МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет
імені Дмитра Моторного
Факультет енергетики і комп'ютерних технологій

ЗАТВЕРДЖУЮ
Зав. каф. "Комп'ютерні науки"
доц._____ _____ Сергій ШАРОВ
«16»_червня__2025 р.

Пояснювальна записка

до кваліфікаційної роботи здобувача СВО Бакалавр
(ступінь вищої освіти)

на тему: «Інформаційна онлайн-система для автоматизації роботи піцерії»

52/4КНД. 9683855.000000ПЗ

Виконав: здобувач вищої освіти 4 курсу, групи 41КН
спеціальності 122 Комп'ютерні науки
за ОПП Комп'ютерні науки
(шифр і назва спеціальності та ОПП)

_____ Алла ЗАЙЦЕВА
(підпис)

Керівник ст.викл.____ _____ Ольга ЗІНОВ'ЄВА
(підпис)

Консультант доц._____Лариса БОЛТЯНСЬКА
(підпис)

Консультант доц._____Михайло ЗОРЯ
(підпис)

Нормоконтроль, ст.викл. _____Ольга ЗІНОВ'ЄВА
(підпис)

Рецензент, доц._____Альона ЧОРНА_____
(підпис)

Запоріжжя – 2025 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Факультет: Енергетики і комп'ютерних технологій
Кафедра: Комп'ютерні науки
Ступінь вищої освіти: Бакалавр
Спеціальність: 122 Комп'ютерні науки
ОПП: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри_КН
к.пед.н., доц. _____ Сергій ШАРОВ
"10" _жовтня__2024__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Зайцевій Аллі Максимівні _
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи __«Інформаційна онлайн-система для автоматизації роботи піцерії»_____
керівник проекту __Зінов'єва Ольга Геннадіївна, ст. викладач_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом університету від "30" вересня_2024_року №_452-С
2. Строк подання здобувачем вищої освіти роботи __12 червня 2025_____
3. Вихідні дані до роботи: _дані обстеження об'єкту, статистичні дані, технічне завдання на дипломне проєктування, матеріали виробничих практик, нормативні документи, науково-технічна література, електронні ресурси та ін._____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)____
___1. Опис предметної області та аналіз існуючих рішень._____
___2. Специфікація вимог до інформаційної системи._
___3. Вибір та обґрунтування технологій розробки інформаційної системи.___
___4. Дослідна експлуатація інформаційної системи._____
___5. Тестування інформаційної системи._____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)
Діаграма використання, діаграми IDEF0, розкадровка інтерфейсу, структурна схема, візуальні компоненти інтерфейсу, алгоритм роботи, інтерфейс програми
Презентація – 10 слайдів
1. Титульний слайд
2. Предмет, об'єкт, мета
3. Завдання дослідження
4. Порівняльна характеристика аналогів програмного продукту

5. Діаграма варіантів використання
6. Інструментальні засоби
7. Інтерфейс і робота інформаційної системи
8. Інтерфейс і робота інформаційної системи
9. Тестування
10. Висновок

6. Консультанти розділів кваліфікаційної роботи

Розділ/ Підрозділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
4.3	Болтянська Л.О.	15.10.2025	15.06.2025
4.4	Зоря М.В.	15.10.2025	15.06.2025

7. Дата видачі завдання___10 жовтня 2024 р. _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	1 Аналіз предметної області	03.10.2024	виконано
2	Специфікація вимог до інформаційної системи	17.10.2024	виконано
3	Проектування інформаційної системи	28.10.2024	виконано
4	Обгрунтування інструментальних засобів	18.11.2024	виконано
5	Розробка інформаційної системи	16.12.2024	виконано
6	Тестування та налагодження інформаційної системи	11.04.2025	виконано
7	Підпис керівником роботи	14.06.2025	виконано
8	Підпис завідувачем кафедри	16.06.2025	виконано

Здобувач вищої освіти _____ _Алла ЗАЙЦЕВА__
(підпис) (власне ім'я та прізвище)

Керівник кваліфікаційної роботи _____ _Ольга ЗІНОВ'ЄВА_
(підпис) (власне ім'я та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 56 с., рис. 18, табл. 8, 21 посилань.

Актуальність теми зумовлена потребою малого та середнього бізнесу в простих і водночас ефективних онлайн-рішеннях, що дозволяють вивести сервіс на новий рівень. Саме тому предметом даної дипломної роботи є розробка інформаційної онлайн-системи – веб-сайту піццерії, яка дозволяє автоматизувати ключові бізнес-процеси: управління меню, прийом замовлень, організацію доставки, облік клієнтів та комунікацію з ними.

Об’єкт дослідження: процеси формування та оформлення замовлень в піцерії.

Предмет дослідження: інформаційна онлайн-система для автоматизації роботи піцерії.

Мета роботи: Розробка зручного та інтуїтивно зрозумілого веб-застосунку для полегшення процесу оформлення замовлення в піцерії, створення веб-орієнтованої інформаційної системи, що забезпечує автоматизацію основних функцій піццерії, підвищуючи її операційну ефективність та якість клієнтського обслуговування

Завдання дослідження:

1. Провести аналіз предметної області.
2. Розглянути існуючі програмні рішення, що забезпечують автоматизацію роботи піцерії.
3. Вибрати інструментальні засоби для реалізації.
4. Розробити технічне завдання для створення інформаційної онлайн-системи для автоматизації роботи піцерії.
5. На основі технічного завдання розробити інформаційну онлайн-систему для автоматизації роботи піцерії.
6. Провести тестування розробленої інформаційної онлайн-системи та оцінити її ефективність.

Основні результати. В результаті роботи було розроблено інформаційну систему для автоматизації роботи піцерії засобами Svelte, SvelteKit, Drizzle та MySQL, з використанням Svelte store.

Практичне значення роботи полягає в розробці інформаційної онлайн-системи для автоматизації роботи піцерії, яка допоможе підвищити її конкурентоспроможність і розширити клієнтську базу

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, ОНЛАЙН-ЗАМОВЛЕННЯ, АВТОМАТИЗАЦІЯ, JAVASCRIPT, SVELTE, SVELTEKIT

SUMMARY

Bachelor's qualification work: 56 p., fig. 18, tab. 8, references 21.

The relevance of the topic is due to the need of small and medium-sized businesses for simple and at the same time effective online solutions that allow taking the service to a new level. That is why the subject of this thesis is the development of an online information system - a pizzeria website, which allows you to automate key business processes: menu management, order acceptance, delivery organization, customer registration and communication with them.

Object of research: processes of forming and processing orders in a pizzeria.

Subject of research: an online information system for automating the work of a pizzeria.

Purpose of the work: Development of a convenient and intuitive web application to facilitate the process of placing an order in a pizzeria, creation of a web-oriented information system that provides automation of the main functions of a pizzeria, increasing its operational efficiency and quality of customer service

Research objectives:

1. Conduct an analysis of the subject area.

2. Consider existing software solutions that provide automation of the pizzeria.

3. Select tools for implementation.

4. Develop a technical task for creating an online information system to automate the operation of a pizzeria.

5. Based on the technical task, develop an online information system to automate the operation of a pizzeria.

6. Conduct testing of the developed online information system and evaluate its effectiveness.

Main results. As a result of the work, an information system was developed to automate the operation of a pizzeria using Svelte, SvelteKit, Drizzle and MySQL, using Svelte store.

