабінеті

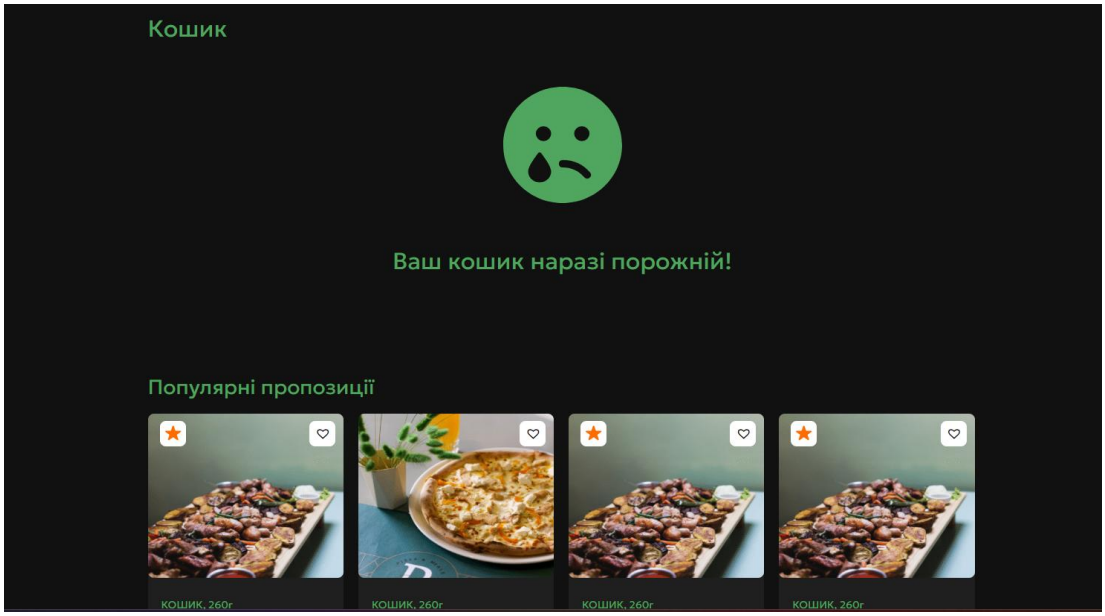


Рисунок В.7 – Вигляд пустого кошика

ДОДАТОК В

Лістинг коду

Лістинг коду шаблону single-product.php:

```
<?php
// Захист від прямого доступу та підключення хедера
defined('ABSPATH') || exit;
get_header('shop'); ?>

<section class="single-product">
  <div class="container">
    <?php do_action('woocommerce_before_main_content'); ?>

    <?php while (have_posts()): the_post(); ?>
      <div id="product-<?php the_ID(); ?>" <?php
wc_product_class(); ?>>
        <div class="custom-product-summary">

          <div class="col-md-5">
            <?php
            $main_image_id = $product->get_image_id();
            $all_image_ids = $product->get_gallery_image_ids();
            if ($main_image_id) array_unshift($all_image_ids,
$main_image_id);

            if (!empty($all_image_ids)) : ?>
              <div class="swiper-container product-main-slider
text-center">
                <div class="swiper-wrapper">
                  <?php foreach ($all_image_ids as $image_id) {
                    echo '<div class="swiper-slide">';
                    echo wp_get_attachment_image($image_id,
'full', false, ['class' => 'img-fluid']);
                    echo '</div>';
                  } ?>
                </div>
              </div>

              <?php if (count($all_image_ids) > 1) : ?>
                <div class="swiper-container product-thumbs-
slider">
                  <div class="swiper-wrapper">
                    <?php foreach ($all_image_ids as $image_id)
{
                      echo '<div class="swiper-slide">';
                      echo wp_get_attachment_image($image_id,
'thumbnail', false, ['class' => 'product-img']);
                      echo '</div>';
                    } ?>
                  </div>
```

```
                    </div>
                  <?php endif; ?>
                <?php else : ?>
                  <img src="<?php echo wc_placeholder_img_src(); ?>"
class="img-fluid" alt="Placeholder">
                <?php endif; ?>
              </div>

              <div class="product-info">
                <?php
                do_action('custom_single_product_title');
                do_action('custom_single_product_weight');
                display_discounted_price_block();
                ?>

                <div class="rating-price">
                  <?php do_action('custom_single_product_price'); ?>
                  <?php do_action('custom_single_product_rating');
?>
                </div>

                <?php $allergens = get_field('allergens');
                if (!empty($allergens)) : ?>
                  <div class="product-allergens">
                    <label class="product-allergens-
toggle">Алергени</label>
                    <div class="product-allergens-content"
style="display: none;">
                      <ul>
                        <?php foreach (explode(',', $allergens) as
$allergen) : ?>
                          <li><?php echo trim($allergen); ?></li>
                        <?php endforeach; ?>
                      </ul>
                    </div>
                  </div>
                <?php endif; ?>

                <?php
do_action('custom_single_product_add_to_cart'); ?>
              </div>
            </div>
          </div>
        <?php endwhile; ?>
      </div>
    </section>

    <section class="add-to-order-single">
      <div class="container">
        <?php $additional_products = get_additional_products();
        if (!empty($additional_products)) : ?>
          <h2>Додати до замовлення:</h2>
          <div class="swiper-container swiper-add-to-order-single">
```

```
            <div class="swiper-wrapper">
              <?php foreach ($additional_products as $add_prod) : ?>
                <div class="swiper-slide product-item">
                  <div class="product-item-img"><?php echo
$add_prod->get_image(); ?></div>
                  <div class="product-item-info">
                    <p class="product-item-title">
                      <?php echo $add_prod->get_name(); ?>
                      <?php if ($add_prod->get_weight()) echo
'<span>' . $add_prod->get_weight() . ' г</span>'; ?>
                    </p>
                    <div class="product-item-price">+<?php echo
$add_prod->get_price_html(); ?></div>

                    <div class="product-item-add-to-cart">
