МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Соловйов Данило Вадимович
(прізвище, ім'я, по батькові здобувача)

на тему **Розробка програмного забезпечення продажу фаст-фуду з можливістю доставки**

(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД

(повна назва бази дослідження)

науковий керівник **к.т.н., доцент** _____ **Хоцкіна В.Б.**
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 № 12
Завідувач кафедри _____
(підпис)
д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
«11» червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи Розробка програмного забезпечення продажу фаст-фуду з можливістю доставки

Керівник роботи _к.т.н., доцент_ Хоцкіна В.Б.
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Проєктування web-сайту

Розділ 3. Проєктування бази даних web-сайту

Розділ 4. Розробка web-сайту та додатку

Об'єкт дослідження: програмне забезпечення, орієнтоване на продаж фаст-фуду з можливістю онлайн-оформлення замовлення та інтерактивної побудови страв.
Предмет дослідження: структура, логіка та інтерфейс веб-застосунку й десктопного конструктора піци, а також способи візуалізації й подання інформації, що сприяють швидкому вибору товарів і зручному оформленню замовлень користувачами
Мета кваліфікаційної роботи: створити комплексну програмну систему, яка дозволяє користувачам переглядати меню, формувати індивідуальні замовлення фаст-фуду, миттєво оцінювати вартість, обирати інгредієнти у візуальному режимі та передавати дані для доставки.

5. Дата видачі завдання «04» квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ Хоцкіна В.Б.

(підпис) (прізвище та ініціали)

Завдання одержав _____ Соловйов Д.В.

(підпис) (прізвище та ініціали)

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про перевірку
кваліфікаційної роботи на прояви академічного плагіату
та розміщення в Репозитарії Університету

Я, Соловйов Данило Вадимович, підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота Розробка програмного забезпечення продажу фаст-фуду з можливістю доставки виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

Дата підпис ініціали, прізвище (власноруч)

АНОТАЦІЯ

на бакалаврську кваліфікаційну роботу

« Розробка програмного забезпечення продажу фаст-фуду з можливістю доставки»

Соловйов Данило Вадимович

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У дипломній роботі розроблено два програмні продукти: веб-сайт для онлайн-продажу фаст-фуду з доставкою та десктопний застосунок-конструктор страв. Веб-сайт реалізовано з використанням мов PHP, JavaScript, HTML, CSS та СУБД MySQL. Реалізовано функціонал реєстрації користувачів, перегляду товарів, фільтрації, додавання до кошика, оформлення замовлень, пошуку та оцінювання продукції. Сайт має адаптивний дизайн і підтримує асинхронну взаємодію з сервером через AJAX.

Десктопний застосунок створено на C# з використанням Windows Forms та бібліотеки OpenGL. Користувач має змогу самостійно сформувати піцу або бургер: вибрати соус, додати інгредієнти, бачити результат у графічному середовищі. Автоматично обчислюється вартість замовлення. Після формування замовлення користувач вводить контактні дані для подальшої обробки.

Ключові слова: ВЕБ-САЙТ, PHP, MYSQL, JAVASCRIPT, AJAX, WINDOWS FORMS, OPENGL, КОНСТРУКТОР ПІЦИ.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД (база даних)	Впорядкований набір логічно взаємопов’язаних даних, що використовується для зберігання інформації про користувачів, замовлення, товари тощо.
СУБД	Система управління базами даних, яка забезпечує доступ, збереження та обробку інформації.
ПЗ	Програмне забезпечення — сукупність програм, що реалізують функціональність інформаційної системи.
AJAX	Технологія асинхронного обміну даними між веб-браузером і сервером без перезавантаження сторінки.
MySQL	Реляційна система управління базами даних з відкритим кодом, що використовується у проєкті для зберігання інформації.
WinForms	Фреймворк від Microsoft для створення графічного інтерфейсу десктопних застосунків на C#.
OpenGL	Бібліотека для рендерингу дво- та тривимірної графіки, що використовується для візуалізації інгредієнтів у конструкторі піци.
HTML	Мова розмітки гіпертексту, яка використовується для структурування веб-сторінок.
PHP	Мова програмування, що використовується на стороні сервера для обробки запитів, взаємодії з базою даних.
JavaScript	Мова програмування для реалізації динамічної поведінки елементів інтерфейсу веб-сайту.
UX/UI	Користувацький досвід (User Experience) / Інтерфейс користувача (User Interface) — поняття, що охоплюють зручність, логіку та зовнішній вигляд взаємодії з системою.

ЗМІСТ

ВСТУП 8
РОЗДІЛ І ПОСТАНОВКА ЗАДАЧІ 10
1.1. Характеристика задачі 10
1.2. Огляд існуючих web-сайтів 13
1.3. Аналіз вимог до веб-сайту 19
1.4. Аналіз вимог до десктопного застосунку 22
РОЗДІЛ ІІ ПРОЄКТУВАННЯ ВЕБ-САЙТУ 25
2.1. Проєктування веб-сайту 25
2.2. Проєктування десктопного застосунку 29
РОЗДІЛ ІІІ ПРОЄКТУВАННЯ БАЗИ ДАНИХ ВЕБ-САЙТУ 33
3.1. Обґрунтування вибору СУБД 33
3.2. Структура таблиць бази даних 35
3.3. Реляційний зв’язок між таблицями 41
РОЗДІЛ IV РОЗРОБКА ВЕБ-САЙТУ ТА ДОДАТКУ 46
4.1. Обґрунтування вибору технологій розробки 46
4.2. Інтерфейс користувача сайту 48
4.3. Десктопний застосунок на C# 54
ВИСНОВКИ 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60

ВСТУП

У наш час інформаційні технології стрімко проникають у всі сфери життя, і галузь торгівлі, зокрема громадського харчування, не є винятком. Зростання популярності онлайн-сервісів доставки їжі зумовлене потребою споживачів у швидкому, зручному та безконтактному способі замовлення продукції. Особливої актуальності набуває розробка ефективних веб-рішень для закладів швидкого харчування, які дозволяють клієнтам не лише переглядати меню, але й формувати замовлення відповідно до власних вподобань, вказувати адресу доставки, обирати спосіб оплати, а також залишати відгуки. У таких системах важливу роль відіграє інтерфейс, функціональність та взаємодія з базою даних.

Сучасні тенденції веб-розробки вимагають поєднання гнучких серверних технологій, зручного інтерфейсу користувача, а також інтеграції з адміністративною частиною для управління замовленнями та товарами. Крім того, дедалі частіше виникає потреба у розробці супутніх десктопних додатків, які дозволяють на локальному рівні або у візуальній формі працювати з даними замовлення, створюючи унікальний досвід для клієнта. Наприклад, можливість у графічному режимі "зібрати" власну піцу чи бургер за допомогою конструктора, що реалізований на основі графічного рушія OpenGL у середовищі C#, дає змогу не лише бачити кінцевий результат, а й автоматично обчислювати ціну, змінювати соуси, інгредієнти, розмір тощо.

Метою даної дипломної роботи є розробка повнофункціонального веб-сайту та супутнього десктопного застосунку для автоматизації процесу замовлення фаст-фуду з доставкою. Система повинна забезпечувати зручну взаємодію користувача з інтерфейсом, підтримувати реєстрацію та вхід, перегляд товарів за категоріями, додавання до кошика, оформлення замовлення, відстеження статусу доставки, залишення відгуків, а також адміністрування замовлень і товарів через окрему панель управління. Додатково має бути реалізований десктопний додаток, який дозволяє візуально зібрати замовлення,

додати інгредієнти, змінити соус, розмір, та отримати фінальний вигляд і вартість замовленої страви.

Для досягнення поставленої мети у роботі необхідно виконати низку послідовних завдань. Насамперед потрібно проаналізувати існуючі онлайн-сервіси доставки їжі та виявити їх сильні й слабкі сторони. На основі цього сформулювати вимоги до майбутньої системи, як з боку користувача, так і адміністратора. Далі необхідно розробити логіку сайту, визначити архітектуру клієнтської і серверної частини, створити реляційну базу даних з відповідними таблицями та зв’язками між ними, реалізувати засоби обробки запитів та збереження інформації. Особливу увагу слід приділити дизайну інтерфейсу — він має бути інтуїтивно зрозумілим, адаптивним до різних пристроїв і привабливим візуально. У завершальному етапі потрібно створити десктопний застосунок на C# з використанням OpenGL, у якому користувач зможе наочно додавати інгредієнти до піци або бургера й бачити результат у реальному часі.

Об’єктом дослідження є процес автоматизації прийому та обробки замовлень на фаст-фуд із використанням веб- та десктопного інтерфейсу. Предметом дослідження виступають технології розробки веб-сайтів і настільних додатків, а також підходи до проєктування інтерактивних інтерфейсів, роботи з базами даних і реалізації бізнес-логіки в програмному забезпеченні. Робота охоплює використання таких інструментів, як PHP, MySQL, JavaScript, HTML, CSS для створення веб-сайту та C# з OpenGL — для розробки десктопного графічного додатку.

РОЗДІЛ І
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Із розвитком цифрових технологій відбувається трансформація способу ведення бізнесу в багатьох галузях. Особливо це помітно у сфері роздрібної торгівлі та громадського харчування, де онлайн-продажі та послуги доставки стали невід'ємною частиною повсякденного життя. У постпандемічному світі роль дистанційного замовлення їжі зросла в рази, і підприємства, які не мають власного веб-сайту або мобільного застосунку, втрачають велику частину потенційних клієнтів.

У цьому контексті постає необхідність у створенні інформаційної системи, яка дозволить не лише переглядати меню, але й забезпечить повноцінний цикл взаємодії з клієнтом: реєстрація, вибір продукції, налаштування параметрів замовлення, оформлення покупки, вибір способу оплати, відстеження статусу доставки, а також надання зворотного зв’язку у вигляді оцінок і відгуків. Така система повинна бути простою, інтуїтивно зрозумілою для кінцевого користувача, водночас багатофункціональною та технічно надійною.

Постановка задачі полягає у розробці веб-сайту з клієнтською та адміністративною частинами, а також десктопного додатку, який дозволяє зібрати страву візуально, як у справжньому кулінарному конструкторі. Рішення повинне задовольняти потреби не лише кінцевого користувача, а й власника закладу, надаючи йому інструменти для керування товарами, перегляду та обробки замовлень, аналітики продажів тощо. Особливу увагу слід приділити інтерфейсу користувача, оскільки зручність використання сайту має прямий вплив на рівень продажів та лояльність клієнтів.

Розробка подібної системи вимагає чітко визначеної структури. Основна задача — реалізувати фронтенд частину сайту з адаптивною верстkою, яка дозволяє зручно взаємодіяти із сервісом як з комп’ютера, так і з мобільного

пристрою. Інтерфейс має включати категорії товарів, картки продуктів із ціною, кнопками «Додати в кошик», перегляд повного опису товару, а також сторінку оформлення замовлення з обов’язковими етапами: введення контактної інформації, вибір адреси доставки, вибір способу оплати, підтвердження замовлення.

Серверна частина задачі передбачає обробку запитів, взаємодію з базою даних, реєстрацію користувачів, авторизацію, збереження та оновлення замовлень, роботу з відгуками та рейтингами, обробку дій адміністратора. Необхідно передбачити логіку обробки помилок, валідацію даних на бекенді, а також реалізацію системи повідомлень або статусів, які інформують користувача про хід виконання його замовлення.

Одним із найбільш інноваційних елементів поставленої задачі є розробка десктопного додатку, який працює як інтерактивний візуальний конструктор страви. Це не просто візуалізація замовлення, а повноцінна графічна система, яка дозволяє в режимі реального часу вибирати соус (наприклад, білий або червоний), додавати інгредієнти (гриби, помідори, бекон, оливки тощо), змінювати розмір піци або тип булки для бургера. Кожна дія користувача візуально відображається на екрані, що створює ефект гейміфікації та підвищує залученість користувача. Застосунок має також розраховувати загальну вартість замовлення залежно від обраних компонентів і формувати структуру для подальшої обробки на стороні сайту.

Технічно реалізація задачі передбачає використання веб-технологій, таких як PHP для серверної логіки, MySQL для створення та підтримки реляційної бази даних, JavaScript (зокрема, з AJAX-запитами) для динамічної взаємодії з інтерфейсом, а також HTML і CSS для створення адаптивного та привабливого візуального оформлення сторінок. Адміністративна панель повинна дозволяти додавати нові товари, змінювати їхні параметри, видаляти з каталогу, переглядати список замовлень, змінювати їх статус (нове, у роботі, доставлено), керувати відгуками та рейтингами.

Складність задачі також полягає в забезпеченні стійкої та безпечної роботи всієї системи. Потрібно врахувати можливість одночасної взаємодії великої кількості користувачів, уникати помилок при обробці даних, захищати персональні дані користувачів, паролі, а також захистити адміністративний інтерфейс від несанкціонованого доступу. Одним із пріоритетів є шифрування паролів користувачів (за допомогою bcrypt або password_hash у PHP), використання захищених сесій, обмеження SQL-запитів до підготовлених (prepared statements), захист форм від CSRF і XSS-атак.

Крім технічної реалізації, задача має також бізнесову складову. Створення такої системи дозволяє оптимізувати витрати на персонал, скоротити кількість помилок при оформленні замовлень, підвищити задоволення клієнтів, сформувати лояльну аудиторію та збільшити прибуток закладу. Крім того, за допомогою збору статистики можна аналізувати популярні позиції меню, пікові години активності, середню тривалість виконання замовлення тощо, що відкриває можливості для подальшої аналітики та вдосконалення сервісу.

Варто також звернути увагу на можливість масштабування проєкту. Створена система повинна мати потенціал для подальшого розширення — наприклад, додавання мобільного додатку, інтеграції з платіжними системами або службами доставки, підтримки декількох мов, створення програми лояльності тощо.

Таким чином, задача дипломної роботи є комплексною, охоплює широкий спектр функцій і потребує використання знань з кількох областей інформаційних технологій: веб-програмування, розробки графічних застосунків, проєктування баз даних, забезпечення інформаційної безпеки, створення адаптивного дизайну та взаємодії між клієнтом і сервером. Успішне вирішення поставленої задачі дозволить створити сучасний, зручний та функціональний інструмент як для споживачів, так і для адміністрації закладу швидкого харчування.

Реалізація такої системи відповідає актуальним тенденціям цифрової трансформації, сприяє підвищенню конкурентоспроможності бізнесу та дозволяє швидко реагувати на зміни споживчих очікувань.

1.2. Огляд існуючих web-сайтів

Перед розробкою власного веб-сайту для продажу фаст-фуду з доставкою важливо провести детальний аналіз уже існуючих рішень, які успішно працюють на ринку. Це дозволяє вивчити як позитивні аспекти реалізації, так і виявити типові недоліки, яких варто уникати під час власної розробки. Крім того, огляд конкурентів допомагає краще зрозуміти очікування користувачів, сформувати уявлення про сучасні тенденції в дизайні та функціональності таких систем, а також надихнутись до створення якісного, унікального та зручного продукту.

У цьому підрозділі буде розглянуто три популярні веб-сайти, тематика яких безпосередньо пов’язана з доставкою страв швидкого харчування. Кожен із них буде проаналізовано за кількома основними критеріями: структура головної сторінки, каталог продукції, сторінка окремого товару, кошик, форма оформлення замовлення, функції авторизації, дизайн, адаптивність та зручність інтерфейсу. Це дозволить оцінити рівень реалізації подібних сервісів та виявити ті рішення, які можуть бути використані або покращені у власному проєкті.

Після короткої загальної характеристики кожного сайту буде проведений поетапний аналіз усіх ключових сторінок та елементів, з прикладами, скриншотами і детальними висновками.

Одним із прикладів українських сервісів, які пропонують доставку готової або напівготової їжі, є сайт «ГУДФУД ВДОМА». Він спеціалізується на продажу свіжоморожених страв з можливістю онлайн-замовлення та доставки. На головній сторінці ресурсу представлено ключові категорії товарів, акційні пропозиції та пошуковий блок, що дозволяє швидко орієнтуватися в асортименті (Рис. 1.1.).

Основне меню розміщено у верхній частині сайту та включає навігацію за розділами, перемикач мови, вхід у профіль і доступ до кошика. Центральне місце займає банер-слайдер із сезонними пропозиціями. Нижче подано добірки страв у вигляді тематичних блоків: «Новинки», «Бестселери», «Акції» тощо. Візуально сайт виглядає сучасно, яскраво і зручно — особливо завдяки великим

зображенням та чітко структурованому інтерфейсу. Водночас відчувається акцент на емоційному залученні користувача через візуалізацію готових страв.

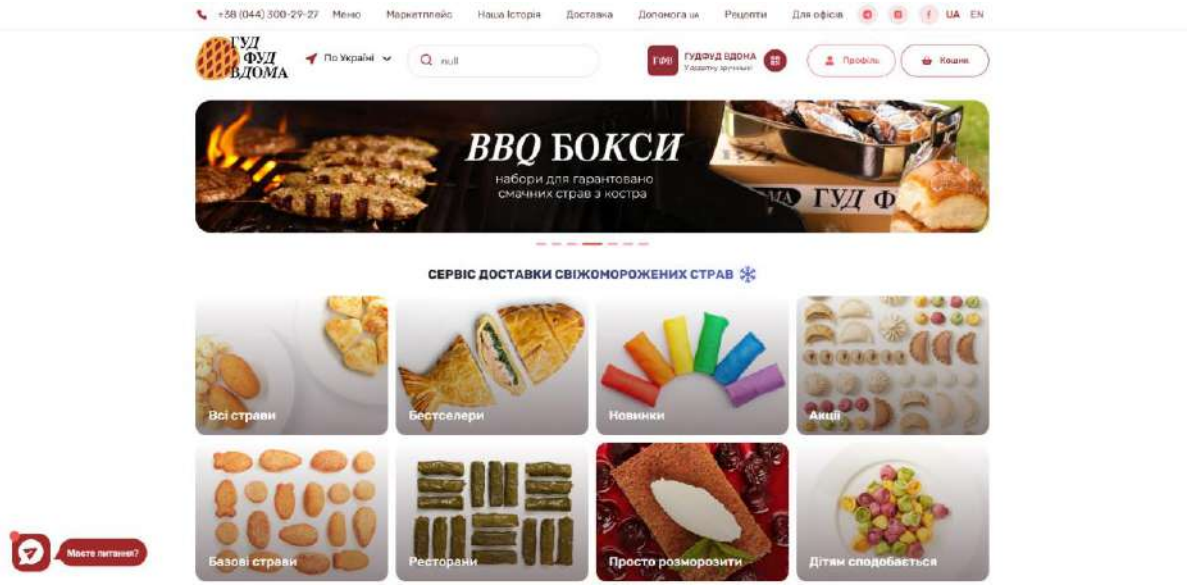

Рис. 1.1. Головна сторінка сайту «ГУДФУД ВДОМА»

Окрема картка товару на сайті «ГУДФУД ВДОМА» оформлена за принципами візуальної привабливості та інформативності. Ліворуч розташовані якісні фотографії страви — пирога з лососем та шпинатом, які демонструють як зовнішній вигляд, так і розріз готової продукції. Праворуч подано назву товару, вагу, ціну за 700 грамів, а також кнопку «До кошика», яка спрощує процес оформлення покупки. Нижче подано два блоки: «Склад» — з детальним описом інгредієнтів та харчової цінності, і «Рецепт» — з покроковою інструкцією приготування (Рис. 1.2.).