The practical significance of the work lies in the development of an online information system to automate the work of a pizzeria, which will help increase its competitiveness and expand its customer base. Keywords: INFORMATION SYSTEM, ONLINE ORDERING, AUTOMATION, JAVASCRIPT, SVELTE, SVELTEKIT

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій суттєво змінює підходи до ведення бізнесу в різних сферах діяльності. Заклади громадського харчування, зокрема піцерії, дедалі частіше використовують цифрові інструменти для оптимізації внутрішніх процесів та покращення взаємодії з клієнтами. Поява онлайн-систем, які дозволяють автоматизувати замовлення, доставку, оплату та управління контентом, відкриває нові можливості для підвищення ефективності та конкурентоспроможності закладу.

У сучасних умовах наявність зручного, функціонального та привабливого веб-сайту для піцерії вже не є просто бажаною опцією, а перетворюється на обов’язковий інструмент успішного ведення бізнесу. Веб-сайт дозволяє спростити процес оформлення замовлень, зменшити навантаження на персонал, підвищити якість обслуговування клієнтів та забезпечити швидкий доступ до інформації про продукцію та послуги.

Актуальність теми зумовлена затребуваність якісного web-сайту для піцерії. Спосіб залучення клієнтів за допомогою web-сайту відрізняється відносно низькими витратами та великою кількістю цільової аудиторії.

Об’єктом дослідження є процеси автоматизації формування та оформлення замовлень в піцерії.

Предметом дослідження є інформаційна онлайн-система для автоматизації роботи піцерії.

Мета роботи – розробка зручного та інтуїтивно зрозумілого веб-застосунку для полегшення процесу оформлення замовлення.

Для досягнення мети дослідження були поставлені наступні завдання:

1. Провести аналіз предметної області.
2. Розглянути існуючі програмні рішення, що забезпечують автоматизацію роботи піцерії.
3. Вибрати інструментальні засоби для реалізації.

4. Розробити технічне завдання для створення інформаційної онлайн-системи для автоматизації роботи піцерії.

5. На основі технічного завдання розробити інформаційну онлайн-систему для автоматизації роботи піцерії.

6. Провести тестування розробленої інформаційної онлайн-системи та оцінити її ефективність.

Методи дослідження У ході роботи застосовано теоретичні методи, серед яких аналіз наукової та методичної літератури, порівняння наявних платформ і моделювання освітніх процесів, а також практичні методи, такі як проектування й розробка програмного забезпечення, тестування функціональних можливостей системи та експериментальна оцінка її ефективності

Практичне значення дослідження полягає в розробці інформаційної онлайн-системи для автоматизації роботи піцерії, яка допоможе підвищити її конкурентоспроможність і розширити клієнтську базу.

Структура і обсяг роботи. Кваліфікаційна робота складається з переліку зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел та додатків. У першому розділі пояснювальної записки проаналізовано предметну область розроблюваного програмного продукту. Досліджено актуальність питання створення сайтів, проаналізовано схожі проекти та представлено розроблене технічне завдання Другий розділ містить формулювання функціональних і нефункціональних вимог до системи, а також опис концептуальної моделі застосунку. У третьому розділі обґрунтовано вибір мови програмування та інструментів розробки, а також описано архітектуру системи. Четвертий розділ присвячено реалізації інформаційної системи, її тестуванню та оцінці ефективності розробленої системи. Загальний обсяг роботи становить 51 сторінка.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1 Узагальнена характеристика предметної області

Сфера онлайн-доставки їжі є одним з найбільш динамічних і швидкозростаючих сегментів цифрової економіки. За даними аналітичних досліджень, глобальний ринок онлайн-доставки їжі демонструє стійкі темпи зростання та має значний потенціал для подальшого розвитку [17]. В Україні цей сегмент також активно розвивається, особливо в умовах урбанізації та зміни споживчих звичок населення [7, 8].

В часи, коли є легкий доступ до інформації, звичайно, простіше знайти потрібний заклад чи замовити послугу через інтернет. Тому створення власного веб-сайту є важливим кроком для бізнесу, тому що він є зручним інструментом для рекламування продуктів і послуг, інформування потенційних клієнтів про наявність і ціни продуктів, а також полегшує наймання працівників, показуючи доступні можливості для працевлаштування.

Піцерії, як один з найпопулярніших типів закладів швидкого харчування, становлять значну частку ринку доставки їжі. Створення ефективних онлайн-платформ для замовлення піци не лише підвищує зручність для споживачів, але й значно впливає на операційну ефективність бізнесу [15]. Розвиток цифрових технологій та зростання проникнення інтернету створили сприятливі умови для трансформації традиційних моделей ведення бізнесу в ресторанній індустрії..

Ринок онлайн-замовлень піци на сьогоднішній день представлений безліччю гравців, включаючи великі мережі, так і дрібні місцеві заклади. Конкуренція вимагає від усіх учасників адаптації до вимог споживачів, що

швидко змінюються. Застосування технологій підвищення ефективності роботи стає потребою.

На жаль, багато компаній досі не використовують можливості, які пропонує цифрова трансформація. Важливо розуміти, що наявність функціонального веб-сайту чи мобільного додатка - це лише перший крок. Головне - надати користувачам унікальний користувальницький досвід, який включає швидке оформлення замовлення, детальну інформацію про меню і якісну підтримку клієнтів.

Розробка системи, яка забезпечує необхідні функції, такі як можливість вибору індивідуальних інгредієнтів, гнучкість в оплаті та можливість відстеження доставки дозволяє вибудовувати стійкі відносини з клієнтами. З використанням аналітичних інструментів, таких як Google Analytics, компанії можуть отримувати цінну інформацію щодо поведінки користувачів та адаптувати свої пропозиції.

Ринок також швидко змінюється під впливом нових технологій. Поява технологій машинного навчання та штучного інтелекту відкриває нові обрії для створення персоналізованого досвіду. Наприклад, системи можуть аналізувати попередні замовлення та пропонувати клієнтам рекомендовані страви, ґрунтуючись на їх перевагах.

Розробка власного сайту сприяє підвищенню іміджу закладу, залученню нових клієнтів, а також покращенню обслуговування постійних відвідувачів

Через значні переваги до конкурентоспроможності, багатьом компаніям необхідно мати власний сайт. І саме такі сайти, як наведено вище, утворюють сферу електронної торгівлі, або e-commerce [11].

Розробка веб-сайтів передбачає створення потужного маркетингового інструменту, який спрямований на підвищення попиту на продукцію чи послуги. Водночас це може бути інформаційний ресурс, що забезпечує цільову аудиторію необхідною інформацією, або ж сервіс, який дозволяє користувачам отримувати певні послуги, які їх цікавлять. Важливо відзначити,

що створення якісних і функціональних сайтів є досить складним процесом, який вимагає високого рівня професіоналізму. Однак, за потреби можна розробляти простіші динамічні чи статичні сайти з обмеженою кількістю функцій, які також здатні ефективно виконувати поставлені завдання [20].

У цій кваліфікаційній роботі розглядається створення web-сайту для піцерії. Такі інтернет-ресурси зазвичай включають наступні розділи:

– перелік товарів, які пропонуються (у цьому випадку піци);

– детальний опис продукції, рекомендації щодо її використання, цінову інформацію та переваги;

– дані щодо умов доставки;

– розділ «Про нас» із інформацією про компанію та партнерів;

– контактні дані для зв’язку з представниками піцерії;

– функціонал для додавання товарів до корзини;

– можливість оформлення замовлення.

Ретельно розроблена інформаційна система здатна суттєво спростити та прискорити процес вирішення виробничих завдань. У зв'язку з цим як об'єкт для впровадження автоматизованої програми було обрано невелике кафе-піцерія, що не відноситься до великих ресторанних мереж.