                      <?php
                      $product_id = $add_prod->get_id();
                      $in_cart = false;
                      foreach (WC()->cart->get_cart() as $cart_item)
{
                        if ($cart_item['data']->get_id() ==
$product_id) {
                          $in_cart = true; break;
                        }
                      }
                      if ($in_cart) : ?>
                        <a href="<?php echo
esc_url(wc_get_cart_url()); ?>" class="view-cart-button">Дивитися
кошик</a>
                      <?php else : ?>
                        <a href="?add-to-cart=<?php echo
$product_id; ?>" class="add-to-cart-button">В Корзину</a>
                      <?php endif; ?>
                    </div>
                  </div>
                </div>
              <?php endforeach; ?>
            </div>
          </div>
        <?php endif; ?>
      </div>
    </section>

    <section class="reviews-single">
      <div class="container">
        <?php
        $reviews = get_comments(['post_id' => get_the_ID(), 'status'
=> 'approve', 'number' => 5]);
        if ($reviews) : ?>
          <div class="client-comments-wrapper">
            <h3>Відгуки</h3>
            <div class="swiper-wrapper">
              <?php foreach ($reviews as $review) : ?>
```

```
                <div class="swiper-slide comment-client">
                  <p class="client-name"><?php echo $review-
>comment_author; ?></p>
                  <p class="date"><?php echo
get_comment_date('m.d.Y', $review->comment_ID); ?></p>

                  <div class="stars">
                    <?php
                    $rating = intval(get_comment_meta($review-
>comment_ID, 'rating', true));
                    for ($i = 1; $i <= 5; $i++) {
                      echo ($i <= $rating) ? '<span class="star
full">&#9733;</span>' : '<span class="star empty">&#9733;</span>';
                    }
                    ?>
                  </div>
                  <p class="comment"><?php echo $review-
>comment_content; ?></p>
                </div>
              <?php endforeach; ?>
            </div>
          </div>
        <?php endif; ?>

        <form id="custom-review-form" method="post">
          <h3>Залишити відгук</h3>
          <div class="rating">
            <label>Ваша оцінка</label>
            <div id="rating-stars">
              <?php for ($i = 1; $i <= 5; $i++) echo '<span
class="star" data-rating="'.$i.'">&#9733;</span>'; ?>
            </div>
            <input type="hidden" name="rating" id="custom-rating"
required>
          </div>

          <input type="text" name="custom_name" placeholder="Введіть
ваше ім'я" required>
          <input type="tel" name="custom_phone"
placeholder="+38(0XX) XXX XX XX" required>
          <textarea name="custom_comment" placeholder="Введіть ваш
коментар"></textarea>

          <input type="hidden" name="product_id" value="<?php echo
get_the_ID(); ?>">
          <input type="submit" name="submit_custom_review"
value="Надіслати відгук">
        </form>
      </div>
    </section>

    <section class="similar-section">
      <div class="container">
```

```
        <?php
        $category_ids =
wp_list_pluck(wp_get_post_terms(get_the_ID(), 'product_cat'),
'term_id');

        if (!empty($category_ids)) :
          $similar_products = new WP_Query([
            'post_type'      => 'product',
            'posts_per_page' => 5,
            'orderby'        => 'rand',
            'tax_query'      => [['taxonomy' => 'product_cat',
'field' => 'id', 'terms' => $category_ids, 'operator' => 'IN']]
          ]);

          if ($similar_products->have_posts()) : ?>
            <h3>Схожі пропозиції</h3>
            <div class="swiper-container swiper-similar">
              <div class="swiper-wrapper">
                <?php while ($similar_products->have_posts()) :
$similar_products->the_post(); ?>
                  <div class="swiper-slide">
                    <?php wc_get_template_part('content',
'product'); ?>
                  </div>
                <?php endwhile; wp_reset_postdata(); ?>
              </div>
            </div>
          <?php endif;
        endif; ?>
      </div>
    </section>

    <?php get_footer('shop'); ?>
```

ДОДАТОК Д

Посилання на репозиторій проєкту

https://github.com/dovban46/dovban46-diplom-project-2026.git