Такий підхід дозволяє користувачу швидко оцінити як вигляд, так і вміст страви, дізнатися про її склад, умови зберігання та спосіб приготування. Сторінка виглядає збалансовано, логічно структурована та відповідає очікуванням сучасного споживача.

Крім функціонального наповнення, слід відзначити й вдалу візуальну реалізацію. Уся структура поділена на окремі інформаційні блоки з чіткою типографікою й відступами, що полегшує сприйняття. Червоні акценти в області складу й кнопки привертають увагу до ключових елементів. Завдяки цьому сторінка не виглядає перевантаженою, попри доволі великий обсяг контенту.

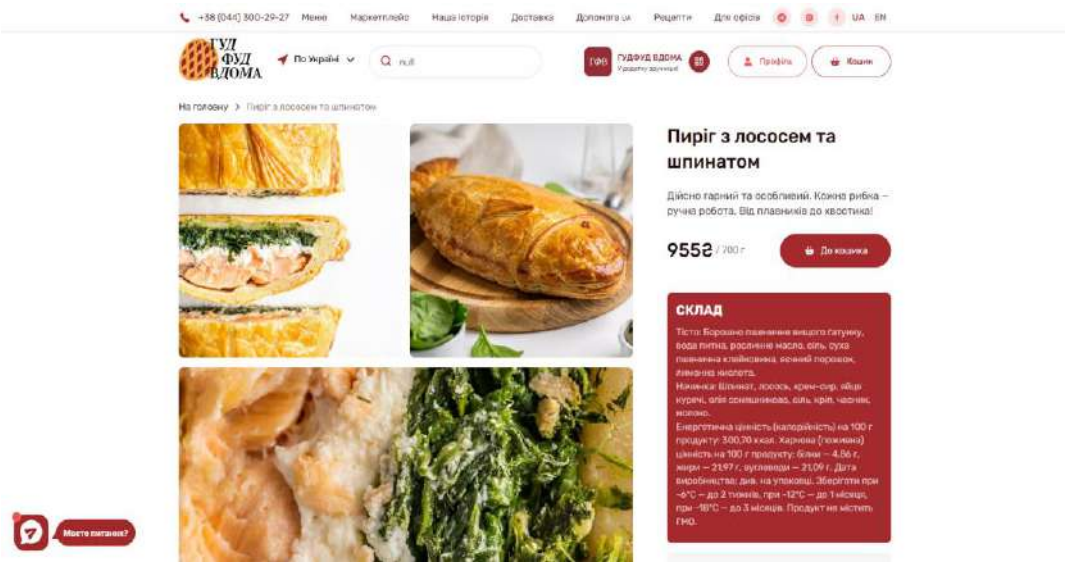

Рис. 1.2. Сторінка картки товару на сайті «ГУДФУД ВДОМА»

Кошик на сайті «ГУДФУД ВДОМА» реалізовано у вигляді бічної панелі, яка відкривається поверх поточної сторінки без перезавантаження. Це зручно, оскільки дозволяє швидко перевірити вміст замовлення та продовжити вибір страв без втрати контексту. У кошику відображається назва товару, його ціна, вага та кількість, яку можна змінити за допомогою кнопок «+» та «–». Також є можливість видалити позицію або очистити кошик повністю (Рис. 1.3.).

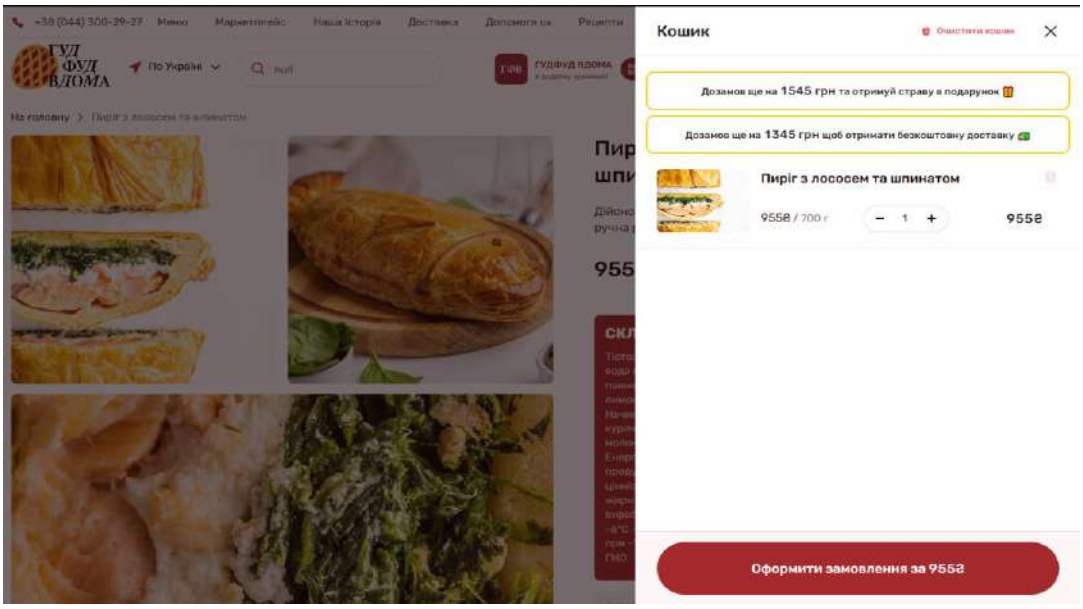

Рис. 1.3. Вікно кошика на сайті «ГУДФУД ВДОМА»

Наступним прикладом є сайт мережі ресторанів «Два Гуся», що пропонує доставку широкого асортименту готових страв: піци, бургерів, супів, салатів, десертів та іншого. Головна сторінка оформлена в активному червоному кольорі, що створює впізнаваний фірмовий стиль бренду. У верхній частині розташовано службову інформацію — контактний номер телефону, графік роботи, умови доставки, а також доступ до особистого кабінету та кошика. Центральну частину займає слайдер із кількома банерами, які змінюються автоматично. У банерах подаються акційні пропозиції, спеціальні набори або сезонні страви, що виконує функцію візуального заохочення користувача до взаємодії (Рис. 1.4.).

Нижче розміщено горизонтальне меню категорій з піктограмами. Проте, попри українську версію сайту, самі назви категорій — наприклад, «Пицца», «Гарниры», «Десерты» — залишаються російською мовою. Така мовна непослідовність може створити враження неуважності до деталей та викликати когнітивний дискомфорт у користувача, що обрав український інтерфейс. Виправлення цієї деталі покращило б загальне сприйняття ресурсу.

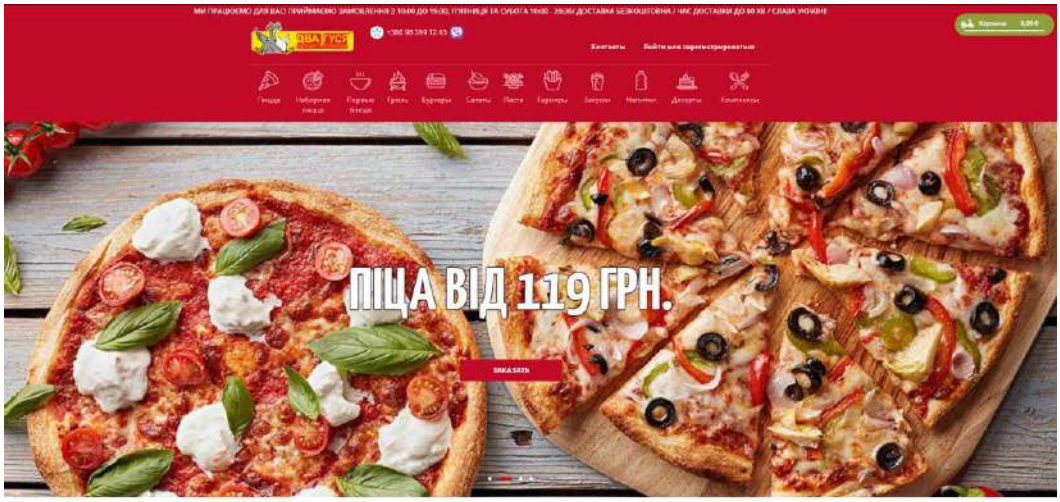

Рис. 1.4. Головна сторінка сайту «Два Гуся»

На сторінці товару сайт «Два Гуся» пропонує детальний огляд страви разом із можливістю індивідуалізації замовлення. Ліворуч розміщено зображення піци, праворуч — її назва, опис складу, вага, ціна та великий перелік додаткових інгредієнтів. Користувач має змогу самостійно обрати бажані добавки, встановивши потрібну кількість навпроти кожного компонента. Це

наближає інтерфейс до конструктора страв, що є сучасним і зручним підходом (Рис. 1.5.).

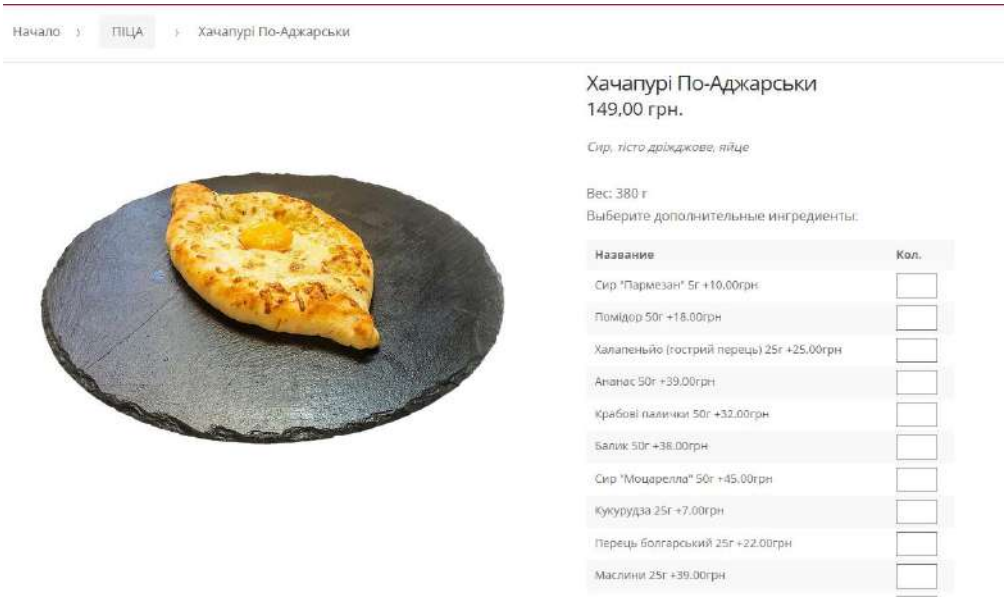

Рис. 1.5. Сторінка товару на сайті «Два Гуся»

Сторінка кошика на сайті «Два Гуся» оформлена лаконічно та функціонально. У центральній частині виводиться список замовлених позицій із назвою, зображенням, ціною та можливістю видалити товар. Нижче розміщено підсумкову вартість та блок входу або реєстрації користувача (Рис. 1.6.).

Інтерфейс у цілому зручний, без надлишкових елементів, але його завершеність могла б бути підсилена можливістю редагування кількості товарів або додавання коментарів до замовлення.

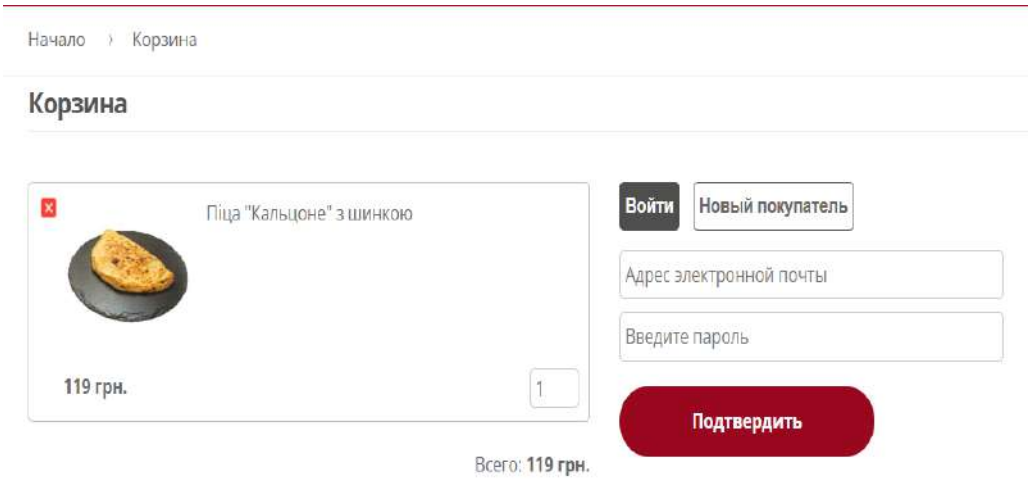

Рис. 1.6. Сторінка кошика на сайті «Два Гуся»

Третім прикладом є сайт компанії «Mr. Grill», орієнтований переважно на бізнес-клієнтів — постачальників їжі, точок продажу та закладів громадського харчування. Головна сторінка чітко демонструє цей фокус, містячи слогани на кшталт «почати бізнес», «адресна доставка по всій Україні» та «оптові ціни від виробника». У верхній частині розміщено фірмовий логотип, контакти, мовне меню, особистий кабінет і кошик. Навігація реалізована через випадаючий «Каталог» та горизонтальне меню зі спеціальними умовами співпраці.

Основна частина головної сторінки — це блок «Популярні категорії», який включає візуальні картки для різних товарів: сосиски для хот-догів, булки, готові бургери, донати тощо. Біля кожної картки вказано кількість доступних позицій у категорії та кнопка перегляду (Рис. 1.7.). Такий підхід дозволяє швидко зорієнтуватися в асортименті, не заглиблюючись у структуру сайту.

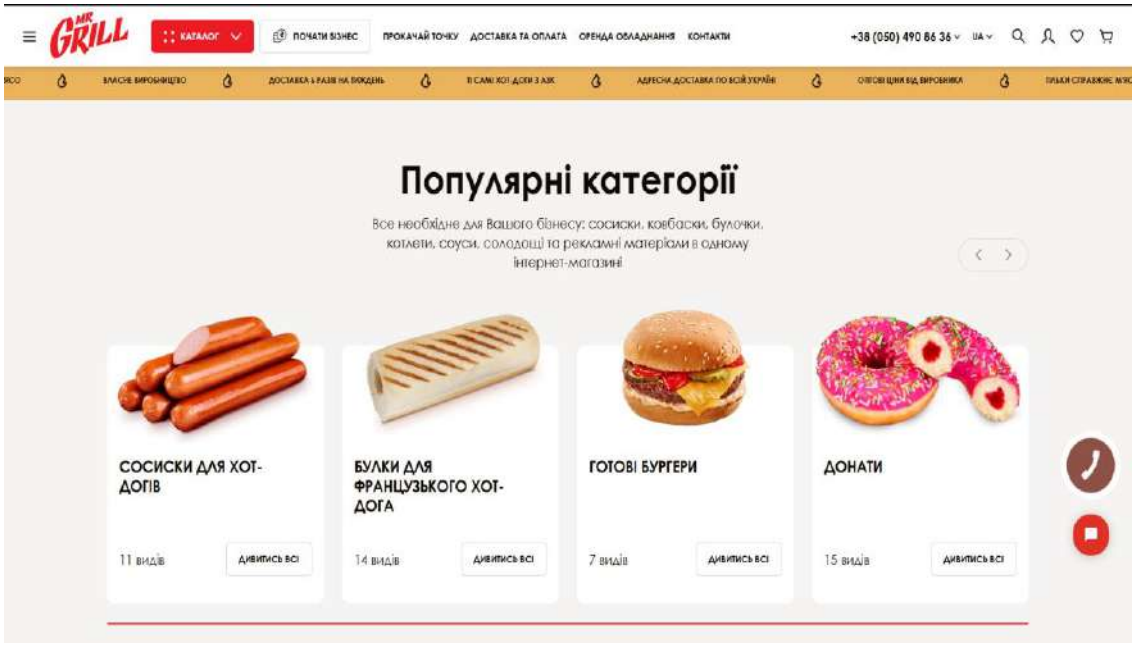

Рис. 1.7. Головна сторінка сайту «Mr. Grill»

Кошик на сайті «Mr. Grill» реалізовано у вигляді висувної правої панелі, яка з’являється поверх сторінки товару. Завдяки цьому користувач не переривається процес перегляду продукції. У кошику відображається зображення товару, його назва, вага, кількість одиниць у наборі та можливість змінювати кількість прямо в панелі. Нижче вказано підсумкову вартість замовлення та розміщено дві

кнопки: «Продовжити покупки» і «Оформити замовлення». Така логіка дозволяє оформлювати замовлення швидко й без зайвих переходів (Рис. 1.8.).

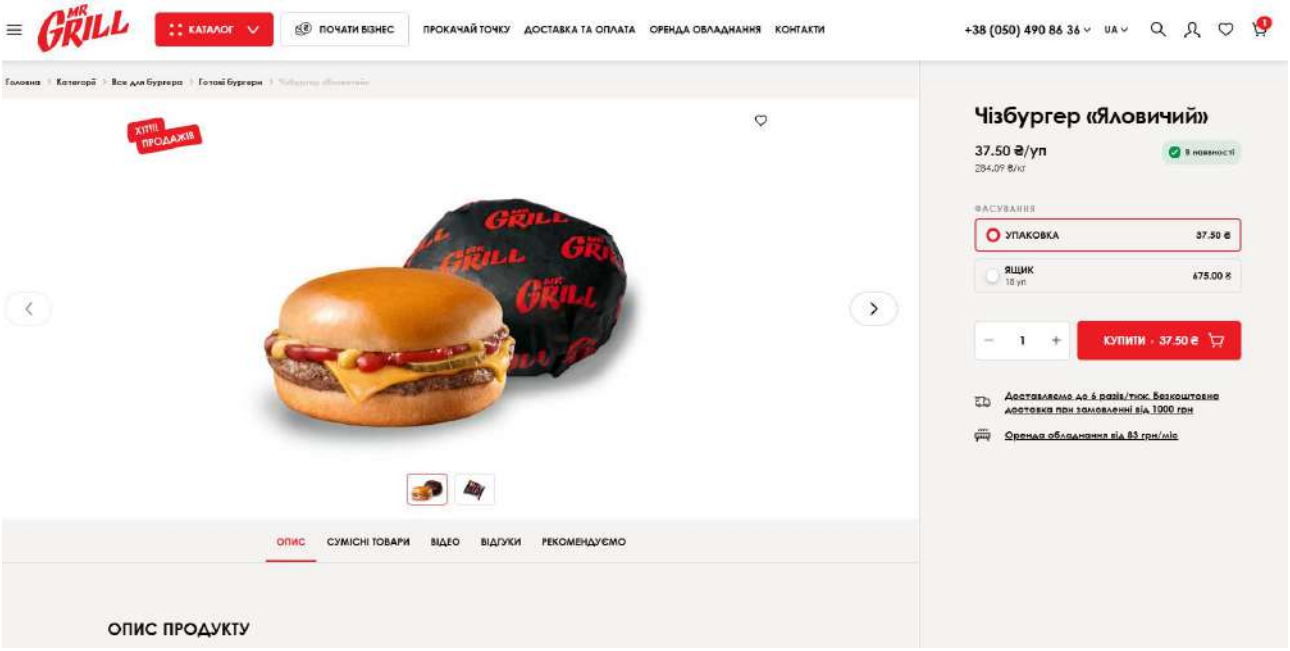

Рис. 1.8. Кошик на сайті «Mr. Grill»

Підсумовуючи, можна відзначити, що всі розглянуті сайти — «ГУДФУД ВДОМА», «Два Гуся» та «Mr. Grill» — мають загалом зручну та логічну структуру. Вони орієнтовані на різні аудиторії: від роздрібних користувачів до бізнес-клієнтів. Візуальні рішення та способи взаємодії з контентом мають як сильні сторони (простота, наочність, швидкість доступу до кошика), так і моменти, які можуть бути вдосконалені (локалізація, редагування кількості товарів, баланс між візуальним та текстовим наповненням). Отримані спостереження будуть враховані під час розробки власного веб-сайту.