У процесі дослідження актуальності проблеми було проаналізовано значну кількість сайтів закладів харчування. На основі отриманої інформації можна зробити висновок, що наявність власного веб-сайту дозволяє закладам харчування значно збільшити кількість замовлень і залучити нових клієнтів. Веб-сайт підвищує популярність закладу, що сприяє зростанню прибутку, відкриває можливості для розширення та надання ширшого спектра послуг.

1.2 Огляд і аналіз існуючих аналогів системи

З поширенням і зростанням популярності інтернету розширювалася і сфера електронної торгівлі. В деякі сайти було вкладено надзвичайно багато

зусиль і грошей. Особливості якщо вони на англійській мові. Сайти замовлення піци можуть містити інструменти, що покращують їх доступність. Наприклад, сайт Marco's Pizza (рис. 1.1) має 6 різних налаштувань під різні типи інвалідності, весь контент можна повністю кастомізувати (навіть вимкнути анімації) і декілька інших корисних функцій, а сайт Hideaway Pizza (рис.1.2) має надзвичайно зручний інтерфейс, та інформація надзвичайно легко сприймається завдяки добре розподіленому тексту, і чудовому вибору шрифтів, які гарно виділяють заголовки.

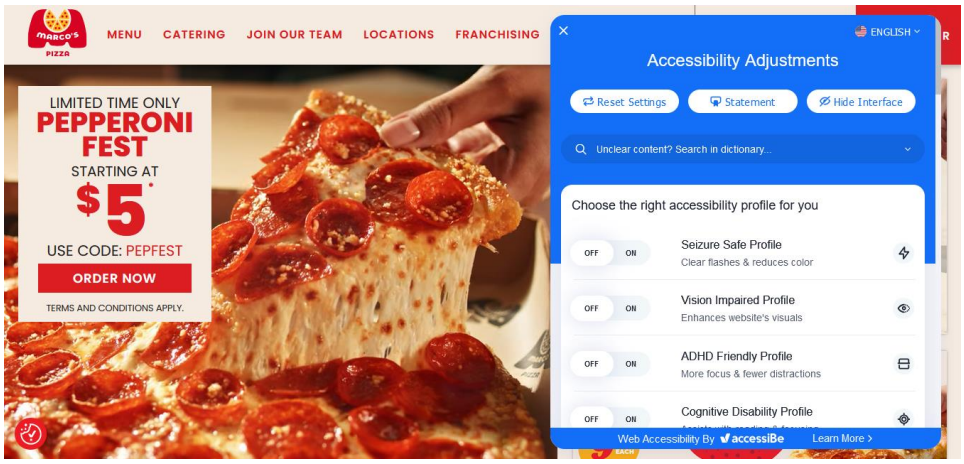

Рисунок 1.1 – Сторінка сайту Marco's Pizza

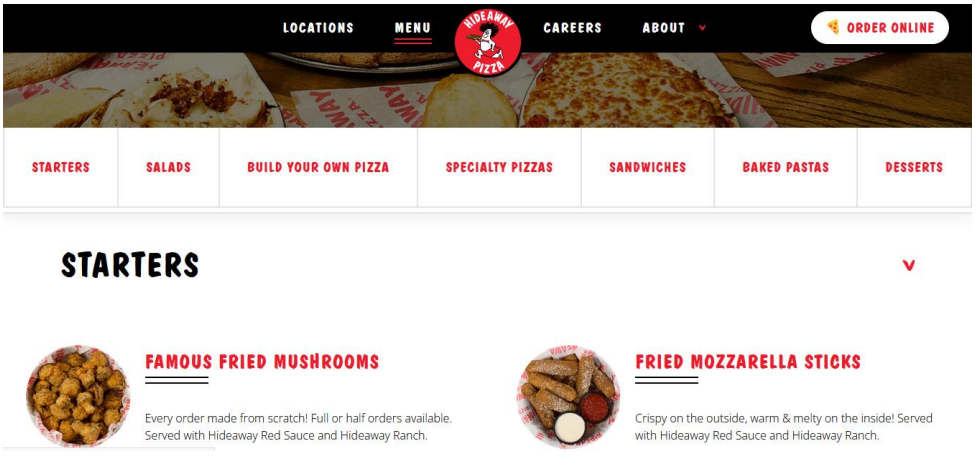

Рисунок 1.2 - Сторінка сайту Hideaway Pizza

Такі сайти здаються витворами мистецтва, і зовсім не зрозуміло, як їх можна покращити. Проте як тільки я змінила мову пошуку на українську, у результатах пошуку стали з’являтися набагато простіші сайти.

Наприклад, IKURA (рис. 1.3). Цей сайт має приємний шрифт, також є функція сортування і фільтрування, яка дає можливість знайти гострі чи вегетаріанські страви. Також є дві стрічки: одна з інформацією про компанію, вибором локації, кнопкою відстеження замовлення, пошуком, номером телефону і акаунтом, а інша стрічка має категорії їжі, сторінку з акціями і кнопку кошика, яка відображає суму і кількість продуктів у кошику

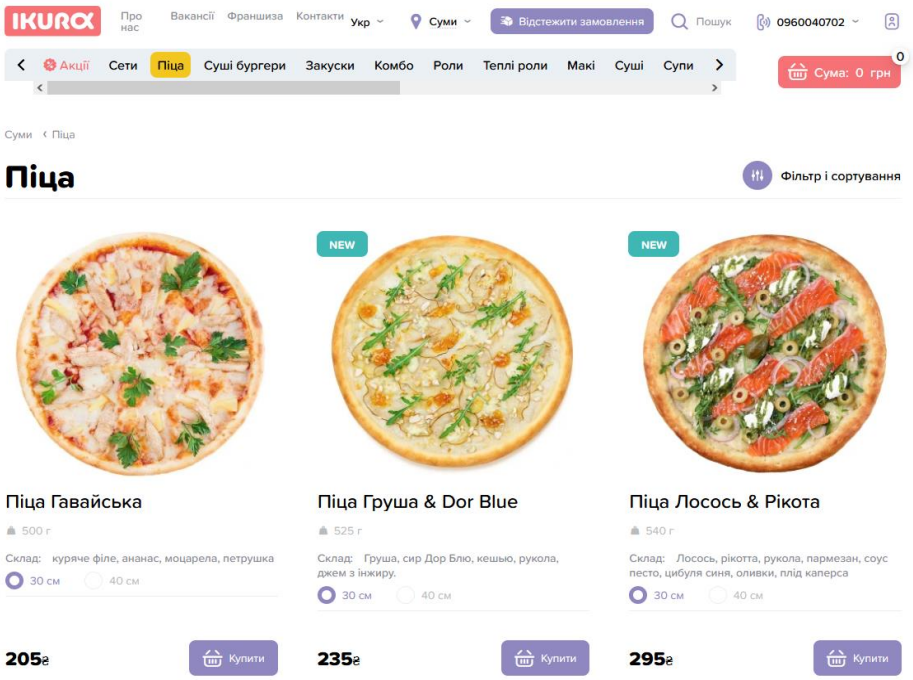

Рисунок 1.3 – Сторінка сайту IKURA

До речі, зображення кошику та індекс кількості є елементами ::before та ::after відповідно. Це псевдо елементи, які відносяться до CSS (CSS спеціальна мова (мова стилів), за допомогою якої описують вигляду документів (як і де відображати елементи веб-сторінки), написаних мовами розмітки даних. [https://css.in.ua/article/shcho-take-html_10]) і додають контент перед і після пов’язаного елементу. Вони використовуються для додавання декорацій. Наприклад, додавання позначення посилання перед кожним із них, що тим самим виділяє їх. Проте їх не рекомендується використовувати для додавання контенту, так як вони не дуже доступні для скрінрідерів. Використання ::before для додавання зображення кошику тут є непоганим вибором, тому що далі

міститься текст, який свідчить про те, що це кнопка кошику, а також тому, що додається лише зображення. Проте додавання кількості продуктів за допомогою ::after може мати непередбачувані наслідки, елемент може бути або проігнорованим, або неправильно зачитаним. [13]

Інформація на картках піци розташована у приємному для читання вигляді, і хоча частина шрифту світліша, контраст з фоном непоганий. Проте не зрозуміло, чому розміри піци мають такий світлий колір, так як вони не є додатковою інформацією, на відміну від ваги та складу. Зручно, що вибір розміру вказується візуально і виділення ціни є приємною деталлю, хоча символ гривні не дуже зрозумілий з таким шрифтом.