1.3. Аналіз вимог до веб-сайту

Перед початком безпосередньої розробки веб-сайту необхідно чітко визначити вимоги до нього. Вимоги виступають основою для архітектурного проєктування, вибору технологій, реалізації інтерфейсу користувача та взаємодії з базою даних. Їх можна умовно поділити на функціональні, нефункціональні, технічні, інтерфейсні та безпекові.

Функціональні вимоги описують конкретні дії, які повинен виконувати сайт. Основна мета веб-ресурсу — надання клієнтам можливості перегляду меню, формування замовлення та його відправлення на обробку. Користувач повинен мати змогу реєструватися та входити до облікового запису, переглядати продукцію за категоріями, фільтрувати та шукати товари, переглядати детальну інформацію про кожен товар (назва, склад, ціна, маса, фото, відгуки). Система повинна забезпечувати додавання товарів у кошик, можливість змінювати кількість, очищати кошик і надсилати замовлення. Також важливо реалізувати можливість вибору способу доставки й оплати, залишення зворотного зв’язку у вигляді відгуку або оцінки.

Крім користувацького функціоналу, сайт повинен мати адміністративну частину. Адміністратор повинен мати змогу входити до окремого розділу сайту, де доступна робота з товарами (створення, редагування, видалення), обробка замовлень (зміна статусу, перегляд деталей), модерація відгуків, перегляд статистики продажів. Усі ці функції реалізуються з правами доступу, що обмежують звичайних користувачів.

Нефункціональні вимоги охоплюють характеристики системи, які не пов’язані з конкретними діями, але впливають на її якість. Веб-сайт повинен бути стабільним, витримувати навантаження щонайменше 100 одночасних користувачів без втрати продуктивності. Він має бути доступним 24/7, з мінімальним часом завантаження сторінок — до 2 секунд. Також передбачено адаптивність — можливість коректного відображення як на десктопах, так і на мобільних пристроях. Резервне копіювання бази даних повинно здійснюватися щонайменше раз на добу.

Сайт повинен бути масштабованим — тобто, у разі потреби, можливість розширення функціоналу (наприклад, додавання системи бонусів, push-повідомлень або інтеграції з кур’єрськими службами) не повинна вимагати повної переробки архітектури.

Технічні вимоги включають платформу розробки, технології та інструменти, які використовуються. Бекенд сайту реалізується мовою PHP, а в

якості СУБД використовується MySQL. Інтерфейс побудований з використанням HTML, CSS, JavaScript (включно з AJAX-запитами для динамічного оновлення контенту без перезавантаження сторінки). Сайт буде розміщено на Linux-сервері з підтримкою Apache або Nginx. Підтримка HTTPS-з'єднання є обов'язковою умовою для забезпечення захищеної передачі даних.

Код має бути структурованим і підтримуваним. Для зберігання файлів та стилів передбачається чітка система каталогів: окремі папки для шаблонів, зображень, скриптів, стилів та обробників серверної логіки. Всі запити до бази даних мають використовувати підготовлені конструкції (prepared statements), щоб виключити можливість SQL-ін'єкцій.

Інтерфейсні вимоги визначають загальний вигляд і поведінку елементів сайту. Важливими є простота, зрозумілість і логічність розміщення елементів. Кнопки повинні бути помітними, тексти — контрастними, поля введення — з підписами або підказками. При некоректному введенні даних (наприклад, при реєстрації або оформленні замовлення) система повинна повідомляти користувача про помилку без перезавантаження сторінки.

Кольорова гама сайту повинна бути сучасною, з чіткими акцентами: наприклад, яскраві кнопки заклику до дії (CTA), фонові кольори не повинні перевантажувати зір. Шрифти — читабельні, з оптимальним міжрядковим інтервалом. На мобільних пристроях інтерфейс повинен адаптуватися автоматично, забезпечуючи комфортну взаємодію користувача з контентом.

Безпекові вимоги мають важливе значення, оскільки сайт обробляє персональні дані користувачів (логіни, паролі, контактні дані, адреси доставки). Паролі мають зберігатися в зашифрованому вигляді (хешування через password_hash або bcrypt). Сесії користувачів повинні бути захищені від перехоплення або підміни (використання HTTPS, httpOnly-cookie, контроль IP або user-agent за потреби).

Також важливо передбачити обробку помилок: замість технічних повідомлень користувач повинен отримувати зрозумілі повідомлення з порадою

подальших дій. А всі внутрішні помилки повинні логуватись в окремий журнал, доступний лише адміністратору.

Загалом, сукупність зазначених вимог створює чітке технічне завдання для розробки ефективного, зручного та безпечного веб-сайту з продажу фаст-фуду.

1.4. Аналіз вимог до десктопного застосунку

Десктопний застосунок, який розробляється як частина загальної системи для продажу фаст-фуду, має специфічну функціональність. Його головне призначення — надати користувачу можливість створити візуальну конфігурацію замовлення, додавати або прибирати інгредієнти, змінювати розмір страви, обирати соус, а також бачити підсумкову вартість. Застосунок має також передбачати можливість збереження сформованого замовлення для подальшої обробки. Це не лише розширює можливості сайту, але й створює більш інтерактивний і цікавий досвід для кінцевого користувача.

Основна функція програми — створення інтерактивного конструктора, що відображає вибрані інгредієнти у вигляді шарів. Наприклад, користувач обирає соус, сир, пепероні, овочі — і кожен із цих елементів візуально додається на основу піци або всередину бургера. Візуалізація має відбуватись у реальному часі, з оновленням графічного відображення при кожній дії. Для цього буде використано мову програмування C# у середовищі Windows Forms з підключенням OpenGL для рендерингу графіки.

Програма має підтримувати базові дії: вибір розміру страви (наприклад, малий, середній, великий), тип соусу (білий або червоний), а також додавання або видалення інгредієнтів із доступного списку. Після кожної дії має оновлюватися не лише графіка, а й блок інформації про вартість та склад. Кожен інгредієнт повинен мати власну візуальну текстуру, що дозволяє створити умовно реалістичне зображення замовлення.

Окремо визначаються вимоги до зручності інтерфейсу. Він має бути простим і зрозумілим: ліворуч — панель інгредієнтів, праворуч — область

відображення страви, внизу — підсумкова інформація. Кнопки повинні бути підписані зрозуміло, без зайвих технічних термінів. Програма повинна запускатися без додаткового встановлення компонентів або залежностей, крім стандартного .NET Framework та бібліотек OpenGL.

Крім базового функціоналу, програмне забезпечення повинно розраховувати загальну вартість страви. Кожен інгредієнт має фіксовану ціну, яка додається до базової вартості залежно від розміру. У підсумковій області має відображатися не лише загальна сума, а й список обраних складників, що дозволяє користувачу перевірити замовлення перед підтвердженням.

Ще одним важливим аспектом є продуктивність графічного відображення. Рендеринг має відбуватись плавно, без затримок, навіть якщо обрано кілька інгредієнтів одночасно. Враховуючи, що мова йде про 2D-графіку без складних ефектів освітлення, цього можна досягти навіть на базових конфігураціях ПК. Проте графіка має залишатись приємною для користувача: необхідно уникати занадто простих або умовних зображень. Візуальні елементи повинні чітко асоціюватись з реальними інгредієнтами.

До вимог входить також можливість почистити страву (кнопка "Очистити"), почати створення нової конфігурації, змінити розмір без втрати доданих інгредієнтів, а також зберегти готове замовлення у вигляді текстового файла або передати його у веб-інтерфейс. Обмін інформацією з веб-сайтом можна реалізувати у вигляді структури JSON або простого переліку ключових параметрів (назва, інгредієнти, кількість, соус, розмір, загальна сума).

Застосунок повинен мати стабільну роботу, не зависати та не викликати помилок під час стандартних дій. Під час запуску користувач має одразу бачити чисте поле та обрати тип страви (піца або бургер). При кожному оновленні має зберігатись повна відповідність між візуальним відображенням та реальним складом. Програма не потребує підключення до мережі інтернет і може використовуватись у локальному режимі — наприклад, на окремому пристрої в точці продажу або на адміністративному комп’ютері.

Загалом, основними вимогами до десктопного застосунку є простота, наочність, стабільність, швидкість роботи та реалістичність візуального результату. Він повинен не просто дублювати функцію сайту, а розширювати її, додаючи інтерактивність і наочність процесу замовлення.

Висновки до розділу 1

У першому розділі було визначено загальну мету проєкту — створення програмного комплексу для замовлення страв швидкого приготування, що складається з веб-сайту та десктопного застосунку. Здійснено огляд існуючих рішень у цій сфері: «ГУДФУД ВДОМА», «Два Гуся» та «Mr. Grill». Проведений аналіз дозволив виявити їхні сильні сторони (зручна структура, візуально привабливий інтерфейс, наявність конструкторів і адаптивний кошик), а також недоліки (мовна несумісність, обмежена інтерактивність, недостатня деталізація при оформленні замовлення).

На основі порівняння розроблено перелік функціональних та нефункціональних вимог до власного веб-сайту, що включає підтримку реєстрації, фільтрації товарів, відгуків, динамічного кошика та адміністративного модуля. Усі вимоги сформульовано з урахуванням швидкодії, безпеки та зручності використання.

Окрему увагу приділено десктопному застосунку, який дозволяє користувачу самостійно зібрати конфігурацію страви за допомогою візуального конструктора. Зафіксовано ключові вимоги до цього компонента — інтерактивність, простий інтерфейс, відображення інгредієнтів у режимі реального часу, точний підрахунок вартості та локальна стабільність роботи.

Таким чином, у першому розділі було повністю окреслено вимоги до обох частин програмного комплексу — як з технічної, так і з користувацької точки зору. Це створює чітку основу для подальшого проєктування структури сайту, бази даних та реалізації графічних компонентів у десктопному середовищі.

РОЗДІЛ ІІ
ПРОЄКТУВАННЯ WEB-САЙТУ

2.1. Проєктування веб-сайту

Проєктування веб-сайту — це етап, на якому формуються логіка роботи, архітектура сторінок, структура даних, а також взаємодія між користувачем, інтерфейсом та серверною частиною. Основна мета полягає у створенні такого ресурсу, який забезпечить швидкий доступ до асортименту страв, інтуїтивно зрозумілий процес замовлення, а також зручність адміністрування контенту.

Передусім, визначається загальна схема функціонування сайту. Користувач потрапляє на головну сторінку, де представлено навігаційне меню, категорії товарів, рекламні банери або слайдери з акційними пропозиціями. Перехід до певної категорії відкриває список страв із коротким описом, зображенням та ціною. Натискання на конкретний товар веде до сторінки з детальною інформацією — складом, доступними розмірами, варіантами соусів, відгуками тощо. Додавання товару в кошик відбувається без перезавантаження сторінки.

Кошик реалізується як окремий інтерфейсний блок, що дозволяє змінювати кількість товарів, переглядати підсумкову вартість, прибирати зайві позиції та переходити до оформлення. На етапі підтвердження замовлення користувач вказує контактні дані, обирає спосіб оплати та доставки, після чого замовлення зберігається в базі даних для подальшої обробки.

З боку адміністратора структура сайту дозволяє виконувати всі ключові операції — додавання, редагування та видалення товарів, перегляд замовлень, зміна їхнього статусу, модерація відгуків. Всі ці дії виконуються через спеціально створену адмін-панель, яка недоступна звичайним користувачам.

Логічна архітектура сайту побудована за принципом розділення відповідальностей: інтерфейс, серверна логіка, база даних. Клієнтська частина (front-end) відповідає за відображення сторінок і взаємодію з користувачем.

Серверна частина (back-end) обробляє запити, взаємодіє з базою даних, відповідає за авторизацію, перевірку введених даних, логіку замовлень. База даних зберігає всю інформацію про товари, категорії, користувачів, замовлення, відгуки та інші об’єкти.

На рівні сторінок веб-сайт має наступну структуру:

- головна сторінка;
- сторінка категорій;
- сторінка окремого товару;
- кошик;
- форма оформлення замовлення;
- сторінка входу та реєстрації (у вигляді спливаючого вікна);
- кабінет користувача (при потребі);
- адмін-панель.

Усі ці сторінки зв’язані між собою через маршрути та посилання. Основні елементи інтерфейсу повторюються для збереження цілісності дизайну: шапка, навігація, футер, кнопки дій.

Важливим аспектом проєктування є реалізація інтерактивності. Наприклад, при натисканні кнопки "Додати в кошик" викликається AJAX-запит, який оновлює вміст кошика без перезавантаження сторінки. Аналогічно, під час логіну чи реєстрації виводяться повідомлення про помилки або успішну авторизацію прямо у формі. Це підвищує зручність і пришвидшує взаємодію з системою.

Певна увага приділяється проєктуванню сторінки товару. Вона повинна містити якісне зображення, опис, вибір розміру або соусу, а також блок відгуків користувачів. У відгуках відображається ім’я покупця, дата та оцінка у вигляді зірок. Додавати відгук можуть лише авторизовані користувачі. В нижній частині сторінки також пропонуються схожі товари з тієї ж категорії.

При проєктуванні враховується також досвід адаптивної верстки: інтерфейс має коректно працювати на екранах мобільних телефонів і планшетів. Для цього елементи інтерфейсу мають гнучку ширину, перемикаються у

вертикальні блоки або спрощуються при звуженні вікна. Меню може змінюватись на бургер-кнопку, а таблиці перетворюються на картки.

Важливим є також візуальне оформлення: кольорова гама, шрифт, розміщення елементів. Вибрано темну тему з акцентними кольорами для кнопок та елементів дії, що надає сайту сучасного вигляду. Для заголовків використовується фірмовий шрифт, а для основного тексту — легкий для читання варіант без засічок.

Проєктування також передбачає окремий інтерфейс для адміністратора. У ньому є сторінка зі списком товарів, форма для створення та редагування товару (назва, опис, ціна, зображення, категорія), таблиця з усіма замовленнями, в якій можна змінити статус (нове, в обробці, відправлено, виконано). Усі ці функції мають бути доступні лише після авторизації адміністратора.

Ще один важливий етап — підготовка шаблонів для сторінок. Це дозволяє забезпечити повторне використання елементів інтерфейсу: шапки, меню, футера, блоків товару. Це прискорює розробку і спрощує підтримку сайту.

Таким чином, проєктування веб-сайту передбачає побудову логічної структури, зручної навігації, продуманої інтерактивності та стабільного серверного оброблення запитів. Усі елементи об’єднані в єдину систему, яка буде зрозумілою, зручною та ефективною як для кінцевого користувача, так і для адміністратора.

У межах адміністративного інтерфейсу також реалізовано механізм контролю над завантаженням файлів. При додаванні товару адміністратор завантажує зображення, яке автоматично перевіряється за типом та розміром. Допускаються лише графічні формати (JPEG, PNG, WEBP), а максимальний розмір файлу обмежується декількома мегабайтами. Це дозволяє уникнути випадкових помилок і захищає сервер від потенційного перевантаження або шкідливих скриптів у вкладених файлах.

Особливу увагу приділено валідації даних, що вводяться як на стороні користувача, так і на сервері. Наприклад, під час реєстрації перевіряється, чи правильно заповнено електронну пошту, чи введено достатньо символів у паролі,

чи немає вже зареєстрованого облікового запису з такою поштою. У формі оформлення замовлення також діє перевірка: чи заповнено всі обов’язкові поля (ім’я, телефон, адреса доставки), чи введено коректний номер телефону тощо. Сервер повторно перевіряє ці дані перед збереженням у базу, що унеможливлює обхід перевірок за допомогою сторонніх інструментів.

Ще один аспект — обробка помилок. У випадку технічного збою (наприклад, відсутність зв’язку з базою даних або помилка при збереженні замовлення), користувачу виводиться зрозуміле повідомлення з пропозицією спробувати пізніше або перевірити правильність введених даних. Помилки логуються на сервері для подальшого аналізу адміністратором або розробником. Це дозволяє швидко реагувати на збої та підтримувати стабільну роботу сервісу.

Захист особистих даних є ще одним критичним компонентом проєктування. Інформація про користувача (логіни, адреси, телефони) передається через захищене з’єднання (HTTPS) та зберігається у зашифрованому вигляді (наприклад, хешовані паролі). Доступ до адміністративної панелі обмежується перевіркою автентифікації: спроби несанкціонованого входу блокуються, IP-адреси з великою кількістю невдалих спроб можуть бути тимчасово заблоковані.

Також реалізовано механізми обмеження прав доступу: звичайний користувач не має доступу до внутрішніх функцій сайту, не бачить адміністративних маршрутів і не може відправити запити, що змінюють базу даних без належного рівня авторизації. Це досягається перевіркою сесій і ролей користувача на рівні сервера.

Оформлення замовлення — ключовий процес, і тому він отримав особливу увагу під час проєктування. Інтерфейс побудовано таким чином, щоб користувач бачив підсумковий склад замовлення, вартість, обрані параметри та форму для введення контактних даних на одній сторінці. Це знижує кількість кроків до покупки і підвищує завершуваність замовлень. Після підтвердження користувач отримує повідомлення про успішне оформлення, а адміністратор бачить нове замовлення у власній системі.

Усі ці компоненти спільно формують єдину гнучку й масштабовану систему, яка дозволяє не лише обслуговувати користувачів, а й керувати вмістом сайту без потреби у технічному втручанні. Така структура є передумовою стабільної роботи, зручної підтримки й можливого розширення функціоналу в майбутньому.

2.2. Проєктування десктопного застосунку

Десктопний застосунок є невід'ємною частиною програмного комплексу, що має на меті забезпечення більш гнучкого, інтерактивного й візуального підходу до формування замовлень. Якщо веб-сайт зосереджений переважно на процесі вибору з готового асортименту, то десктопна програма дозволяє користувачеві зібрати власну конфігурацію страви у візуальному режимі, що значно підвищує привабливість і залученість.

Основною метою створення десктопного застосунку є реалізація конструктора страв, орієнтованого на піцу. Користувач може поетапно додавати інгредієнти, змінювати тип соусу, обирати розмір основи, а також спостерігати за змінами у реальному часі на графічному відображенні. Такий функціонал реалізовано за допомогою мови програмування C# у середовищі Windows Forms із використанням графічної бібліотеки OpenGL для рендерингу 2D-графіки.

При проєктуванні інтерфейсу було прийнято рішення розділити вікно програми на три основні частини. Ліворуч розташовується список інгредієнтів із можливістю додавання. Праворуч — панель інформації: розмір піци, обраний соус, поточна вартість та кнопка очищення. Центральну частину займає область рендерингу, у якій шар за шаром накладаються обрані компоненти. Всі зміни користувач бачить миттєво, без потреби натискати додаткові кнопки оновлення.

Кожен інгредієнт реалізується у вигляді графічної текстури з прозорим фоном, що дозволяє комбінувати їх між собою без спотворення зображення. Наприклад, сир накладається поверх основи, а шматочки пепероні — на сир.

Текстури завантажуються у пам’ять при запуску програми та відображаються у відповідному порядку згідно з логікою побудови страви.

Розмір піци задається ще на початковому етапі. Від вибору розміру залежить як візуальний масштаб у вікні рендерингу, так і базова ціна. Наприклад, піца середнього розміру має базову вартість 100 одиниць, до якої додається ціна кожного інгредієнта. Всі значення зберігаються у структурі даних, яка оновлюється після кожної дії користувача.

Соус обирається через перемикачі: білий або червоний. Візуально соус не відображається як окрема текстура, а змінює відтінок основи. Таким чином, користувач бачить різницю в оформленні, що відповідає зробленому вибору. У разі зміни соусу або розміру програма не скидає інші параметри, а лише оновлює відповідну частину рендерингу та цінову інформацію.

Обробка взаємодії з інтерфейсом реалізована на основі подій: кожне натискання кнопки або зміна параметра запускає відповідну функцію оновлення. Це дозволяє уникнути дублювання коду та спрощує підтримку програми. Всі основні дані про поточне замовлення зберігаються у спеціальній структурі — об’єкті замовлення, який містить перелік інгредієнтів, соус, розмір та загальну вартість.

Програма також підтримує можливість збереження замовлення у вигляді текстового файла або передавання у форматі JSON, що забезпечує потенційну інтеграцію з веб-сайтом. Таким чином, адміністрація закладу може обробити замовлення як через десктопний застосунок, так і з веб-інтерфейсу, зберігаючи єдину структуру.

Окрема увага приділяється продуктивності. Хоча йдеться лише про 2D-графіку, велика кількість шарів і часте оновлення зображення можуть створювати навантаження. Тому при розробці застосовано механізм буферизації: оновлюється лише та частина сцени, яка була змінена. Це дозволяє зберігати плавність навіть на слабших системах.

З точки зору стійкості до помилок, програма перевіряє коректність даних: наприклад, не дозволяє обрати некоректний файл або запустити з порожньою

конфігурацією. Також передбачено механізм скидання — при натисканні кнопки “Очистити” застосунок повертається до початкового стану без необхідності перезапуску.

Проєктування інтерфейсу здійснювалося з урахуванням зручності: кнопки мають великі підписи, усі елементи згруповані логічно, кольорова схема витримана в єдиному стилі. Особливо важливо забезпечити візуальне співпадіння між діями користувача та результатами на екрані — кожен інгредієнт одразу відображається на основі, зміна розміру одразу впливає на ціну й масштаб, обраний соус — на кольоровий відтінок.

Функціонально застосунок охоплює весь цикл створення замовлення: від вибору компонентів до формування готової конфігурації. У перспективі можлива розробка багатостравного режиму, підтримки користувацьких шаблонів, авторизації для збереження історії. Проте у поточній версії основний фокус зроблено на стабільності, швидкодії та простоті використання.

Таким чином, десктопний застосунок не лише доповнює можливості сайту, а й виконує функцію інтерактивного конструктора, що може використовуватися як клієнтами, так і персоналом на точці прийому замовлень. Це підвищує універсальність системи та створює перевагу перед конкурентами, які обмежуються лише типовими інтернет-магазинами.

Висновки до розділу 2

У другому розділі було здійснено детальне проєктування двох ключових компонентів програмного комплексу — веб-сайту для онлайн-замовлень і десктопного застосунку для інтерактивного створення страв.

Для веб-сайту визначено чітку структуру інтерфейсу, побудовану навколо каталогу продукції, сторінки товару, кошика, форми оформлення замовлення та адміністративної панелі. Сформовано логіку маршрутизації, використання шаблонів, реалізацію AJAX-взаємодії без перезавантаження сторінок, а також обробку помилок і валідацію форм. Приділено увагу безпеці, зручності навігації

та адаптивності для мобільних пристроїв. Адміністративна частина дозволяє повноцінне керування товарами й замовленнями через захищений інтерфейс.

Для десктопного застосунку описано реалізацію візуального конструктора страв із використанням мови C# та графічної бібліотеки OpenGL. Спроєктовано інтерфейс, логіку накладання шарів інгредієнтів, облік вибору соусу та розміру, формування загальної вартості та структуру об’єкта замовлення. Також передбачено буферизацію для забезпечення швидкої роботи, а також функції збереження або передачі даних у зовнішню систему.

Таким чином, результати проєктування підтверджують доцільність обраної архітектури, яка поєднує веб- і десктопну складову в єдину гнучку систему. Обидва компоненти орієнтовані на користувача, забезпечують взаємодію в режимі реального часу та можуть масштабуватися відповідно до потреб бізнесу.

РОЗДІЛ ІІІ
ПРОЄКТУВАННЯ БАЗИ ДАНИХ WEB-САЙТУ

3.1. Обґрунтування вибору СУБД

Одним із ключових етапів при створенні будь-якого веб-проєкту є вибір системи управління базами даних (СУБД), що відповідає функціональним, технічним і масштабним вимогам проєкту. У контексті розробки веб-сайту для онлайн-замовлення фаст-фуду необхідно забезпечити надійне зберігання та обробку інформації про товари, категорії, користувачів, замовлення та відгуки, що робить роль СУБД центральною в архітектурі всього проєкту.

Першочергово слід окреслити основні вимоги, які були поставлені до майбутньої бази даних:

- підтримка багатокористувацького доступу з високою швидкодією;
- забезпечення транзакційності для операцій оформлення замовлення;
- проста інтеграція з PHP та іншими компонентами веб-сервера;
- можливість масштабування проєкту в майбутньому;
- активна підтримка та документованість системи;
- розвинений механізм захисту та керування правами доступу;
- кросплатформеність для можливого перенесення на інші сервери.

На основі цих критеріїв було розглянуто кілька популярних варіантів СУБД, серед яких: MySQL, PostgreSQL, SQLite, Microsoft SQL Server та MongoDB.

СУБД SQLite була відхилена ще на початковому етапі, оскільки не забезпечує повноцінного багатокористувацького доступу та не має вбудованого серверного механізму. Хоча SQLite зручна для простих мобільних застосунків або прототипів, для повноцінного веб-сайту з обробкою замовлень вона не підходить.

Microsoft SQL Server хоча й є потужним інструментом із широким спектром функцій, орієнтований переважно на Windows-платформи та глибоку

інтеграцію з продуктами Microsoft. Крім того, його використання часто пов’язане з ліцензуванням, що не відповідає критерію вільного та кросплатформного розгортання. У випадку нашого проєкту, розрахованого на стандартне хостингове середовище з PHP та Apache або Nginx, цей варіант був визнаний недоцільним.

MongoDB — документно-орієнтована NoSQL СУБД — має свої переваги при роботі з гнучкими структурами даних або в умовах високої динаміки змін у схемі. Однак для даного проєкту, де структура даних досить чітко визначена й передбачає наявність складних зв’язків (реляцій) між таблицями (наприклад, між користувачами та замовленнями, товарами та категоріями), реляційна модель виявляється більш придатною.

СУБД PostgreSQL є сучасною, потужною, повністю відкритою системою, що підтримує розширені типи даних, транзакції, перевірки, зовнішні ключі, складні запити та інші можливості. Проте при розробці проєктів із використанням PHP MySQL залишається де-факто стандартом, із яким найкраще інтегруються CMS, PHP-фреймворки та хостинг-платформи. PostgreSQL потребує глибшої конфігурації і, хоч і перевершує MySQL у ряді аспектів, може бути надлишковим для типового e-commerce-проєкту.

Після проведеного аналізу було прийнято рішення на користь використання MySQL — однієї з найпопулярніших у світі реляційних СУБД, яка є відкритою, швидкою, стабільною та чудово інтегрується з PHP. Вона має широку спільноту, активну підтримку, регулярні оновлення безпеки, а також просту у використанні систему керування користувачами та правами доступу. MySQL забезпечує ефективне виконання запитів навіть за умов значного навантаження, що важливо для сервісів, які працюють з великою кількістю одночасних користувачів.

Крім того, у MySQL зручно реалізовуються усі ключові об’єкти бази даних, необхідні для функціонування сайту:

- таблиці з зовнішніми ключами;
- індекси для швидкого пошуку;

- обмеження (constraints) для перевірки цілісності даних;
- транзакції для збереження цілісності при оформленні замовлень;
- механізми збережених процедур та тригерів для розширення логіки при необхідності.

З технічної точки зору, MySQL дозволяє налаштування як локального серверного середовища (наприклад, XAMPP, WAMP), так і повноцінного віддаленого розгортання на хостингу або VPS. У разі потреби перенесення — база легко експортується у форматі SQL і імпортується на новий сервер без змін у коді веб-застосунку.

Слід також зазначити, що обрана СУБД дозволяє при необхідності інтеграцію з десктопним застосунком. Оскільки структура таблиць є стандартизованою, а формат обміну даними (наприклад, JSON або SQL-інтерфейс) — відкритий, у майбутньому можна забезпечити передачу замовлення із візуального конструктора безпосередньо в єдину систему збереження.

Таким чином, вибір MySQL є обґрунтованим як з точки зору технічної відповідності проєктним вимогам, так і з позицій практичності, стабільності та підтримки. Це дозволяє зосередити увагу на логіці застосунку, не витрачаючи ресурси на додаткове адміністрування або налагодження взаємодії між компонентами.

3.2. Структура таблиць бази даних

Структура бази даних є основою функціонування будь-якого сучасного веб-сервісу. Саме вона визначає спосіб зберігання, обробки та взаємозв’язку всієї інформації, необхідної для коректної роботи системи. У межах проєкту онлайн-замовлення фаст-фуду база даних виконує роль центрального сховища даних для веб-сайту та, за потреби, для інтеграції з десктопним застосунком.

Ключовими об’єктами, що потребують зберігання, є користувачі, категорії товарів, окремі позиції продукції, замовлення, вміст кожного замовлення,

відгуки клієнтів, способи оплати та статуси доставки. Крім того, враховано можливість створення індивідуальної піци — для цього реалізована окрема таблиця, яка зберігає склад зібраних страв.

Усі таблиці побудовані згідно з принципами реляційної моделі: кожен запис має унікальний ідентифікатор (первинний ключ), а зв'язки між таблицями реалізовані за допомогою зовнішніх ключів. Завдяки цьому забезпечується цілісність даних, зменшується дублювання інформації та покращується масштабованість.

Всього база даних містить дев'ять таблиць. Нижче наведено їхній повний опис, призначення та структуру.

Таблиця users (табл. 3.1) є ключовим компонентом, що забезпечує ідентифікацію та персоналізацію взаємодії користувача з сайтом. У ній зберігаються облікові записи клієнтів, зокрема їхні контактні дані, захешовані паролі та дата реєстрації. Поле is_admin дозволяє розмежовувати типи доступу: адміністратор має змогу редагувати асортимент, переглядати замовлення та змінювати їх статус, тоді як звичайний користувач обмежений лише функціями замовлення та залишення відгуків. Таблиця пов'язана з іншими — наприклад, з orders та reviews, — через зовнішній ключ user_id, що дозволяє точно відстежити, хто здійснив покупку або залишив відгук.

Таблиця 3.1

Опис таблиці users

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Унікальний ідентифікатор користувача
name	VARCHAR(255)	NOT NULL	Ім'я користувача
email	VARCHAR(255)	NOT NULL	Email (унікальний)
password	VARCHAR(255)	NOT NULL	Захешований пароль
created_at	DATETIME	NOT NULL	Дата створення облікового запису
is_admin	BOOLEAN	NOT NULL	Ознака адміністратора

Таблиця categories (табл. 3.2) служить для логічного поділу всього асортименту продукції на окремі групи, такі як «Піца», «Бургери», «Напої», «Десерти» тощо. Кожен товар, що зберігається у таблиці products, обов’язково належить до певної категорії через поле category_id. Завдяки цьому реалізується фільтрація на головній сторінці, організація меню, побудова навігації та блок «схожі товари». Такий підхід забезпечує зручність користування сайтом, а також спрощує адміністрування та подальше розширення асортименту.

Таблиця 3.2

Опис таблиці categories

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор категорії
name	VARCHAR(255)	NOT NULL	Назва категорії

Таблиця products (табл. 3.3) є однією з центральних у структурі бази даних. Вона містить усю ключову інформацію про кожен товар, доступний на сайті: назву, опис, ціну, зображення та категорію. Через поле category_id вона зв’язана з таблицею categories, що дозволяє відображати товари в правильних розділах. Інші таблиці, такі як order_items та reviews, також посилаються на products, дозволяючи формувати замовлення й залишати відгуки. У перспективі таблиця може бути доповнена додатковими полями (наприклад, для наявності на складі, калорійності чи позначки «новинка»), що робить її гнучкою основою для масштабування сайту.

Таблиця 3.3

Опис таблиці products

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор товару
name	VARCHAR(255)	NOT NULL	Назва товару
description	TEXT	NULL	Опис
price	DECIMAL(10,2)	NOT NULL	Ціна
image	VARCHAR(255)	NULL	Шлях до зображення
category_id	INT	NOT NULL	Зовнішній ключ на таблицю categories

Таблиця orders (табл. 3.4) зберігає дані про всі оформлені замовлення. Вона є центральною у процесі купівлі та слугує точкою зв’язку між користувачем і конкретним вмістом його замовлення. Поле user_id вказує, хто саме зробив замовлення (авторизований користувач або анонім — якщо дозволено), а payment_method та delivery_status посилаються на довідкові таблиці, що дозволяє легко змінювати або розширювати способи оплати та статуси обробки. Поля phone та address використовуються для доставки, а created_at дозволяє відслідковувати хронологію покупок. Таблиця orders безпосередньо пов’язана з таблицями order_items та custom_pizzas, що забезпечує зберігання всіх позицій замовлення, як звичайних, так і сконструйованих індивідуально.

Таблиця 3.4

Опис таблиці orders

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор замовлення
user_id	INT	NULL	Зовнішній ключ на користувача
total_price	DECIMAL(10,2)	NOT NULL	Загальна сума
payment_method	INT	NOT NULL	Зовнішній ключ на спосіб оплати
delivery_status	INT	NOT NULL	Зовнішній ключ на статус доставки
phone	VARCHAR(20)	NOT NULL	Телефон
address	TEXT	NOT NULL	Адреса доставки
created_at	DATETIME	NOT NULL	Час створення

Таблиця order_items (табл. 3.5) реалізує зв’язок «багато до багатьох» між таблицями orders і products. Кожен рядок у order_items вказує, яка саме позиція (товар) входить до конкретного замовлення і в якій кількості. Завдяки цьому забезпечується гнучка модель додавання необмеженої кількості товарів у

замовлення. Таблиця також дозволяє вести статистику за популярністю страв, підраховувати виторг і формувати звіти по продажах.

Таблиця 3.5

Опис таблиці order_items

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор елемента
order_id	INT	NOT NULL	Зовнішній ключ на таблицю orders
product_id	INT	NOT NULL	Зовнішній ключ на таблицю products
quantity	INT	NOT NULL	Кількість одиниць

Таблиця reviews (табл. 3.6) відповідає за функціонал зворотного зв’язку. У ній користувачі залишають оцінки й коментарі до замовлених товарів. Зв’язок з таблицями users та products дозволяє відстежити, хто саме і до якого продукту залишив відгук. Такі дані можуть бути використані не лише для демонстрації довіри новим клієнтам, а й для вдосконалення асортименту відповідно до побажань покупців. Оцінки (від 1 до 5) можуть бути використані для побудови рейтингової системи та сортування продукції.

Таблиця 3.6

Опис таблиці reviews

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор відгуку
user_id	INT	NOT NULL	Зовнішній ключ на користувача
product_id	INT	NOT NULL	Зовнішній ключ на товар
rating	INT	NOT NULL	Оцінка (1–5)
message	TEXT	NULL	Текст відгуку
created_at	DATETIME	NOT NULL	Дата й час

Таблиця payment_methods (табл. 3.7) містить довідкову інформацію про способи оплати, доступні під час оформлення замовлення. Поточна реалізація включає варіанти «готівка» і «банківська картка», але таблиця легко розширюється без зміни логіки системи. Використання окремої таблиці замість збереження рядкового значення в orders дозволяє підтримувати цілісність даних та уникати помилок у написанні.

Таблиця 3.7

Опис таблиці payment_methods

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор способу
name	VARCHAR(255)	NOT NULL	Назва (готівка, картка)

Таблиця delivery_statuses (табл. 3.8) є ще одним довідником, який визначає етапи виконання замовлення: «Прийнято», «Готується», «В дорозі», «Доставлено» тощо. Це дозволяє адміністраторам змінювати статус вручну або автоматично, а користувачам — відстежувати прогрес свого замовлення. Такий підхід підтримує прозорість сервісу та покращує взаємодію з клієнтами.

Таблиця 3.8

Опис таблиці delivery_statuses

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор статусу
name	VARCHAR(255)	NOT NULL	Назва статусу доставки

Таблиця custom_pizzas (табл. 3.9) реалізує можливість створення користувачем власної піци за допомогою візуального конструктора. До таблиці заносяться параметри кожної зібраної піци: соус, розмір, перелік інгредієнтів (у форматі списку або JSON) та обчислена загальна вартість. Кожен запис також пов’язаний з певним замовленням через поле order_id. Завдяки цьому

реалізується не лише індивідуальний підхід до клієнта, а й можливість аналітики щодо найчастіше обраних комбінацій.

Таблиця 3.9

Опис таблиці custom_pizzas

Поле	Тип даних	NULL	Опис
id	INT	NOT NULL	Ідентифікатор піци
order_id	INT	NULL	Замовлення, до якого належить піца
sauce	VARCHAR(50)	NOT NULL	Вибраний соус
size	VARCHAR(20)	NOT NULL	Розмір (малий, середній, великий)
ingredients	TEXT	NOT NULL	Список інгредієнтів (JSON або список)
price	DECIMAL(10,2)	NOT NULL	Загальна ціна піци

Таким чином, структура бази даних охоплює всі необхідні аспекти для повноцінного функціонування системи онлайн-замовлення фаст-фуду: управління асортиментом, обробку замовлень, зберігання інформації про клієнтів, опрацювання відгуків і підтримку індивідуальних замовлень. Всі таблиці логічно взаємопов’язані, а модель бази є розширюваною, що дозволяє надалі доповнювати її новими можливостями без кардинальної перебудови. Така структурована організація даних забезпечує не лише ефективність обробки запитів, але й підтримку високого рівня зручності як для користувача, так і для адміністратора системи.

3.3. Реляційний зв’язок між таблицями

Однією з основних переваг реляційної моделі баз даних є можливість створення чітких, формалізованих зв’язків між таблицями, що дозволяє ефективно організувати та підтримувати логічну цілісність даних. У межах даного проєкту розроблена структура, яка відображає всі ключові сутності

системи (користувачі, товари, замовлення тощо) та їхні взаємозв'язки, необхідні для коректного функціонування як клієнтської, так і адміністративної частини веб-сайту.

Для моделювання зв'язків між сутностями були використані зовнішні ключі (foreign keys), які дозволяють гарантувати зв'язність записів і запобігають появі неконсистентних даних у системі. Кожен зовнішній ключ вказує на первинний ключ пов'язаної таблиці, встановлюючи логічну залежність між рядками.

Загальна структура взаємозв'язків між таблицями зображена на схемі (Рис. 3.1).