Картки з напоями (рис. 1.4) розташовані дуже широко (навіть на мобільних девайсах), і мають великий розмір. Деякі мають опис, проте він все ще залишає багато вільного місця.

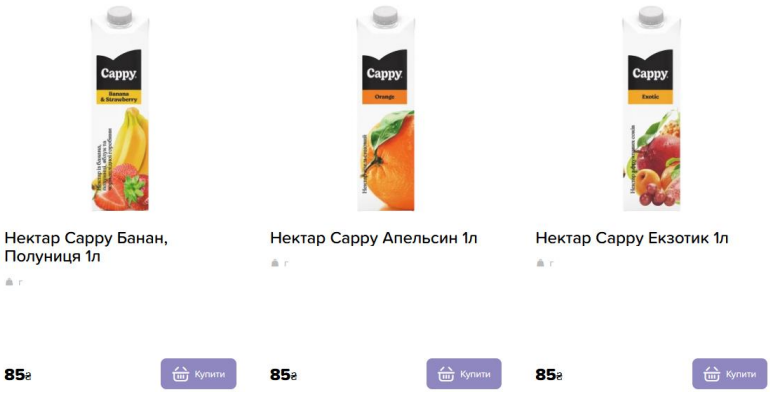

Рисунок 1.4 – Сторінка сайту IKURA

Кошик, як і у багатьох схожих сайтів, відображається справа сторінки поверх усіх елементів. Є окрема сторінка для оформлення замовлення. На вузькому екрані кнопка кошику стає плаваючою (рис. 1.5).

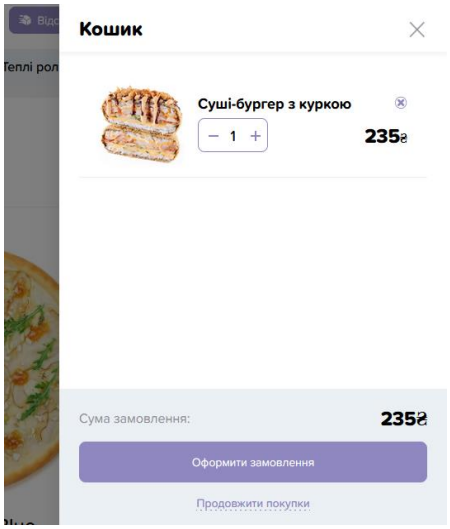

Рисунок 1.5 – Сторінка кошику на сайті IKURA

Сайт Safari (рис. 1.6) також містить дві стрічки, і одна так само залишається нагорі. Проте кошик знаходиться у першій, що доволі незручно. І поле кошику виглядає трохи дивно, і якщо його не закрити, то панель залишиться перекритою доки не прогорнути його повністю. Сума кошику позначається збоку, що доволі зручно. Кошик має власну сторінку.

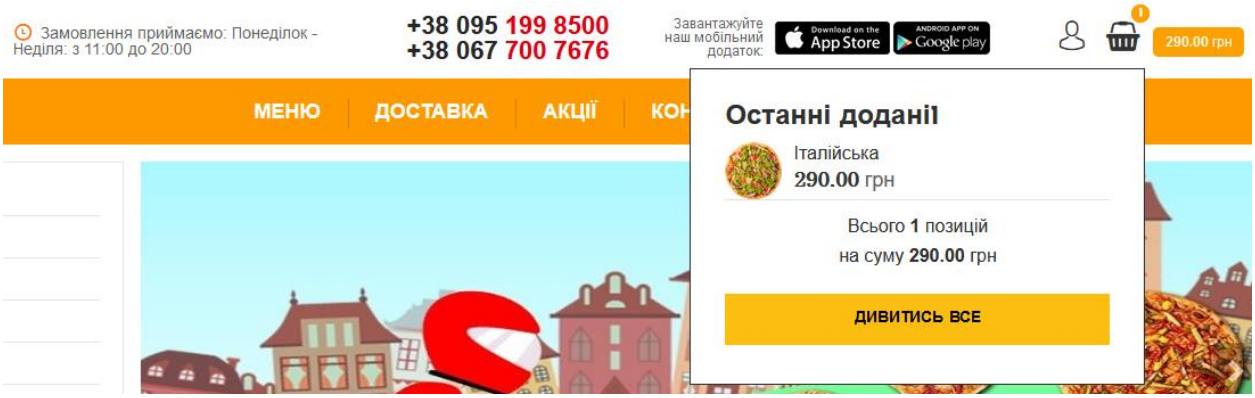

Рисунок 1.6 – Сторінка сайту Safari

Сторінка кошику (рис. 1.7) також має частину оформлення замовлення, яка розташована зправа, що значно продовжує сторінку і залишає багато порожнього місця, якщо піц небагато.

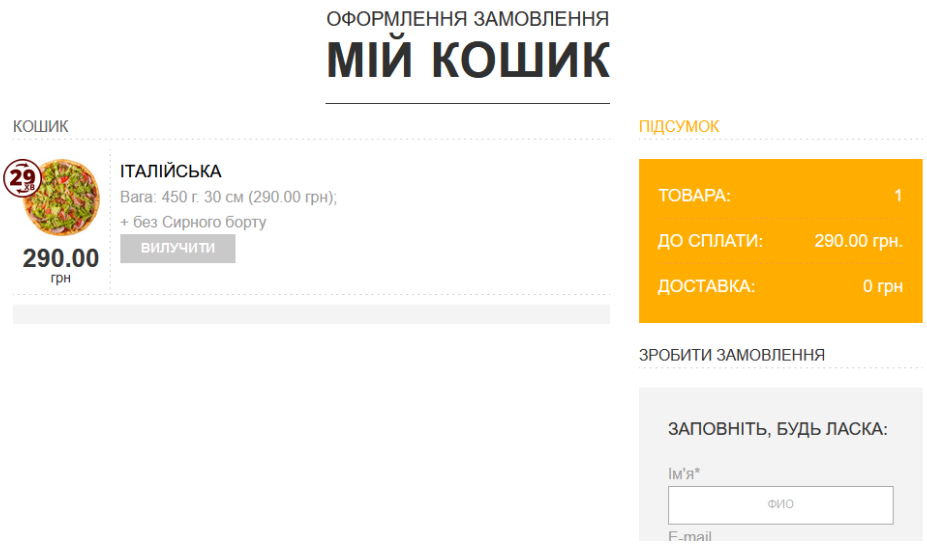

Рисунок 1.7 – Сторінка кошику на сайті Safari

Додаткові позначки на картках є фіксованими в розмірі, через що на деяких розмірах екранів вони здаються завеликими. Для піц не позначено їх склад (рис. 1.8). Шрифт є доволі великим. Вибір розміру піци і сирного борту має невелику контрастність із фоном, проте таке форматування добре виділяє вибраний варіант, і контраст не є великою проблемою завдяки розміру шрифту, а також тому, що варіанти вибору доволі типічні для кожної з піц.

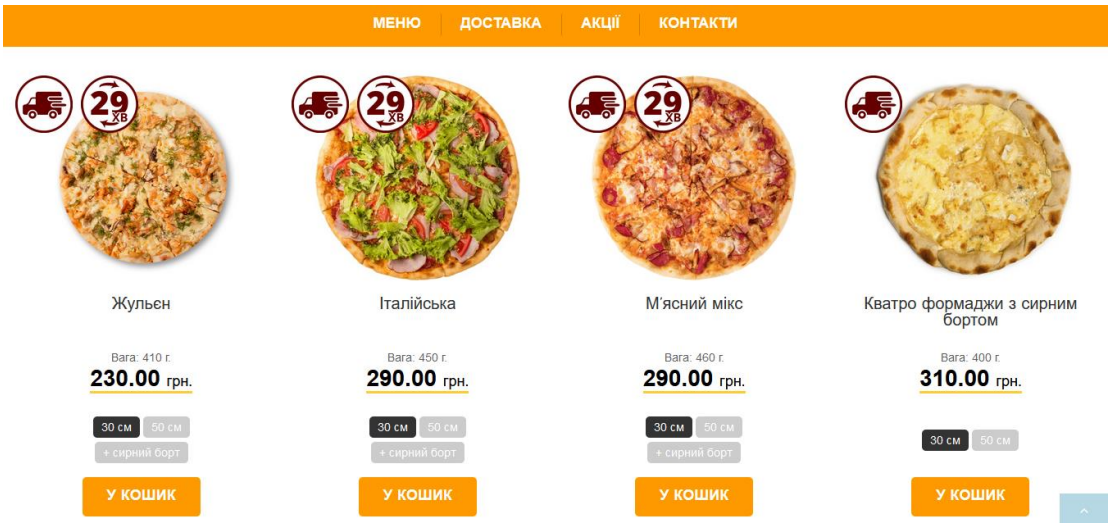

Рисунок 1.8 – Сторінка сайту Safari

Сайт має доволі незручні пропорції і на мобільних девайсах, це дуже добре видно при порівнянні тексту ваги піци, іконок у стрічці, назви розділу і позначок на картках (рис. 1.9).

Рисунок 1.9 – Відображення сайту Safari на мобільному девайсі

Сайт Kaifui (рис. 1.10) має дуже широку другу стрічку і обидві з них залишаються нагорі сторінки при переміщенні вниз по сайту. Вона закриває кнопку кошика і містить додаткову кнопку меню, яка ховає стрічки і діє як меню переповнення і заміна стрічок.

Картки піц дуже великі, а кнопка “Замовити” мала і широка. Ціна виділена добре.

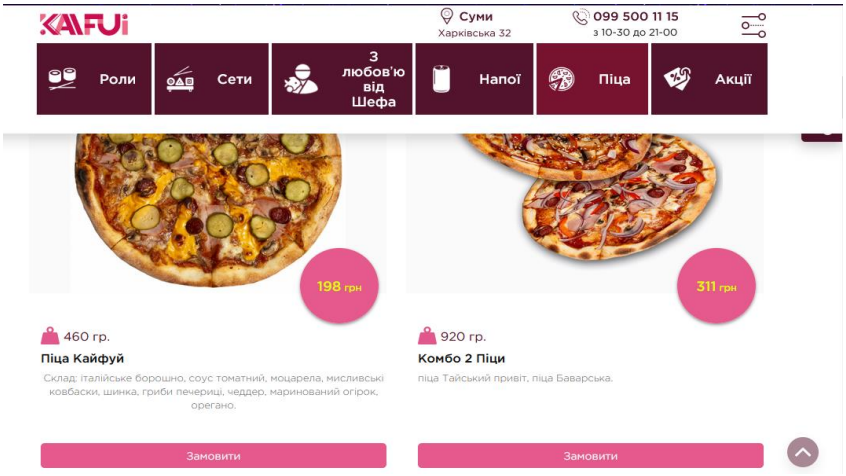

Рисунок 1.10 – Сторінка сайту Kaifui

На меншому екрані (рис. 1.11) сайт виглядає набагато краще, і кнопка кошику залишається доступною при гортанні сайту. Проте друга стрічка стає частково непомітною, і завдяки додатковому меню здається зовсім непотрібною.

Рисунок 1.11 – Відображення сайту Kaifui на меншому екрані

Як і IKURA, сайт має дуже великі розміри карток для напоїв, що також підкреслюється широкою стрічкою (рис. 1.12).

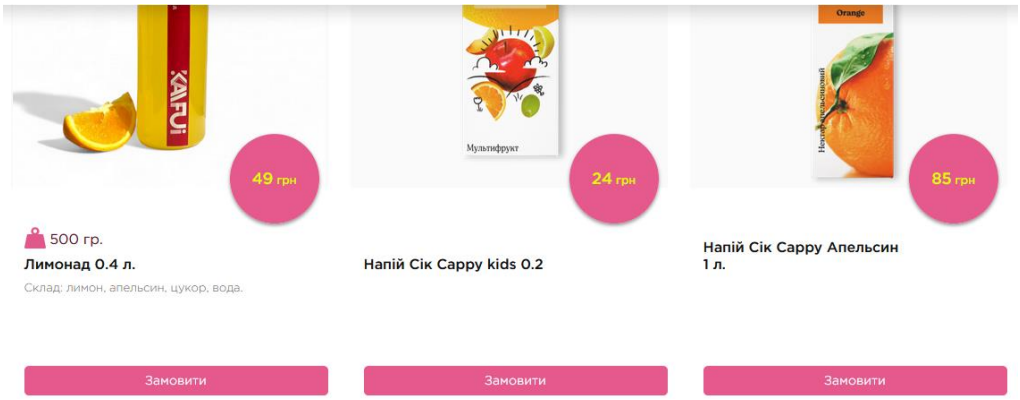

Рисунок 1.12 – Сторінка напоїв на сайті Kaifui

В таблиці 1.1 наведено порівняльні характеристики описаних вище веб-сайтів

Таблиця 1.1 – Порівняння функціоналу обраних веб-сайтів

	IKURA	Safari	Kaifui
Фільтрування і сортування	+	-	-
Вибір розміру піци	+	+	-
Додати інгредієнти	-	+	-
Відображення складу піци	+	В меню піци	+
Кнопка вибору кількості	+	+	В кошику
Вказування виду напою	Пошук і вибір	В тексті замовлення	Пошук і вибір

1.3 Розробка технічного завдання

1. Загальні визначення

Повна назва веб-продукту: веб-застосунок для автоматизації роботи піцерії.

Замовник – організація, яка замовляє створення веб-застосунку.

Виконавець – розробник, який розробляє веб-застосунок та забезпечує його розгортання.

Sveltekit – Svelte-фреймворк для створення веб-застосунків з серверним рендерингом.

MySQL – реляційна система управління базами даних.

У цьому документі наводиться повний перелік вимог до програмної реалізації та впровадження веб-застосунку. У цьому документі Замовник і Виконавець підтверджують їх згоду з нижченаведеними фактами і умовами:

Виконавець підготував і розробив цей документ, що іменується Технічне Завдання, який містить перелік вимог до виконуваних робіт.

2. Призначення розробки

2.1 Призначення і мета розробки веб-застосунку

Предметом розробки є веб-застосунок для автоматизації роботи піцерії. Веб-застосунок призначений для: відображення доступних для замовлення піц, напоїв та десертів з детальною інформацією (фотографії, частковий склад, ціна), поміщення та прибирання їх з кошику.

2.2 Характеристика об’єктів автоматизації

Об’єктами автоматизації є процеси формування і оформлення замовлень.