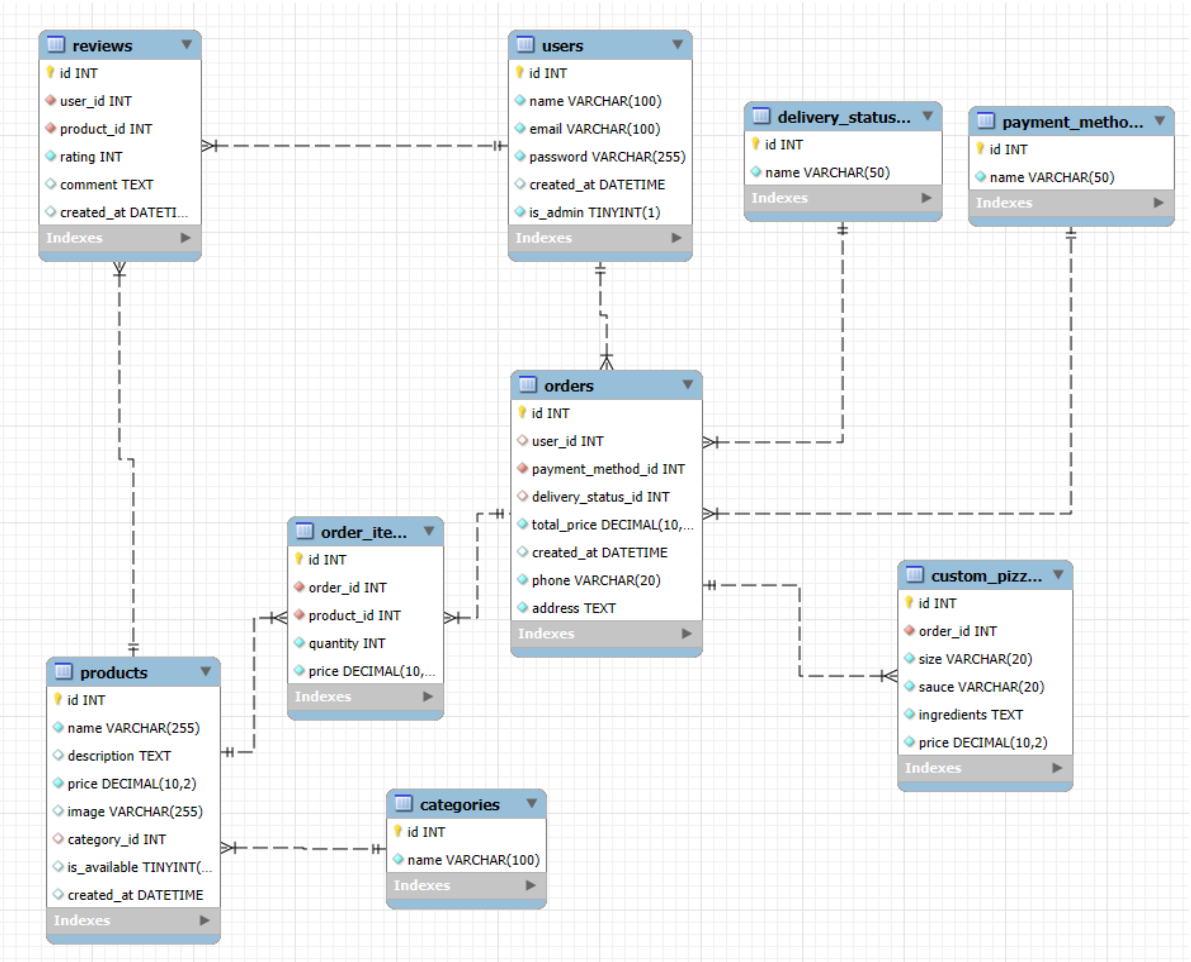

Рис. 3.1. Реляційна схема бази даних проєкту

На схемі можна виділити кілька основних груп зв'язків:

1. Таблиця orders пов'язана з таблицею users через поле user_id. Це типовий зв'язок «один до багатьох», коли один користувач може мати декілька

замовлень, але кожне замовлення належить лише одному користувачу. Такий підхід дозволяє будувати історію покупок і здійснювати аналітику клієнтської активності.

2. Таблиця order_items пов’язана з orders через поле order_id, що означає, що кожне замовлення може включати декілька товарів. У свою чергу, поле product_id вказує на конкретний товар із таблиці products. Це реалізує зв’язок «багато до багатьох» між замовленнями та товарами, але через проміжну таблицю order_items.
3. Кожен товар у таблиці products має поле category_id, що є зовнішнім ключем на таблицю categories. Це дозволяє логічно групувати продукцію за розділами меню. Зв’язок має тип «багато до одного», де одна категорія може включати багато товарів, але кожен товар належить лише до однієї категорії.
4. Таблиця reviews одночасно пов’язана з users (через user_id) і products (через product_id). Це дає змогу зберігати не лише зміст відгуку, але й дані про те, хто його залишив і до якого товару він належить. Таким чином реалізовано два зв’язки «багато до одного»: багато відгуків — до одного товару; багато відгуків — від одного користувача.
5. У таблиці orders поля payment_method_id та delivery_status_id є зовнішніми ключами на довідкові таблиці payment_methods та delivery_statuses відповідно. Ці зв’язки мають тип «багато до одного». Вони дозволяють зберігати поточний стан замовлення і обраний спосіб оплати у вигляді числового ідентифікатора, що спрощує підтримку системи та її масштабування (наприклад, додавання нових статусів або способів оплати).
6. Таблиця custom_pizzas пов’язана з таблицею orders через поле order_id, що дозволяє прикріплювати кастомні піци до конкретного замовлення. Усі параметри, включно з соусом, розміром та списком інгредієнтів, зберігаються безпосередньо в одному рядку. Це приклад зв’язку «багато

до одного», який розширює можливості замовлення без ускладнення загальної структури.

Крім самих зв’язків, важливо також зазначити, що використання зовнішніх ключів у базі даних дозволяє:

- автоматично підтримувати референтну цілісність: наприклад, не можна видалити користувача, якщо в нього є замовлення;
- унеможливити помилкові посилання: якщо вказаний product_id, якого немає у products, запис не буде збережено;
- спростити JOIN-запити до БД, що підвищує швидкодію сайту.

Таким чином, побудована реляційна модель відображає всі ключові бізнес-зв’язки між об’єктами системи. Вона не лише забезпечує цілісність даних і логічну структуру, а й дозволяє ефективно масштабувати й адаптувати проєкт у майбутньому.

Висновки до розділу 3

У третьому розділі було здійснено проєктування логічної структури бази даних, що є основою для зберігання, обробки та аналізу інформації в системі онлайн-замовлення фаст-фуду. На основі аналізу вимог до функціоналу веб-сайту обґрунтовано вибір системи управління базами даних — MySQL, як надійного і продуктивного рішення з відкритим кодом, яке підтримує складні реляційні зв’язки, забезпечує високу швидкодію та масштабованість.

Було розроблено логічну модель, яка включає дев’ять основних таблиць: користувачі, категорії товарів, окремі позиції продукції, замовлення, склад замовлення, індивідуальні піци, відгуки, способи оплати та статуси доставки. Для кожної з таблиць визначено структуру, типи даних, описано функціональне призначення та взаємозв’язки з іншими таблицями. Особливу увагу приділено практичному аспекту реалізації — таблиці спроєктовані з урахуванням реального навантаження та гнучкої підтримки змін.

Реляційні зв’язки між таблицями реалізовано за допомогою зовнішніх ключів, що гарантує цілісність даних і логічну узгодженість усіх записів. Побудована схема дозволяє ефективно обслуговувати ключові бізнес-процеси, зокрема обробку замовлень, перегляд історії покупок, формування складу замовлення, відображення категорій і товарів, а також взаємодію з користувачами через відгуки.

Завдяки продуманій структурі та коректному використанню реляційної моделі, база даних повністю готова до інтеграції з клієнтською частиною веб-сайту, а також із адміністративним інтерфейсом. Розроблена логічна модель може бути використана як основа для фізичної реалізації проєкту та подальшого вдосконалення функціоналу системи без зміни її фундаментальної структури.

РОЗДІЛ IV
РОЗРОБКА WEB-САЙТУ ТА ДОДАТКУ

4.1. Обґрунтування вибору технологій розробки

У межах реалізації інформаційної системи для продажу фаст-фуду з можливістю доставки ключовим етапом стало обґрунтування вибору програмних технологій та інструментів, які забезпечать ефективну, стабільну та масштабовану роботу як користувацької, так і адміністративної частин веб-сайту. У виборі технологій було враховано не лише технічні характеристики, але й доступність засобів розробки, простоту підтримки, сумісність із середовищем хостингу, безпеку та можливість майбутнього розширення функціоналу.

Для створення користувацького інтерфейсу веб-сайту обрано класичний стек технологій — HTML5, CSS3 та JavaScript. Ці мови є універсальними стандартами у фронтенд-розробці та дозволяють створити адаптивний, кросбраузерний і мобільно-орієнтований інтерфейс. HTML5 забезпечує семантичну структуру контенту та дозволяє використовувати сучасні елементи, що спрощують розмітку. CSS3 дає змогу реалізувати привабливий зовнішній вигляд, включно з анімаціями, адаптивністю та гнучкою системою позиціонування елементів. JavaScript використовується для динамічної взаємодії з користувачем: реалізації спливаючих вікон, динамічної зміни форм, обробки натискань без перезавантаження сторінки, а також — для реалізації асинхронних запитів через AJAX.

На серверному боці для обробки запитів та логіки веб-сайту обрано мову PHP, яка є однією з найпоширеніших у веб-розробці завдяки відкритому коду, широкому ком’юніті, простоті інтеграції з MySQL та доступності в більшості хостингів. PHP застосовується для реалізації авторизації та реєстрації користувачів, обробки замовлень, взаємодії з базою даних (через MySQLi), валідації даних, реалізації адміністративного інтерфейсу, завантаження зображень та обробки форм. Використання AJAX — технології, що дозволяє

обмінюватися даними з сервером у фоновому режимі — дало змогу реалізувати зручну логіку без перезавантаження сторінок, зокрема під час взаємодії з кошиком, реєстрації, авторизації та надсилання відгуків.

У якості системи управління базами даних використано MySQL — реляційну СУБД з відкритим кодом, яка відома своєю стабільністю, високою продуктивністю та широкою підтримкою серед PHP-розробників. Вона дозволяє будувати складні SQL-запити, забезпечує підтримку зовнішніх ключів, індексів і транзакцій, що є критичним для збереження цілісності даних у замовленнях, користувачах і товарах. Крім того, MySQL легко інтегрується з phpMyAdmin — інструментом, який спрощує управління базою даних на рівні адміністратора.

Оскільки система передбачає обробку персональних даних користувачів (логіни, адреси доставки, телефони), важливим критерієм став рівень безпеки обраних технологій. PHP дозволяє хешувати паролі за допомогою функції password_hash(), захищати базу від SQL-ін’єкцій за допомогою підготовлених запитів, попереджати XSS-атаки завдяки екрануванню HTML-вмісту.

Для забезпечення доступності сайту на мобільних пристроях застосовано адаптивну верстку з використанням Flexbox та медіа-запитів. Такий підхід дозволяє автоматично адаптувати розташування блоків до розміру екрана, забезпечуючи комфортну роботу на смартфонах і планшетах. Зображення товарів оптимізовано для швидкого завантаження. Сайт підтримує як класичні формати (JPEG, PNG), так і WebP — сучасний формат, що дозволяє зменшити розмір файлу без втрати якості.

Особливу роль у структурі проєкту відіграє адміністративна частина. Вона реалізована на тому ж стеку технологій, що й клієнтська частина, але містить додаткові перевірки автентичності користувача. Адміністративна панель дозволяє керувати асортиментом товарів, переглядати й змінювати статуси замовлень, аналізувати залишені відгуки. Такий підхід забезпечує повний контроль над вмістом сайту з боку адміністратора, без необхідності звертатися до бази даних напряму.

Важливо, що обрані технології дозволяють масштабувати систему у майбутньому. Проєкт легко розгортається на більшості серверів, підтримує резервне копіювання, та може бути адаптований для роботи з API, фреймворками або мобільними застосунками. Крім того, обраний стек дозволяє організувати взаємодію з десктопним застосунком, який може працювати з тією ж базою даних — забезпечуючи єдиний інформаційний простір для всієї системи.

4.2. Інтерфейс користувача сайту

Однією з центральних частин клієнтського інтерфейсу є головна сторінка, яка виконує роль каталогу продукції. Її оформлення побудоване за принципом сітки товарних карток, що дозволяє користувачеві швидко переглядати доступні страви, порівнювати їх візуально та приймати рішення про покупку. Вгорі розташовано навігаційне меню з категоріями («Бургери», «Десерти», «Напої», тощо), а також зручно винесені елементи інтерфейсу — кнопка переходу до профілю, кошика, трекінг замовлення та пошукове поле.

Кожна товарна картка містить зображення продукту, назву, короткий опис, ціну та кнопку «Додати до кошика». Завдяки темній темі інтерфейсу яскраві елементи привертають увагу, а кольорові кнопки не зливаються з фоном, що підвищує зручність навігації. Сортування за ціною дозволяє переглядати асортимент відповідно до власного бюджету. Уся структура побудована з акцентом на швидкість взаємодії та мінімізацію дій з боку користувача (Рис. 4.1).

Особливістю реалізації є адаптивність головної сторінки: структура автоматично змінюється відповідно до ширини екрана, що дозволяє зручно користуватись сайтом як на комп’ютерах, так і на мобільних пристроях. Картки продуктів масштабуються та впорядковуються по-новому залежно від розміру вікна браузера, а меню трансформується в компактну навігацію. Це дозволяє не лише зберегти всі функціональні можливості на різних пристроях, але й підтримувати привабливість дизайну. Загалом, головна сторінка сайту поєднує

функціональність інтернет-магазину з простотою інтерфейсу мобільного додатку.

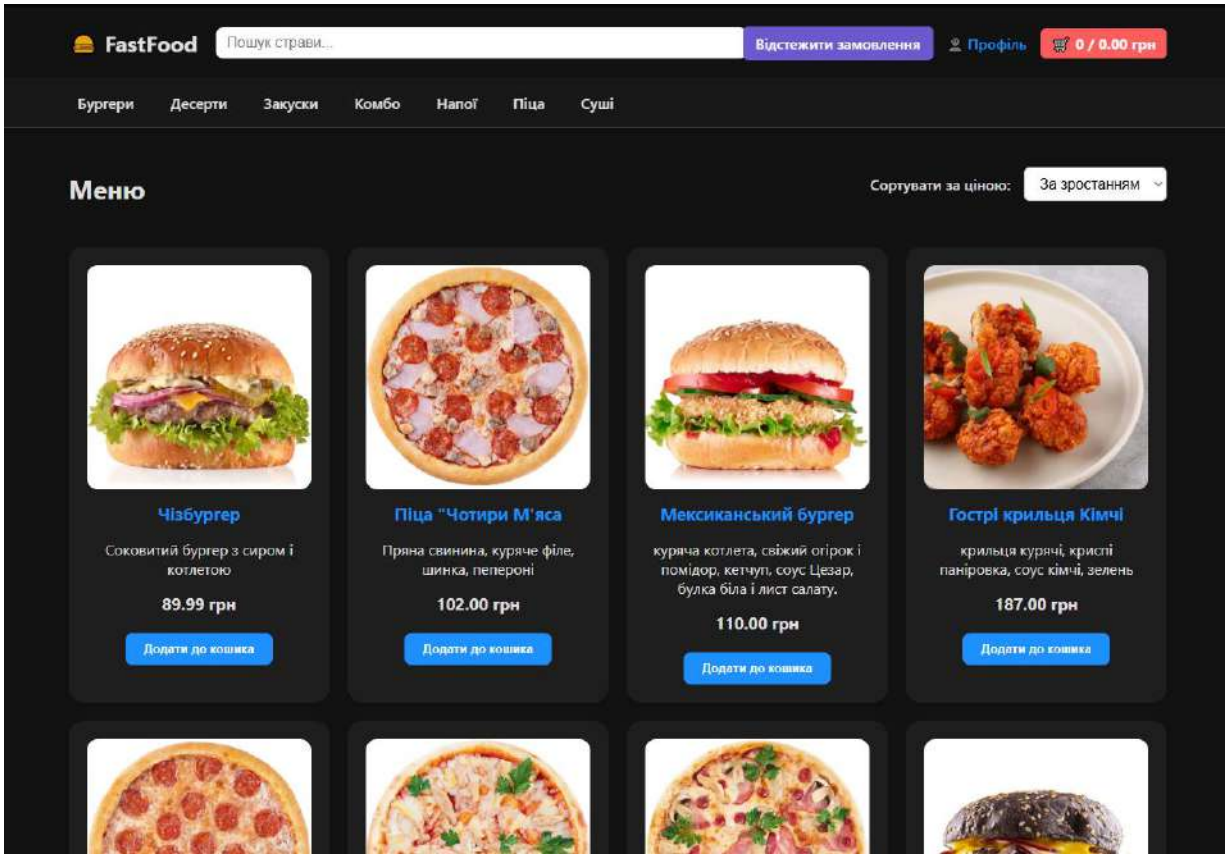

Рис. 4.1. Головна сторінка сайту

При переході на сторінку конкретного товару користувач отримує детальну інформацію про вибраний продукт. Основна частина сторінки складається з великого зображення, назви, опису, ціни, а також елементів управління кількістю — кнопок додавання або зменшення товару перед додаванням до кошика. Така структура дозволяє максимально швидко ознайомитись із позицією, не відволікаючись на другорядні елементи. Кнопки взаємодії винесено окремо, вони мають яскравий вигляд і доступні навіть на мобільних пристроях без прокручування (Рис. 4.2).

Нижче розміщується блок відгуків. Для неавторизованих користувачів виводиться підказка з пропозицією увійти для залишення власної оцінки. Вже наявні коментарі відображаються разом з ім’ям користувача, кількістю зірок та текстом. Такий механізм підтримує прозорість і підвищує довіру до товару, особливо в ситуації великого вибору. Окремим блоком під карткою товару

виводяться схожі позиції, що дозволяє користувачеві швидко перейти до альтернатив без повернення на головну сторінку.