Веб-застосунок має забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

3. Вимоги до веб-застосунку

3.1 Вимоги до оформлення інтерфейсу

Графічний інтерфейс веб-застосунку повинен бути сучасним, адаптивним (для різних пристроїв), інтуїтивно зрозумілим та зручним у використанні. Повинні бути передбачені:

–фільтрація піц за видами (з м’ясом, без м’яса, рибна);

–сортування продуктів за ціною;

–вибір виду соку;

–зміна кількості смаку соку;

–прибирання смаку соку з кошику;

–зміна кількості продукту

–прибирання продукту з кошику

3.2 Вимоги до зберігання даних і технологій розробки

– Система керування реляційними базами даних: MySQL.
– Фронтенд та бекенд: Sveltekit (Svelte).
– Мова програмування: JavaScript.

3.3 Вимоги до програмного та технічного забезпечення

Для функціонування веб-застосунку необхідний веб-браузер на стороні користувача.

4. Стадії і етапи розробки веб-застосунку

4.1 Склад і зміст робіт із розробки веб-застосунку

Розробка веб-застосунку передбачає виконання наступних етапів:

- аналіз предметної області та збір вимог;
- проектування бази даних;
- розробка фронтенду;
- тестування та налагодження.

5. Порядок контролю і приймання

5.1 Перевірка працездатності роботи веб-застосунку

Приймання веб-застосунку до експлуатації складається з наступної перевірки:

1. Перевірка функціональності сортування та фільтрації.
2. Перевірка процесу оформлення замовлення.
3. Перевірка адаптивності інтерфейсу.

5.2 Вимоги до введення веб-застосунку в дію

Для введення веб-застосунку, розробленого в Sveltekit, в дію необхідний адаптер. Адаптер – це структурний патерн проектування, що дає змогу об’єктам із несумісними інтерфейсами працювати разом [9]. Вони можуть бути призначені для конкретної платформи (офіційні та неофіційні), або автоматичний, який автоматично встановлює і використовує правильний адаптер [20]. Інструкції для введення в дію можна знайти на сайті обраної платформи.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Специфікація функціональних та нефункціональних вимог

Функціональні вимоги – це конкретні вказівки, які описують поведінку, функції та операції, які має виконувати програмне забезпечення або система [9].

Таблиця 2.2 – Специфікація функціональних вимог

№	Назва вимоги (варіанта використання)	Атрибути вимог		
		пріоритет	складність	контакт/ виконавець
1	2	3	4	5
1	Зміна кількості продукту в кошику	Обов’язково	Низька	Користувач
2	Прибирання продукту з кошику	Обов’язково	Низька	Користувач
3	Фільтрація та сортування	Рекомендовано	Низька	Користувач
4	Прибирання смаку соку з кошику	Обов’язково	Середня	Користувач

Нефункціональні вимоги – це обмеження, накладені на систему, які визначають її атрибути якості [9].

Таблиця 2.2 – Специфікація нефункціональних вимог

№	Назва вимоги	Характеристика
1	Навчання користувачів	Веб-додаток повинен бути зрозумілим на інтуїтивному рівні
2	Масштабованість	Веб-додаток повинен бути готовим до збільшення контенту

Визначення функціональних і нефункціональних вимог допомагає виконавцю розподілити пріоритети та забезпечує, що замовник отримає додаток, що відповідає його потребам.

2.2 Концептуальна модель використання інформаційної системи

Наступним етапом після визначення вимог у створенні будь-якого сайту чи додатку є проектування. На цьому етапі визначається структура системи, її елементи і як вони взаємодіють між собою. В цьому допомагає діаграма варіантів використання. На ній зображуються діячі (актори) і варіанти використання, що розподілені між акторами. Таке зображення дозволяє зобразити, які можливості має кожен вид актору, що значно полегшує розподіл рівнів доступу, а також ця діаграма дозволяє позначити взаємодію із сервісами.

Діаграма варіантів використання (Use Case Diagram) є графічним зображенням взаємодії деякої сутності (діючої особи) і моделюється системи. Кожен варіант використання охоплює деяку функцію системи та вирішує деяку дискретну задачу, поставлену сутністю перед системою. Список усіх

Варіантів використання фактично визначає функціональні вимоги до системи. Таким чином, діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделлю системи у процесі її проектування та розробки [18].

Під час розробки діаграми варіантів використання мають бути вирішені такі задачі:

- повинні бути визначені загальні межі та контекст предметної галузі модельованої системи;
- мають бути сформульовані загальні вимоги до функціональної поведінки проектованої системи;
- повинна бути розроблена вихідна концептуальна модель системи.

Основними елементами діаграми варіантів використання є дійова особа, варіант використання, інтерфейс, стосунки, примітки.

Актор - це сутність, яка знаходиться поза межами моделі, що моделюється, і використовує її функціональні можливості для досягнення своїх цілей. Під актором можуть розуміти як конкретного виконавця, так і людину чи пристрій.

Варіант використання - це послідовність дій, що здійснюються в процесі взаємодії з актором. Його завдання полягає у визначенні завершеної частини чи сутності системи без розкриття її внутрішньої будови [4].

Для побудови діаграми варіантів використання необхідно:

1) Визначити акторів - групи дійових осіб, які працюють з системою по-різному через різні права доступу.

2) Ідентифікувати якнайбільше варіантів використання (процесів, які можуть виконувати актори).

На рисунку 2.1 наведена діаграма варіантів використання інформаційної онлайн-системи піцерії.

Дана діаграма зображує можливості користувача. Варіанти, до яких підходять стрілки “include” є розширенням більш узагальненого варіанта використання, “керування кошиком”. Цей варіант розташований для покращення перегляду діаграми і тому що похідні варіанти всі відносяться до функціонування кошику

Користувач має можливість застосувати фільтри, застосувати сортування, отримати код замовлення, та з керування кошиком додати чи прибрати продукт з кошику, очистити кошик, обрати режим перегляду соків чи їжі та змінити кількість продукту

Рисунок 2.1 – Діаграма варіантів використання

.

2.3 Розробка функціональної моделі

Сучасні підходи до управління бізнес-процесами передбачають застосування методології IDEF0 (Integration Definition for Function Modeling). Цей підхід дає змогу ефективно структурувати та аналізувати функції, а також їх взаємозв'язки у рамках організаційних процесів. Основою методології IDEF0 є принцип представлення функцій як ключових елементів бізнес-процесів. Кожна функція або діяльність системи відображається на діаграмі у вигляді прямокутного блоку з чітко визначеною метою та результатом [4].

IDEF0 – це методологія графічного опису систем і процесів діяльності організації як безлічі взаємозалежних функцій. Вона дозволяє досліджувати

функції організації, не пов'язуючи їх з об'єктами, що забезпечують їх реалізацію [5].

Графічний стандарт IDEF0 є частиною методології SADT (Structured Analysis and Design Technique - метод структурного аналізу і проєктування). Скорочення IDEF походить від ICAM Definition, де ICAM розшифровується як Integrated Computer Aided Manufacturing, що перекладається як "інтегрована комп'ютеризована система виробництва". Методологія SADT охоплює 15 різних моделей, які разом спрямовані на вивчення структури, параметрів і характеристик систем організаційно-економічного та виробничо-технічного спрямування.

Контекстна діаграма – вид IDEF0-діаграми. Це діаграма, розташована на вершині деревоподібної структури діаграм, що являє собою найзагальніший опис системи та її взаємодію із зовнішнім середовищем (як правило, тут описується основне призначення об'єкта, що моделюється).

Контекстна діаграма складається з одного блоку, що описує функцію верхнього рівня, її входи, виходи, управління та механізми, разом з формулюваннями мети моделі та точки зору, з якою будується модель.

Зліва від блоку зображено входи, справа – виходи, згори – керування (те, що регулює процес), знизу – механізми (виконавці або інструменти, що застосовуються в процесі. На рис. 2.2 зображено контекстну діаграму інформаційної системи, що узагальнено ілюструє основну функцію інформаційної системи.

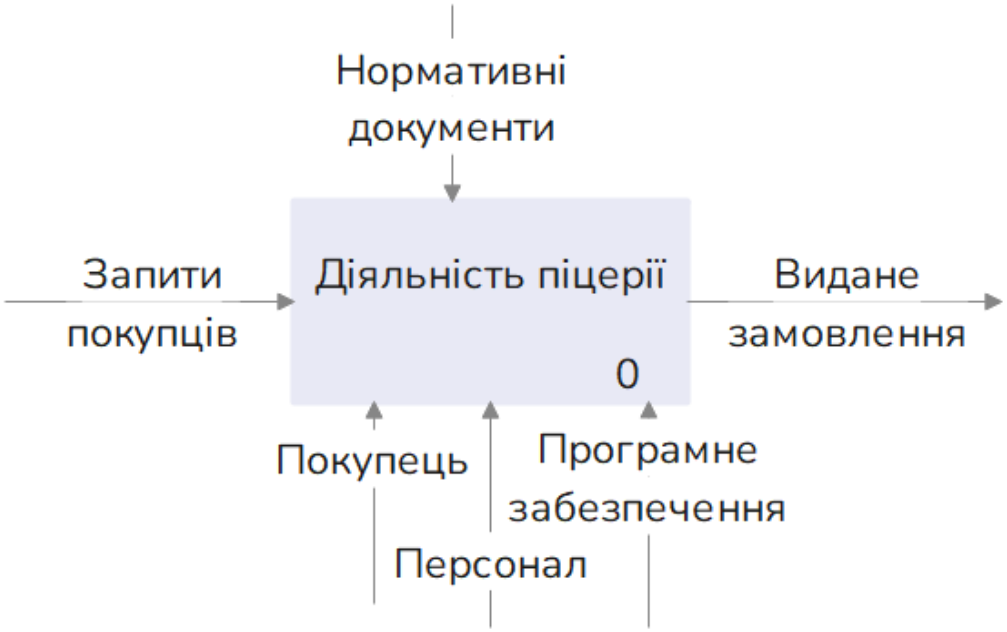

Рисунок 2.2 – Контекстна діаграма інформаційної системи

Для подальшої формалізації процесу «Діяльність піцерії» необхідно провести його декомпозицію. Результати декомпозиції наведені на рисунку 2.3.

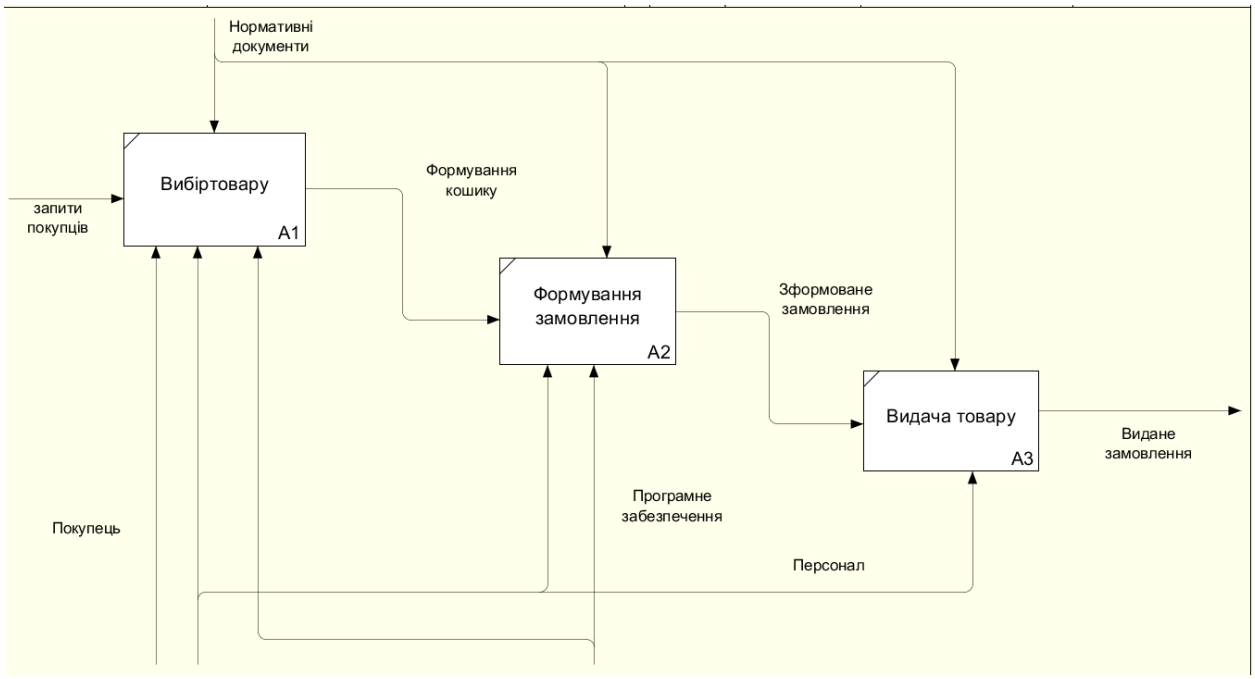

Рисунок 2.3 – Декомпозиція контекстної діаграми

Процес діяльність піцерії можна розбити на наступні підпроцеси:

- вибір товару;
- формування замовлення;
- видача замовлення.

Дана діаграма допомагає в розумінні функціонування інформаційної системи, дозволяє краще зрозуміти вхідні та вихідні параметри, зображує регулювання кожного процесу і вказує задіяних в кожному процесі особи та інструменти. Вона зображує, що необхідно для виконання кожної функції.

РОЗДІЛ 3
ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ ТА ТЕХНОЛОГІЧНИХ РІШЕНЬ

3.1 Розробка об’єктної моделі

Діаграма класів – це статична структурна діаграма, що демонструє властивості системи, класи, операції та зв’язки між об’єктами для опису структури системи.[https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/#part1] Ця діаграма зручно зображує компоненти об’єктно-орієнтованої системи і дозволяє краще розуміти залежності класів. Вона допомагає розробникам системи розуміти зміни, внесені до дизайну її структури. Діаграма класів розроблюваної системи зображена на рисунку 3.1.