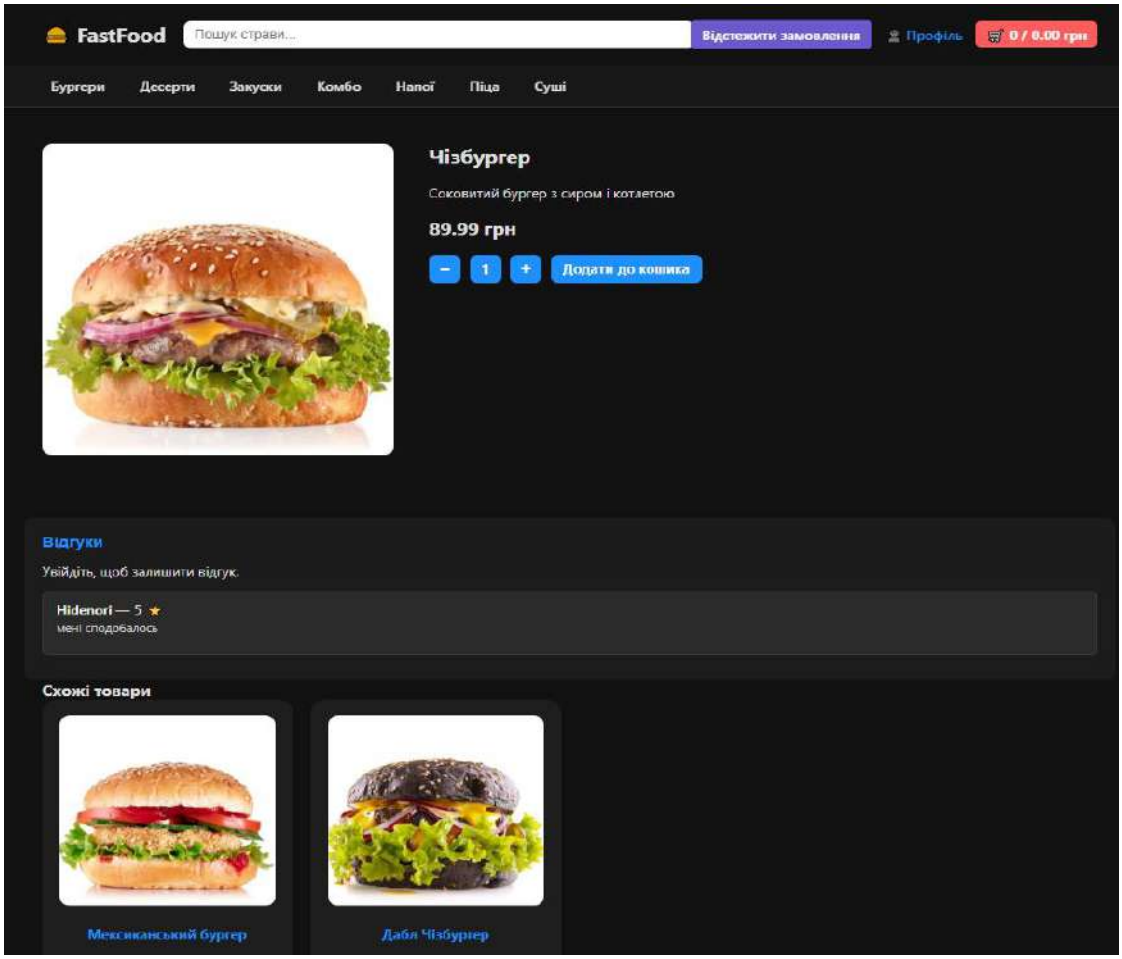

Рис. 4.2. Сторінка окремого товару

У системі передбачено механізм авторизації користувачів через спливаюче модальне вікно, що з’являється без необхідності переходу на окрему сторінку. Це рішення підвищує зручність взаємодії, оскільки користувач залишається на тій самій сторінці, де почав сесію (наприклад, картці товару). Реєстраційна форма є мінімалістичною і містить лише три поля — ім’я, email і пароль, що знижує бар’єр для першого входу. Дані вводяться у прості текстові поля, після чого обробляються через AJAX-запит без перезавантаження сторінки (Рис. 4.3).

Таке спрощення процедури реєстрації відповідає принципам сучасного UX-дизайну: швидка дія, зрозуміла структура, миттєвий результат. Крім того, під формою виведено посилання для переходу до входу в акаунт, що дозволяє без зайвих кліків змінити режим у випадку, якщо користувач вже має обліковий

запис. Усі валідаційні помилки, включно з порожніми полями чи некоректною електронною адресою, виводяться одразу в цьому ж вікні.

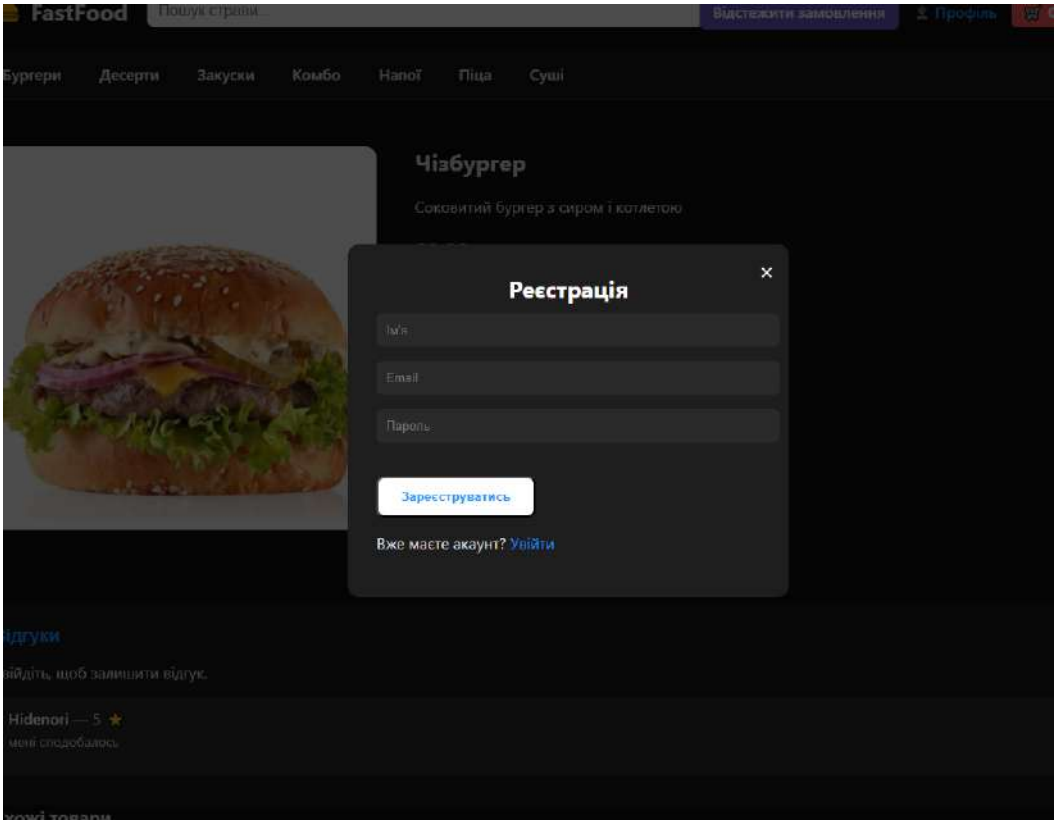

Рис. 4.3. Вікно реєстрації

Кошик є ключовим елементом процесу замовлення, адже саме тут користувач отримує повний огляд вибраних товарів перед підтвердженням покупки. У таблиці відображається зображення страви, її назва, ціна, кількість і підсумкова сума по кожній позиції. Для зручності додано кнопки зміни кількості (+/−), які дозволяють оперативно змінювати вміст кошика без необхідності оновлення сторінки. Кожна позиція може бути також видалена натисканням на іконку червоного хрестика (Рис. 4.4).

Внизу сторінки підсумовується загальна вартість усіх обраних товарів, а кнопка «Оформити замовлення» забезпечує перехід до фінального етапу — введення контактних даних та підтвердження. Завдяки простому дизайну та акцентам на діях користувача (наприклад, виділення кількості й підсумкової ціни), інтерфейс є інтуїтивним і не перевантаженим другорядною інформацією.

Така реалізація знижує ймовірність помилок при оформленні та підвищує ефективність роботи з кошиком.

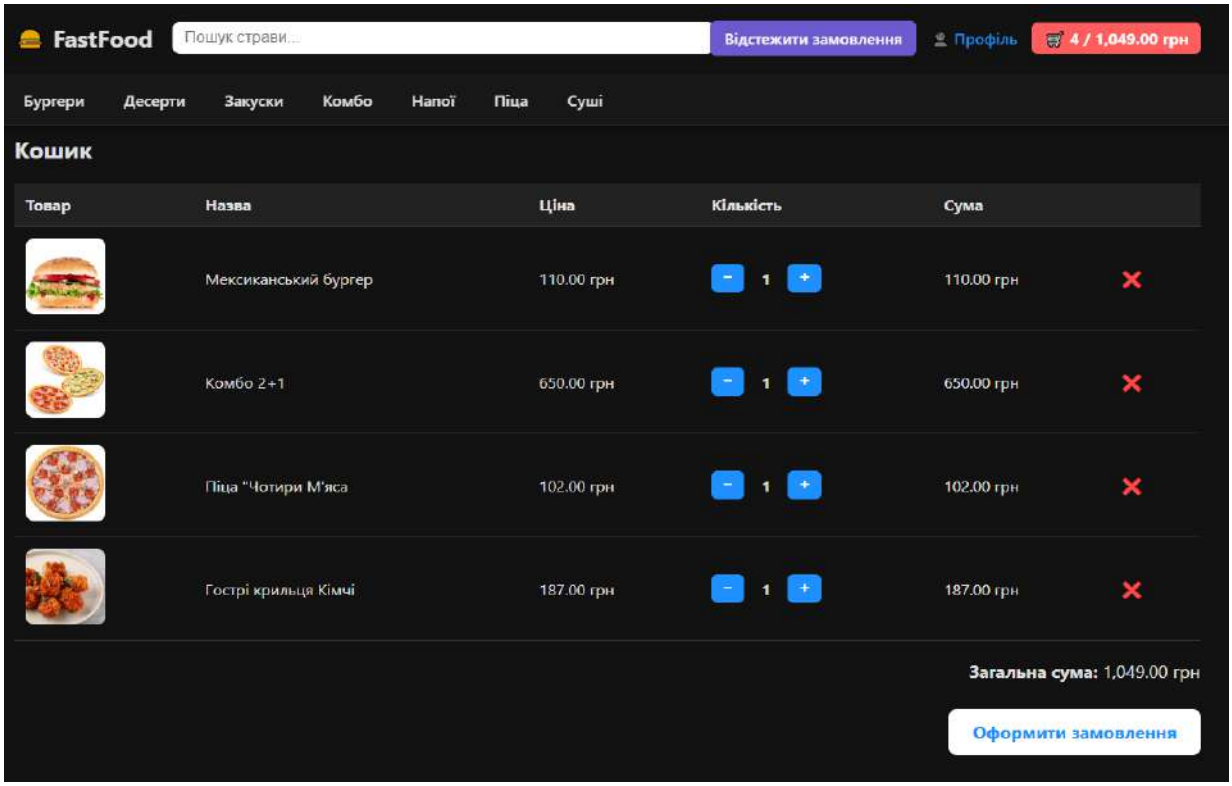

Рис. 4.4. Сторінка кошику

Після заповнення кошика користувач переходить до завершального етапу — оформлення замовлення. Ця форма побудована таким чином, щоб зібрати мінімальний, але достатній обсяг інформації для виконання доставки. Інтерфейс містить три поля: номер телефону, адресу доставки та спосіб оплати. Кожне поле супроводжується підказкою або типовим форматом введення (наприклад, номер телефону у форматі +380), що допомагає зменшити кількість помилок при заповненні (Рис. 4.5).

Спосіб оплати реалізовано у вигляді випадаючого списку, що дозволяє обрати один із доступних варіантів — наразі це готівковий розрахунок. В майбутньому можливе розширення цього списку з додаванням онлайн-платежів чи оплати при отриманні карткою. Кнопка «Підтвердити замовлення» має яскравий вигляд і розташована безпосередньо під формою, що забезпечує швидкий і логічний фінал взаємодії користувача із сайтом. Після підтвердження

всі дані надсилаються на сервер через безпечний POST-запит для збереження в базі.

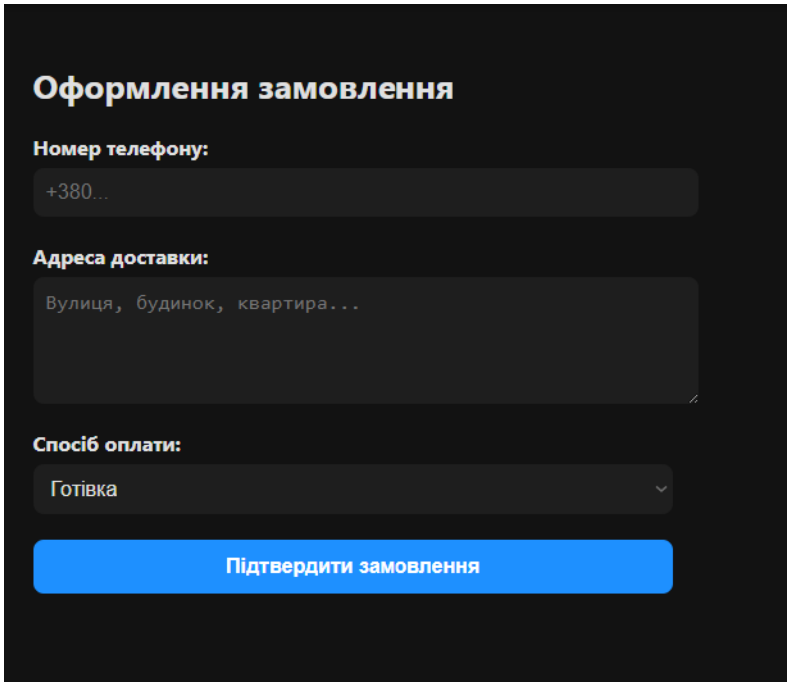

Рис. 4.5. Сторінка оформлення замовлення

Однією з ключових функцій для зручності навігації стала реалізація пошуку товарів. Вона доступна у вигляді текстового поля у верхній частині інтерфейсу. Користувач може ввести назву страви повністю або частково — і система миттєво виведе релевантні результати. Наприклад, введення запиту «чізбургер» повертає всі товари, у назві яких присутнє це слово: як класичний, так і варіації, що відрізняються складом або ціною (Рис. 4.6).

Функція пошуку реалізована на стороні сервера за допомогою SQL-запиту з оператором LIKE, який дозволяє шукати часткові збіги. Завдяки цьому користувач не зобов’язаний точно пам’ятати назву страви — достатньо кількох символів. Усі знайдені результати відображаються в тій самій візуальній формі, що й на головній сторінці: картка з фото, назва, ціна та кнопка «Додати до кошика». Це дозволяє одразу виконати дію без переходу на сторінку товару.

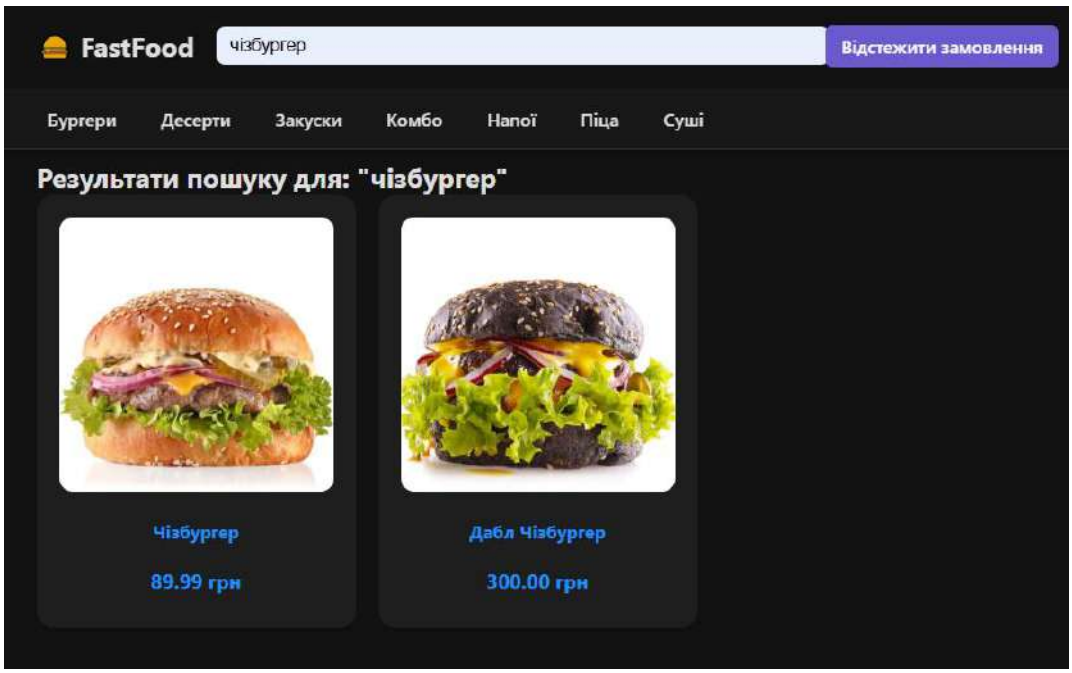

Рис. 4.6. Система пошуку товарів

Загалом, інтерфейс користувача реалізовано з урахуванням сучасних принципів UX-дизайну: зрозуміле розміщення елементів, інтуїтивна навігація, зручні інструменти взаємодії (пошук, кошик, фільтрація, AJAX-підтвердження). Темна тема з яскравими кнопками та адаптивна верстка роблять сайт зручним як на ПК, так і на мобільних пристроях. Усі ключові процеси — перегляд, додавання до кошика, реєстрація, замовлення — виконуються з мінімальною кількістю дій. Інтерфейс побудований логічно, з чітким поділом зон відповідальності: меню, контент, дії. Це забезпечує позитивний досвід взаємодії для користувача, а також зменшує навантаження на службу підтримки та адміністрування ресурсу.

4.3. Десктопний застосунок на C#

Крім веб-сайту, для зручності локального використання було реалізовано десктопний застосунок, розроблений мовою C# з використанням Windows Forms у поєднанні з бібліотекою OpenGL. Основна мета — надати клієнту візуальний інструмент для формування індивідуального замовлення піци або бургера з

можливістю перегляду у режимі конструктора. Такий формат дозволяє підвищити інтерактивність, а також відкриває перспективу для роботи на сенсорних панелях у закладах харчування.

Головне вікно застосунку являє собою інтерактивне середовище, в якому користувач може вибрати базу (корж піци або булку), обрати тип соусу (білий чи червоний), а також поступово додавати інгредієнти з набору доступних компонентів. Інгредієнти представлені у вигляді кнопок із відповідними позначеннями, а їх візуальне розміщення здійснюється у вікні відображення через OpenGL. Таким чином, користувач отримує графічну модель обраної комбінації, що підвищує залучення та дозволяє перевірити вигляд страви ще до підтвердження (Рис. 4.7).

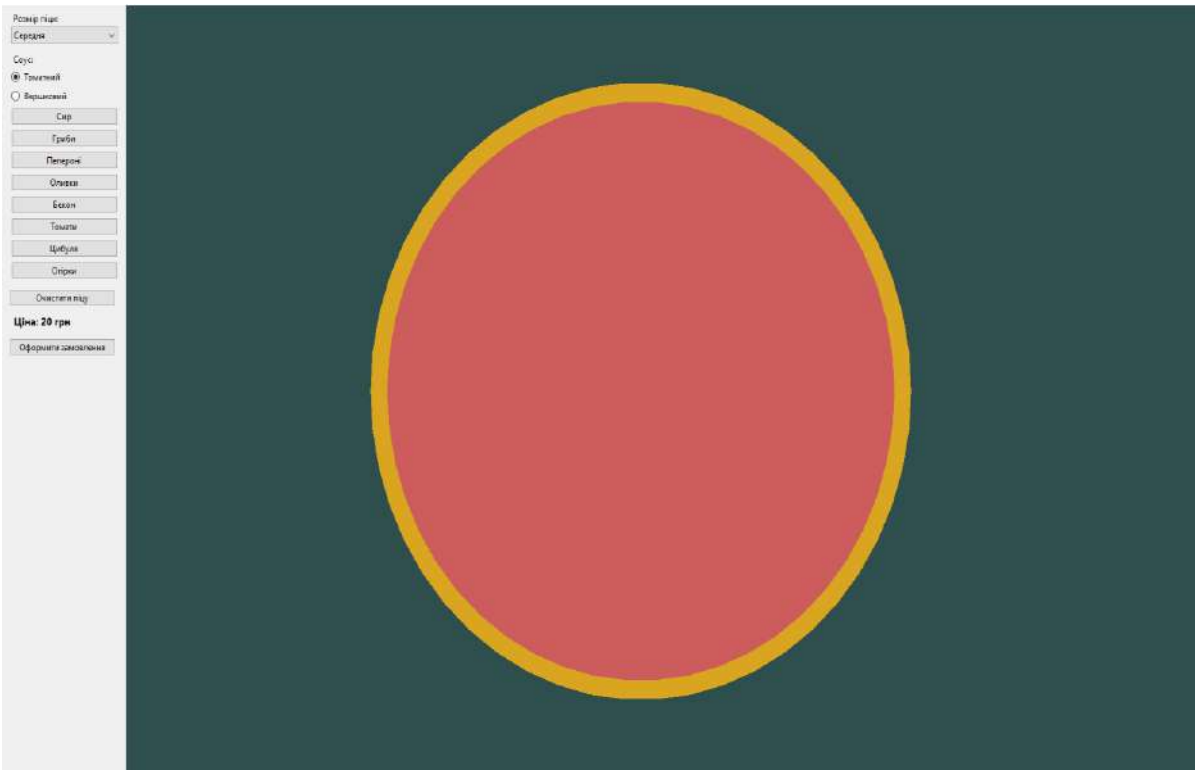

Рис. 4.7. Інтерфейс додатку

У процесі додавання кожен інгредієнт відображається у центрі піци або бургера з прозорістю та позиційним зміщенням, щоб зберігати візуальну реалістичність. До наявного списку компонентів входять: сир, пепероні, гриби, томати, цибуля, оливки, бекон, огірки. Додаток також дозволяє повторно додавати той самий інгредієнт, і для кожного розраховується індивідуальна ціна.

Загальна вартість формується автоматично на основі базової вартості й доданих елементів. Це забезпечує прозору систему ціноутворення для користувача (Рис. 4.8).

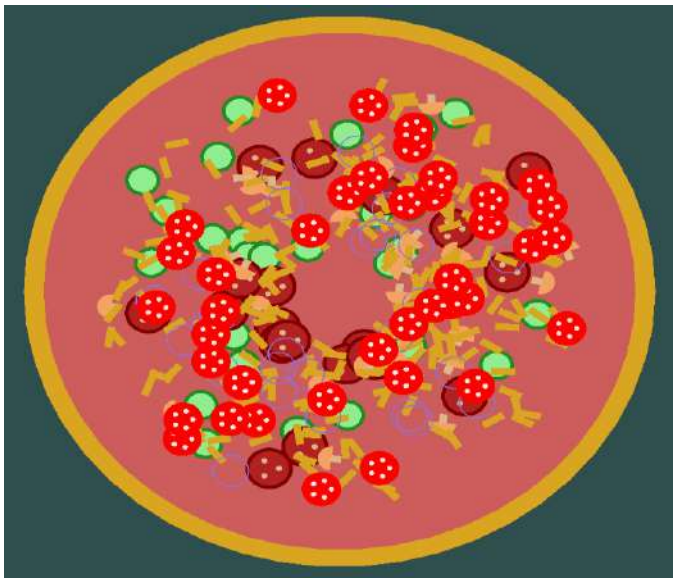

Рис. 4.8. Створена піца

Після завершення вибору доступна форма підтвердження, де користувач вводить номер телефону та адресу доставки. Такі дані зберігаються в локальній базі або можуть передаватись через API (у разі підключення до мережевої інфраструктури сайту). Кнопка «Відправити замовлення» фіксує всі обрані параметри у форматі замовлення: тип соусу, розмір, набір інгредієнтів, ціна, а також контактні дані (Рис. 4.9). В перспективі передбачено розширення: виведення QR-коду, друк чека або синхронізацію з онлайн-базою.

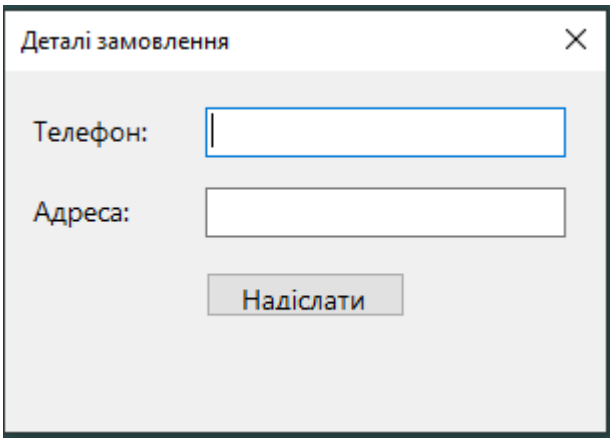

Рис. 4.9. Форма відправки замовлення

Весь застосунок побудовано на архітектурі, що дозволяє легко модифікувати список інгредієнтів, змінювати базову графіку або додавати нові типи страв. Розділення логіки (формування графічної моделі, обробка кнопок, обчислення вартості, збереження замовлення) спрощує підтримку й масштабування проєкту.

Таким чином, десктопний застосунок виступає як ефективне доповнення до веб-сайту, орієнтоване на локальні інтерактивні точки замовлення. Він поєднує в собі наочність, гнучкість та швидкодію, а також дозволяє користувачам самостійно формувати бажану страву у візуальному режимі з урахуванням власних вподобань.

Висновки до розділу 4

У межах четвертого розділу було безпосередньо реалізовано інформаційну систему, що складається з двох ключових компонентів: веб-сайту для оформлення онлайн-замовлень та десктопного застосунку-конструктора. Детально обґрунтовано вибір технологій розробки, таких як HTML, CSS, JavaScript, PHP та MySQL, що забезпечили повну функціональність веб-платформи, включаючи каталог товарів, кошик, реєстрацію користувачів, оформлення замовлення, пошук і особистий кабінет. Акцент було зроблено на інтуїтивний інтерфейс користувача з адаптивною версткою, що гарантує комфортну роботу на будь-якому пристрої.

Крім того, у розділі було представлено візуальний десктопний застосунок, реалізований засобами C# та OpenGL. Він дозволяє формувати індивідуальну піцу або бургер шляхом вибору соусу, інгредієнтів та розміру, з інтерактивною візуалізацією й підрахунком вартості. Завдяки простій формі оформлення замовлення, програма може бути використана як локальний інструмент у точках продажу.

ВИСНОВКИ

У межах виконання дипломної роботи було спроєктовано та реалізовано інформаційну систему для продажу фаст-фуду з можливістю доставки, яка включає в себе функціональний веб-сайт та візуальний десктопний застосунок. Робота охопила повний цикл розробки: від аналізу вимог і дослідження аналогічних платформ — до проектування бази даних, створення користувацького інтерфейсу, логіки обробки замовлень та технічного впровадження.

В результаті аналізу існуючих рішень було визначено основні потреби користувачів та типові проблеми (перевантаження інтерфейсу, складність пошуку товарів, обмежений візуальний зворотний зв’язок), що дало змогу сформувати цільову структуру функціоналу. Базуючись на цьому, було обґрунтовано вибір архітектури та стека технологій: PHP, JavaScript, HTML/CSS, MySQL для сайту та C# з OpenGL — для десктопного компонента.

Базу даних побудовано на основі дев’яти взаємопов’язаних таблиць з підтримкою реляційної цілісності, що дозволяє зберігати інформацію про користувачів, товари, замовлення, відгуки, статуси та оплати. Веб-інтерфейс реалізує повноцінну взаємодію з клієнтом — від перегляду асортименту до реєстрації, фільтрації, формування кошика та оформлення замовлення. Десктопна частина, у свою чергу, дозволяє формувати персоналізовані піци або бургери із візуалізацією в режимі реального часу, що підвищує зручність та залучення користувача.

Розроблений програмний продукт може бути використаний як самостійний ресурс для малого бізнесу в галузі доставки їжі, так і як частина більшого комплексу — з можливістю масштабування, підключення мобільного застосунку чи адміністративної CRM-системи. Структура коду, архітектура бази та загальний підхід до побудови інтерфейсу відповідають сучасним вимогам до розробки клієнтоорієнтованих сервісів.

Таким чином, поставлена мета дипломної роботи — створення зручної, ефективної та функціональної системи для продажу фаст-фуду з підтримкою доставки — досягнута повністю. Отримані результати мають практичне значення та можуть бути розширені в подальшому для реального комерційного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мельник Р**.** Програмування веб-застосувань (фронт-енд та бек-енд). Навчальний посібник. — Львів : Львівська політехніка, 2018. — 248 с.
2. Юрчак І.Ю., Гузинець Н.В. Базові засади веб-розробки. Навчальний посібник. — Львів : Магнолія, 2023. — 180 с.
3. Пасічник В.В., Пасічник О.В. Веб-дизайн. Підручник. — Львів : Магнолія, 2024. — 520 с.profbook.com.ua
4. Пастернак І.І., Костик А.Т. Інструментальні засоби веб-технологій. Навчальний посібник. — Львів : Магнолія, 2024. — 197 с.
5. Шевченко В. Графічне програмування засобами OpenGL. Навчальний посібник. — Київ : НТУУ «КПІ», 2020. — 124 с.
6. Гайна Г.А. Основи проектування баз даних. Навчальний посібник. — Київ : Кондор, 2018. — 204 с.yakaboo.uayakaboo.ua
7. Балик Н.Р., Мандзюк В.І. Бази даних MySQL. Навчальний посібник. — Тернопіль : Навчальна книга – Богдан, 2010. — 160 с.
8. Круґ С. Не змушуйте мене думати! Практичний посібник. — Київ : ArtHuss, 2024. — 198 с.
9. Норман Д.А. Дизайн звичних речей. Монографія. — Харків : Клуб Сімейного Дозвілля, 2019. — 320 с.
10. Ґотельф Д., Сейден Дж. Lean UX: Створення класних продуктів із командами Agile. Практичний посібник. — Київ : ArtHuss, 2024. — 206 с.
11. Ґрівер Т. Формулювання дизайнерських рішень: Як спілкуватися зі стейкхолдерами, зберігати здоровий глузд і забезпечувати найкращий користувацький досвід. Практичний посібник. — Київ : ArtHuss, 2024. — 264 с.
12. Лавріщева К.М. Програмна інженерія. Підручник. — Київ : Академперіодика, 2008. — 319 с.

13. Трофименко О.Г., Манаков С.Ю., Ларін Д.Г. Основи програмної інженерії. Навчально-методичний посібник. — Одеса : Фенікс, 2022. — 197 с.
14. Shreiner D., Sellers G., Kessenich J., Licea-Kane B. OpenGL Programming Guide (9th Edition). Довідник. — Boston : Addison-Wesley, 2017. — 984 p.
15. Pressman R.S., Maxim B.R. Software Engineering: A Practitioner’s Approach. Підручник. — New York : McGraw-Hill, 2015. — 944 p.

ДОДАТКИ

Додаток А

Головна форма програми «PizzaBuilder»

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Net.Http;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Windows.Forms;
using OpenTK;
using OpenTK.Graphics.OpenGL;

namespace PizzaBuilder
{
    public class IngredientItem
    {
        public string Type { get; set; }
        public float X { get; set; }
        public float Y { get; set; }
        public float Rotation { get; set; }
    }

    public partial class MainForm : Form
    {
        private GLControl glControl;
        private System.Windows.Forms.Timer renderTimer;

        private string selectedSauce = "red";
        private string selectedSize = "medium";
        private List<IngredientItem> ingredientItems = new
List<IngredientItem>();
        private Label priceLabel;
        private ComboBox sizeComboBox;

        private readonly Dictionary<string, decimal> prices = new
Dictionary<string, decimal>
        {
            {"cheese", 5},
            {"mushrooms", 4},
            {"pepperoni", 6},
            {"olives", 3},
            {"bacon", 7},
            {"tomato", 4},
            {"onion", 2},
            {"cucumber", 3},
            {"sauce_red", 5},
            {"sauce_white", 6},
            {"base", 15},
            {"size_small", -3},
            {"size_medium", 0},
            {"size_large", 7}
        };
```

```
        public MainForm()
        {
            InitializeComponent();
            InitializeUI();
            InitializeOpenGL();
        }

        private void InitializeUI()
        {
            var panel = new FlowLayoutPanel
            {
                Dock = DockStyle.Left,
                Width = 200,
                Padding = new Padding(10),
                FlowDirection = FlowDirection.TopDown,
                AutoScroll = true,
                AutoSize = true,
                WrapContents = false
            };

            var sizeLabel = new Label { Text = "Розмір піци:", AutoSize =
true };
            sizeComboBox = new ComboBox { Width = 160, DropDownStyle =
ComboBoxStyle.DropDownList };
            sizeComboBox.Items.AddRange(new[]   {   "Мала",   "Середня",
"Велика" });
            sizeComboBox.SelectedIndex = 1;
            sizeComboBox.SelectedIndexChanged += (s, e) =>
            {
                selectedSize = sizeComboBox.SelectedIndex switch
                {
                    0 => "small",
                    1 => "medium",
                    2 => "large",
                    _ => "medium"
                };
                if  (string.IsNullOrEmpty(selectedSize))  selectedSize  =
"medium";
                UpdatePrice();
            };

            var labelSauce = new Label { Text = "Соус:", AutoSize = true,
Margin = new Padding(3, 10, 3, 3) };
            var rbRed = new RadioButton { Text = "Томатний", Checked =
true, AutoSize = true };
            var rbWhite = new RadioButton { Text = "Вершковий", AutoSize
= true };
            rbRed.CheckedChanged  +=  (s,  e)  =>  {  if  (rbRed.Checked)
selectedSauce = "red"; UpdatePrice(); };
            rbWhite.CheckedChanged += (s, e) => { if (rbWhite.Checked)
selectedSauce = "white"; UpdatePrice(); };

            panel.Controls.Add(sizeLabel);
```

```
            panel.Controls.Add(sizeComboBox);
            panel.Controls.Add(labelSauce);
            panel.Controls.Add(rbRed);
            panel.Controls.Add(rbWhite);

            AddIngredientButton(panel, "Сир", "cheese", 20);
            AddIngredientButton(panel, "Гриби", "mushrooms", 10);
            AddIngredientButton(panel, "Пепероні", "pepperoni", 10);
            AddIngredientButton(panel, "Оливки", "olives", 12);
            AddIngredientButton(panel, "Бекон", "bacon", 8);
            AddIngredientButton(panel, "Томати", "tomato", 10);
            AddIngredientButton(panel, "Цибуля", "onion", 10);
            AddIngredientButton(panel, "Огірки", "cucumber", 10);

            var clearButton = new Button { Text = "Очистити піцу", Width
= 160, Margin = new Padding(0, 10, 0, 0) };
            clearButton.Click += (s, e) =>
            {
                ingredientItems.Clear();
                UpdatePrice();
                glControl.Invalidate();
            };
            panel.Controls.Add(clearButton);

            priceLabel = new Label { Text = "Ціна: 0 грн", AutoSize = true,
Margin = new Padding(3, 10, 3, 3), Font = new Font("Segoe UI", 10,
FontStyle.Bold) };
            selectedSize = "medium";
            panel.Controls.Add(priceLabel);
            var submitButton = new Button { Text = "Оформити замовлення",
Width = 160, Margin = new Padding(0, 10, 0, 0) };
            submitButton.Click += async (s, e) =>
            {
                using var form = new SubmitForm(this);
                if (form.ShowDialog() == DialogResult.OK)
                {
                    await                SubmitOrderAsync(form.UserPhone,
form.UserAddress);
                }
            };
            panel.Controls.Add(submitButton);
            Controls.Add(panel);

            // Оновлення ціни після початкового створення інтерфейсу
            UpdatePrice();
        }

        private void AddIngredientButton(FlowLayoutPanel panel, string
label, string type, int count)
        {
            var btn = new Button { Text = label, Width = 160 };
            btn.Click += (s, e) =>
            {
                AddIngredientPortion(type, count);
```

```
            UpdatePrice();
        };
        panel.Controls.Add(btn);
    }

    private void UpdatePrice()
    {
        decimal total = 0;
        foreach (var item in ingredientItems)
        {
            if (prices.TryGetValue(item.Type, out decimal value))
                total += value / 10m;
        }

        // додаємо ціну основи та соусу
        total += prices["base"];
        if (selectedSauce == "red") total += prices["sauce_red"];
        if (selectedSauce == "white") total += prices["sauce_white"];
        total += prices[$"size_{selectedSize}"];

        priceLabel.Text = $"Ціна: {Math.Round(total)} грн";
    }

    private void InitializeOpenGL()
    {
        glControl = new GLControl();
        glControl.Dock = DockStyle.Fill;
        glControl.Load += GlControl_Load;
        glControl.Paint += GlControl_Paint;
        Controls.Add(glControl);

        renderTimer = new System.Windows.Forms.Timer();
        renderTimer.Interval = 16;
        renderTimer.Tick += (s, e) => glControl.Invalidate();
        renderTimer.Start();
    }

    private void GlControl_Load(object sender, EventArgs e)
    {
        GL.ClearColor(Color.DarkSlateGray);
    }

    private void GlControl_Paint(object sender, PaintEventArgs e)
    {
        int width = glControl.Width;
        int height = glControl.Height;

        GL.Viewport(0, 0, width, height);
        GL.Clear(ClearBufferMask.ColorBufferBit);

        GL.MatrixMode(MatrixMode.Projection);
        GL.LoadIdentity();
        float aspect = (float)width / height;
        if (aspect >= 1)
```

```
                GL.Ortho(-aspect, aspect, -1, 1, -1, 1);
            else
                GL.Ortho(-1, 1, -1 / aspect, 1 / aspect, -1, 1);
            GL.MatrixMode(MatrixMode.Modelview);
            GL.LoadIdentity();

            DrawPizzaBase();
            DrawSauce();
            DrawIngredients();

            glControl.SwapBuffers();
        }

        private void DrawPizzaBase()
        {
            GL.Color3(Color.Goldenrod);
            DrawCircle(0f, 0f, 0.8f);
        }

        private void DrawSauce()
        {
            if (selectedSauce == "red")
                GL.Color3(Color.IndianRed);
            else if (selectedSauce == "white")
                GL.Color3(Color.Beige);
            else
                return;

            DrawCircle(0f, 0f, 0.75f);
        }

        private void DrawIngredients()
        {
            foreach (var item in ingredientItems)
            {
                switch (item.Type)
                {
                    case   "cheese":   DrawCheeseSlice(item.X,   item.Y,
item.Rotation); break;
                    case "pepperoni": DrawPepperoniSlice(item.X, item.Y);
break;
                    case  "mushrooms":  DrawMushroomShape(item.X,  item.Y,
item.Rotation); break;
                    case "olives": DrawOliveShape(item.X, item.Y); break;
                    case    "bacon":    DrawBaconSlice(item.X,    item.Y,
item.Rotation); break;
                    case   "tomato":   DrawTomatoSlice(item.X,   item.Y);
break;
                    case "onion": DrawOnionRing(item.X, item.Y); break;
                    case  "cucumber":  DrawCucumberSlice(item.X,  item.Y);
break;
                }
            }
        }
```

```
private void AddIngredientPortion(string type, int count)
{
    Random rand = new Random();

    for (int i = 0; i < count; i++)
    {
        float angle = (float)(rand.NextDouble() * 2 * Math.PI);
        float distance = 0.15f + (float)rand.NextDouble() * 0.45f;
        float x = (float)(Math.Cos(angle) * distance);
        float y = (float)(Math.Sin(angle) * distance);
        float rot = (float)(rand.NextDouble() * 360);

        ingredientItems.Add(new IngredientItem
        {
            Type = type,
            X = x,
            Y = y,
            Rotation = rot
        });
    }
}

// ВІЗУАЛІЗАЦІЯ ІНГРЕДІЄНТІВ

private void DrawCheeseSlice(float x, float y, float rotation)
{
    GL.PushMatrix();
    GL.Translate(x, y, 0);
    GL.Rotate(rotation, 0, 0, 1);
    GL.Color3(Color.Goldenrod);
    GL.Begin(PrimitiveType.Quads);
    GL.Vertex2(-0.03f, -0.01f);
    GL.Vertex2(0.03f, -0.01f);
    GL.Vertex2(0.03f, 0.01f);
    GL.Vertex2(-0.03f, 0.01f);
    GL.End();
    GL.PopMatrix();
}

private void DrawPepperoniSlice(float x, float y)
{
    GL.Color3(Color.Maroon);
    DrawCircle(x, y, 0.06f);
    GL.Color3(Color.Firebrick);
    DrawCircle(x, y, 0.05f);
    GL.Color3(Color.BurlyWood);
    for (int i = 0; i < 3; i++)
    {
        float angle = (float)(i * 2 * Math.PI / 3);
        float dx = (float)Math.Cos(angle) * 0.025f;
        float dy = (float)Math.Sin(angle) * 0.025f;
        DrawCircle(x + dx, y + dy, 0.008f);
    }
}
```

```
private void DrawMushroomShape(float cx, float cy, float rotation)
{
    GL.PushMatrix();
    GL.Translate(cx, cy, 0);
    GL.Rotate(rotation, 0, 0, 1);
    GL.Color3(Color.SandyBrown);
    GL.Begin(PrimitiveType.TriangleFan);
    GL.Vertex2(0f, 0f);
    for (int i = 0; i <= 20; i++)
    {
        double angle = Math.PI * i / 20;
        float x = (float)(Math.Cos(angle) * 0.035f);
        float y = (float)(Math.Sin(angle) * 0.035f);
        GL.Vertex2(x, y);
    }
    GL.End();
    GL.Color3(Color.BurlyWood);
    GL.Begin(PrimitiveType.Quads);
    GL.Vertex2(-0.01f, 0f);
    GL.Vertex2(0.01f, 0f);
    GL.Vertex2(0.01f, -0.025f);
    GL.Vertex2(-0.01f, -0.025f);
    GL.End();
    GL.PopMatrix();
}

private void DrawOliveShape(float x, float y)
{
    GL.Color3(Color.Black);
    DrawCircle(x, y, 0.025f);
    GL.Color3(Color.DarkSlateGray);
    DrawCircle(x, y, 0.012f);
}

private void DrawBaconSlice(float x, float y, float rotation)
{
    GL.PushMatrix();
    GL.Translate(x, y, 0);
    GL.Rotate(rotation, 0, 0, 1);
    GL.Color3(Color.DeepPink);
    GL.Begin(PrimitiveType.Quads);
    GL.Vertex2(-0.06f, -0.015f);
    GL.Vertex2(0.06f, -0.015f);
    GL.Vertex2(0.06f, 0.015f);
    GL.Vertex2(-0.06f, 0.015f);
    GL.End();


    GL.Color3(Color.PeachPuff);
    GL.Begin(PrimitiveType.LineStrip);
    for (float i = -0.06f; i <= 0.06f; i += 0.01f)
    {
        float yOffset = (float)(Math.Sin(i * 20) * 0.003f);
        GL.Vertex2(i, yOffset);
```

```
    }
    GL.End();

    GL.PopMatrix();
}

private void DrawTomatoSlice(float x, float y)
{
    GL.Color3(Color.Red);
    DrawCircle(x, y, 0.05f);
    GL.Color3(Color.MistyRose);
    for (int i = 0; i < 5; i++)
    {
        float angle = (float)(2 * Math.PI * i / 5);
        float dx = (float)Math.Cos(angle) * 0.025f;
        float dy = (float)Math.Sin(angle) * 0.025f;
        DrawCircle(x + dx, y + dy, 0.007f);
    }
}

private void DrawOnionRing(float x, float y)
{

    GL.Color3(Color.MediumPurple);
    GL.LineWidth(2f);
    GL.Begin(PrimitiveType.LineLoop);
    int segments = 50;
    for (int i = 0; i < segments; i++)
    {
        double angle = i * 2.0 * Math.PI / segments;
        float cx = x + (float)Math.Cos(angle) * 0.045f;
        float cy = y + (float)Math.Sin(angle) * 0.045f;
        GL.Vertex2(cx, cy);
    }
    GL.End();
    GL.LineWidth(1f);
}

private void DrawCucumberSlice(float x, float y)
{
    GL.Color3(Color.ForestGreen);
    DrawCircle(x, y, 0.045f);
    GL.Color3(Color.LightGreen);
    DrawCircle(x, y, 0.035f);
}

private void DrawCircle(float cx, float cy, float radius)
{
    GL.Begin(PrimitiveType.TriangleFan);
    GL.Vertex2(cx, cy);
    int segments = 50;
    for (int i = 0; i <= segments; i++)
    {
        double angle = i * 2.0 * Math.PI / segments;
```

```
                float x = cx + (float)(Math.Cos(angle) * radius);
                float y = cy + (float)(Math.Sin(angle) * radius);
                GL.Vertex2(x, y);
            }
            GL.End();
        }
        // Метод для підрахунку повної ціни
        private decimal CalculateTotalPrice()
        {
            decimal total = 0;
            foreach (var item in ingredientItems)
            {
                if (prices.TryGetValue(item.Type, out decimal price))
                    total += price / 10m;
            }

            total += prices["base"];
            if (selectedSauce == "red") total += prices["sauce_red"];
            if (selectedSauce == "white") total += prices["sauce_white"];
            total += prices[$"size_{selectedSize}"];

            return Math.Round(total);
        }

        // Метод для надсилання замовлення
        private async Task SubmitOrderAsync(string phone, string address)
        {
            var order = new
            {
                phone = phone,
                address = address,
                sauce = selectedSauce,
                size = selectedSize,
                ingredients = ingredientItems.ConvertAll(i => i.Type),
                price = CalculateTotalPrice()
            };

            using var client = new HttpClient();
            var content = new StringContent(
                JsonSerializer.Serialize(order),
                Encoding.UTF8,
                "application/json");

            try
            {
                var response = await
client.PostAsync("http://fastfood/order_receiver.php", content);
                string responseText = await
response.Content.ReadAsStringAsync();

                if (response.IsSuccessStatusCode)
                {
                    MessageBox.Show("Замовлення успішно надіслано!");
                }
```

```
                else
                {
                    MessageBox.Show("Помилка при надсиланні
замовлення:\n" + responseText);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка: " + ex.Message);
            }
        }

    }
}
```

Додаток Б

Форма відправки замовлення додатку «PizzaBuilder»

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PizzaBuilder
{
    public class SubmitForm : Form
    {
        private TextBox phoneTextBox;
        private TextBox addressTextBox;
        private Button submitButton;

        public string UserPhone { get; private set; }
        public string UserAddress { get; private set; }

        public SubmitForm(Form parent)
        {
            Text = "Деталі замовлення";
            StartPosition = FormStartPosition.CenterParent;
            FormBorderStyle = FormBorderStyle.FixedDialog;
            MaximizeBox = false;
            MinimizeBox = false;
            ClientSize = new Size(300, 180);
            Font = new Font("Segoe UI", 10);

            Label phoneLabel = new Label
            {
                Text = "Телефон:",
                AutoSize = true,
                Location = new Point(10, 20)
            };

            phoneTextBox = new TextBox
            {
                Location = new Point(100, 18),
                Width = 180
            };

            Label addressLabel = new Label
            {
                Text = "Адреса:",
                AutoSize = true,
                Location = new Point(10, 60)
            };

            addressTextBox = new TextBox
            {
                Location = new Point(100, 58),
                Width = 180
            };

            submitButton = new Button
```

```
            {
                Text = "Надіслати",
                Location = new Point(100, 100),
                Width = 100
            };
            submitButton.Click += (s, e) => Submit();

            Controls.Add(phoneLabel);
            Controls.Add(phoneTextBox);
            Controls.Add(addressLabel);
            Controls.Add(addressTextBox);
            Controls.Add(submitButton);
        }

        private void Submit()
        {
            if (string.IsNullOrWhiteSpace(phoneTextBox.Text) ||
string.IsNullOrWhiteSpace(addressTextBox.Text))
            {
                MessageBox.Show("Будь ласка, заповніть усі поля.");
                return;
            }

            UserPhone = phoneTextBox.Text.Trim();
            UserAddress = addressTextBox.Text.Trim();
            DialogResult = DialogResult.OK;
            Close();
        }
    }
```

Додаток В

Header сайту

```
<?php
session_start();
include 'includes/db.php';

// Розрахунок кількості та суми кошика
$cart_count = 0;
$cart_total = 0;
if (isset($_SESSION['cart'])) {
    foreach ($_SESSION['cart'] as $item) {
        $cart_count += $item['quantity'];
        $cart_total += $item['quantity'] * $item['price'];
    }
}
?>
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>FastFood</title>
  <link rel="stylesheet" href="/assets/css/style.css">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <header class="site-header">
    <div class="container top-bar">
      <div class="logo"><a href="/"> FastFood</a></div>

      <form class="search-form" action="/search.php" method="GET">
        <input type="text" name="q" placeholder="Пошук страви..." />
      </form>

      <div class="header-actions">
  <a href="/track.php" class="btn-secondary">Відстежити замовлення</a>

  <?php if (isset($_SESSION['user'])): ?>
    <?php if (!empty($_SESSION['user']['is_admin'])): ?>
      <a href="/admin/index.php" class="btn-secondary"> Адмін-панель</a>
    <?php endif; ?>

    <a href="/logout.php" class="icon-link"> Вийти</a>
  <?php else: ?>
    <a href="#" class="icon-link" id="open-auth-modal"> Профіль</a>
  <?php endif; ?>

  <a href="/cart.php" class="cart">
     <span id="cart-count"><?php echo $cart_count; ?></span> /
    <span id="cart-total"><?php echo number_format($cart_total, 2); ?>
грн</span>
  </a>
</div>

    </div>
```

```
    <nav class="category-nav">
      <div class="container category-list">
        <?php
        $cat_result = $conn->query("SELECT * FROM categories ORDER BY
name");
        while ($cat = $cat_result->fetch_assoc()):
        ?>
          <a href="/index.php?category=<?php echo $cat['id']; ?>">
            <?php echo htmlspecialchars($cat['name']); ?>
          </a>
        <?php endwhile; ?>
      </div>
    </nav>
  </header>

  <!-- Модальне вікно логіну/реєстрації -->
  <div id="auth-modal" class="auth-modal hidden">
    <div class="auth-modal-content">
      <span class="auth-close">&times;</span>

      <!-- Вхід -->
      <form id="login-form" method="post" action="/login.php"
class="auth-form">
        <h2>Вхід</h2>
        <div class="form-errors"></div>
        <input type="email" name="email" placeholder="Email" required>
        <input type="password" name="password" placeholder="Пароль"
required>
        <button type="submit" class="btn-primary">Увійти</button>
        <p>Немає акаунту? <a href="#" id="show-
register">Зареєструватись</a></p>
      </form>

      <!-- Реєстрація -->
      <form id="register-form" method="post" action="/register.php"
class="auth-form hidden" novalidate>
        <h2>Реєстрація</h2>
        <div class="form-errors"></div>
        <input type="text" name="name" placeholder="Ім'я">
        <input type="email" name="email" placeholder="Email">
        <input type="password" name="password" placeholder="Пароль">
        <button type="submit" class="btn-
primary">Зареєструватись</button>
        <p>Вже маєте акаунт? <a href="#" id="show-login">Увійти</a></p>
      </form>
    </div>
  </div>

  <main class="main-content">
```

Додаток Г

Footer сайту

```
</main>
<footer class="site-footer">
  <div class="container">
    <p>&copy; <?php echo date('Y'); ?> FastFood. Всі права захищені.</p>
  </div>
</footer>

<script>
// Кошик: логіка кнопки "Додати до кошика"
document.querySelectorAll('.card').forEach(card => {
  const addBtn = card.querySelector('.add-to-cart');
  const qtyBox = card.querySelector('.quantity-selector');
  const minusBtn = card.querySelector('.qty-minus');
  const plusBtn = card.querySelector('.qty-plus');
  const qtyVal = card.querySelector('.qty-value');

  let quantity = 1;

  addBtn.addEventListener('click', () => {
    addBtn.style.display = 'none';
    qtyBox.style.display = 'flex';
  });

  plusBtn.addEventListener('click', () => {
    quantity++;
    qtyVal.textContent = quantity;
  });

  minusBtn.addEventListener('click', () => {
    if (quantity > 1) {
      quantity--;
      qtyVal.textContent = quantity;
    } else {
      qtyBox.style.display = 'none';
      addBtn.style.display = 'inline-block';
      quantity = 1;
      qtyVal.textContent = quantity;
    }
  });
});
</script>

<script>
// Логіка спливаючого вікна логіну/реєстрації
document.getElementById('open-auth-modal').addEventListener('click', (e)
=> {
  e.preventDefault();
  document.getElementById('auth-modal').classList.remove('hidden');
});

document.querySelector('.auth-close').addEventListener('click', () => {
  document.getElementById('auth-modal').classList.add('hidden');
```

```
});

document.getElementById('show-register').addEventListener('click', (e)
=> {
  e.preventDefault();
  document.getElementById('login-form').classList.add('hidden');
  document.getElementById('register-form').classList.remove('hidden');
});

document.getElementById('show-login').addEventListener('click', (e) => {
  e.preventDefault();
  document.getElementById('register-form').classList.add('hidden');
  document.getElementById('login-form').classList.remove('hidden');
});
</script>

<script>
// AJAX-реєстрація + виведення помилок
document.getElementById('register-form').addEventListener('submit',
function (e) {
  e.preventDefault();

  const form = this;
  const formData = new FormData(form);

  // Видалити попередні помилки
  form.querySelectorAll('.error-text').forEach(el => el.remove());

  fetch('/register.php', {
    method: 'POST',
    body: formData
  })
  .then(res => res.json())
  .then(data => {
    if (data.success) {
      location.reload();
    } else {
      data.errors.forEach(err => {
        let field = null;
        const lower = err.toLowerCase();

        if (lower.includes("ім’я") || lower.includes("ім'я")) {
          field = form.querySelector('input[name="name"]');
        } else if (lower.includes("email") || lower.includes("пошта")) {
          field = form.querySelector('input[name="email"]');
        } else if (lower.includes("пароль")) {
          field = form.querySelector('input[name="password"]');
        }

        if (!field) {
          field = form.querySelector('input[name="password"]');
        }

        const errEl = document.createElement('div');
```

```
        errEl.classList.add('error-text');
        errEl.style.color = 'tomato';
        errEl.style.fontSize = '13px';
        errEl.style.marginTop = '4px';
        errEl.textContent = err;
        field.insertAdjacentElement('afterend', errEl);
      });
    }
  });
});
</script>

</body>
</html>
```

Додаток Д

Сторінка окремого товару

```
<?php
session_start();
include '../includes/db.php';

if (!isset($_SESSION['user']) || empty($_SESSION['user']['is_admin'])) {
    header("Location: /");
    exit;
}

// Обробка видалення
if (isset($_GET['delete'])) {
    $del_id = intval($_GET['delete']);
    $stmt = $conn->prepare("DELETE FROM products WHERE id = ?");
    $stmt->bind_param("i", $del_id);
    $stmt->execute();
    $stmt->close();
    header("Location: products.php?deleted=1");
    exit;
}

// Обробка додавання товару
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = trim($_POST['name']);
    $desc = trim($_POST['description']);
    $price = floatval($_POST['price']);
    $cat_id = intval($_POST['category_id']);
    $img = $_FILES['image']['name'];
    $img_tmp = $_FILES['image']['tmp_name'];

    if ($name && $desc && $price > 0 && $img && $cat_id) {
        $target_path = '/uploads/' . basename($img);
        move_uploaded_file($img_tmp, $_SERVER['DOCUMENT_ROOT'] .
$target_path);

        $stmt = $conn->prepare("INSERT INTO products (name, description,
price, image, category_id, is_available) VALUES (?, ?, ?, ?, ?, 1)");
        $stmt->bind_param("ssdsi", $name, $desc, $price, $target_path,
$cat_id);
        $stmt->execute();
        $stmt->close();
        header("Location: products.php?added=1");
        exit;
    }
}

// Завантаження даних
$products = $conn->query("SELECT p.*, c.name AS category_name FROM
products p LEFT JOIN categories c ON p.category_id = c.id ORDER BY p.id
DESC");
$categories = $conn->query("SELECT id, name FROM categories ORDER BY
name");
?>
```

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Керування товарами</title>
  <link rel="stylesheet" href="/assets/css/style.css">
  <style>
    .admin-section { padding: 40px; }
    .admin-section h2 { margin-bottom: 20px; }
    .product-form input, .product-form textarea, .product-form select {
      display: block; margin-bottom: 15px; width: 100%; padding: 10px;
    }
    .admin-table { width: 100%; border-collapse: collapse; margin-top:
30px; }
    .admin-table th, .admin-table td {
      padding: 10px; border-bottom: 1px solid #ddd;
    }
    .admin-table th { background: #f2f2f2; }
    .btn-danger, .btn-edit {
      cursor: pointer; text-decoration: underline;
    }
    .btn-danger { color: red; }
    .btn-edit { color: blue; margin-right: 10px; }
  </style>
</head>
<body>
  <div class="container admin-section">
    <h2>Додати новий товар</h2>
    <form method="POST" class="product-form" enctype="multipart/form-
data">
      <input type="text" name="name" placeholder="Назва товару"
required>
      <textarea name="description" placeholder="Опис"
required></textarea>
      <input type="number" step="0.01" name="price" placeholder="Ціна"
required>
      <input type="file" name="image" accept="image/*" required>
      <select name="category_id" required>
        <option value="">Оберіть категорію</option>
        <?php while ($cat = $categories->fetch_assoc()): ?>
          <option value="<?php echo $cat['id']; ?>"><?php echo
htmlspecialchars($cat['name']); ?></option>
        <?php endwhile; ?>
      </select>
      <button type="submit" class="btn-primary">Додати товар</button>
    </form>

    <?php if (isset($_GET['added'])): ?>
      <p style="color: green; margin-top: 10px;"> Товар успішно
додано!</p>
    <?php elseif (isset($_GET['deleted'])): ?>
      <p style="color: red; margin-top: 10px;"> Товар видалено!</p>
    <?php endif; ?>
```

```
    <h2>Список товарів</h2>
    <table class="admin-table">
      <thead>
        <tr>
          <th>ID</th>
          <th>Назва</th>
          <th>Ціна</th>
          <th>Категорія</th>
          <th>Доступність</th>
          <th>Дії</th>
        </tr>
      </thead>
      <tbody>
        <?php while ($p = $products->fetch_assoc()): ?>
          <tr>
            <td><?php echo $p['id']; ?></td>
            <td><?php echo htmlspecialchars($p['name']); ?></td>
            <td><?php echo $p['price']; ?> грн</td>
            <td><?php echo htmlspecialchars($p['category_name']);
?></td>
            <td><?php echo $p['is_available'] ? '' : ''; ?></td>
            <td>
              <a class="btn-edit" href="edit_product.php?id=<?php echo
$p['id']; ?>">Редагувати</a>
              <a class="btn-danger" href="products.php?delete=<?php echo
$p['id']; ?>" onclick="return confirm('Ви впевнені, що хочете видалити
товар?');">Видалити</a>
            </td>
          </tr>
        <?php endwhile; ?>
      </tbody>
    </table>

    <p style="margin-top: 30px;"><a href="index.php">← Назад до адмін-
панелі</a></p>
  </div>
</body>
</html>
```