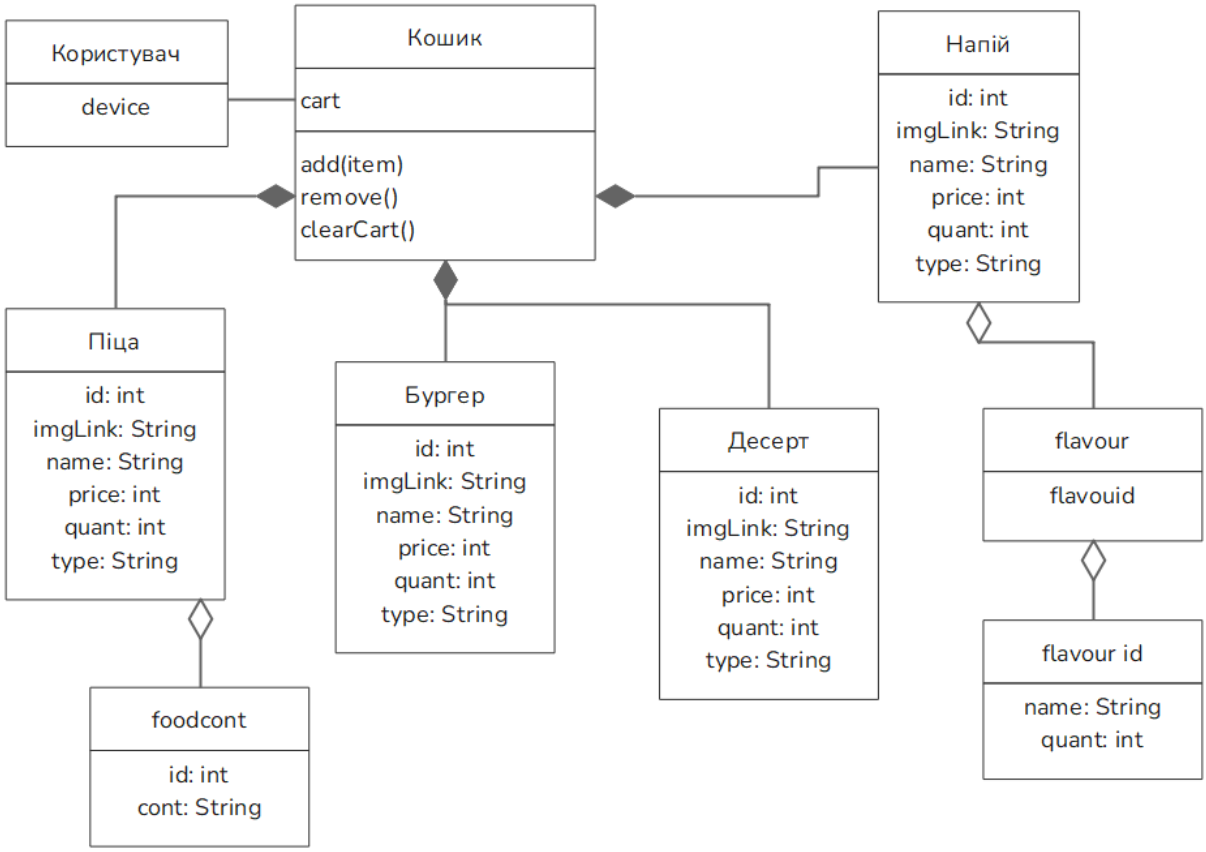

Рисунок 3.1 – Діаграма класів інформаційної системи

ER-діаграма ілюструє зв’язок сутностей у базі даних. Кожен рядок представляє стовпець бази даних і вказує його тип та довжину, якщо встановлено обмеження. Розділ “Індекси” містить інформацію про первинні та зовнішні ключі.

Первинний ключ – це унікальний ідентифікатор, що визначає запис таблиці бази даних. Зовнішні ключі використовуються для забезпечення цілісності даних при зміні записів таблиць. За допомогою їх можна вказати дії, які будуть виконуватися при зміні чи видаленні відповідних їм записів, або додаванню нових, так як вони здатні забезпечувати відповідність даних однієї таблиці іншій (наприклад, стовпець однієї таблиці може містити тільки ті записи, що вже є у відповідному стовпці іншої таблиці. Відносини допомагають підтримувати цілісність даних при запитах до таблиць. Це дозволяє включити до результату запиту додаткову інформацію. В розробці інформаційної системи ця властивість використовувалась для додавання відповідної інформації про склад піци та про смаки соків.

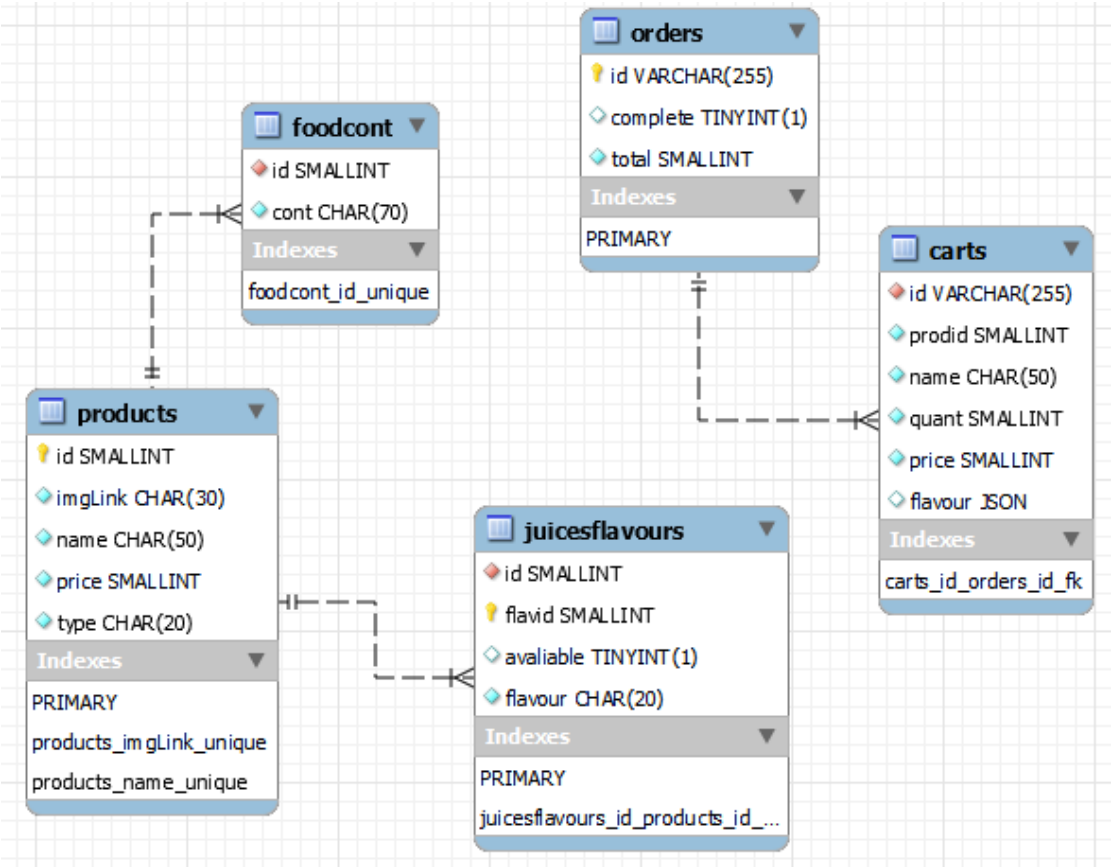

Рисунок 3.2 – ER-діаграма бази даних

3.4 Обґрунтування вибору мови програмування

Для проектування інформаційної системи були обрані такі інструменти: Svelte (SvelteKit), MySQL та Drizzle.

Svelte – фреймворк для створення інтерфейсу, що використовує компілятор для конвертування Svelte-коду в оптимізований JavaScript.

Virtual DOM – це концепція програмування, в якій ідеальне чи «віртуальне» представлення інтерфейсу користувача зберігається в пам’яті і синхронізується зі «справжнім» DOM за допомогою бібліотеки, такої як ReactDOM [19].

На відміну від більшості фреймворків, Svelte не використовує Virtual DOM для своєї реактивності, а оновлює тільки ті компоненти, які того потребують і коли вони того потребують, тому такі сайти завантажуються швидко. Також Svelte дуже легко читається, що пришвидшує розробку.

Таблиця 3.1 – Переваги та недоліки інструментів в контексті проекту

Назва	Переваги	Недоліки
1	2	3
Фреймворки розробки інтерфейсу		
Svelte	Швидко завантажує сайт і реагує на зміни в структурі, легкий в навчанні та легко читається	Має невелику екосистему, що ускладнює процес пошуку додаткових модулів та навчальних матеріалів
React	Велика кількість бібліотек та інструментів	Часто потребує додаткові бібліотеки, складніший в навчанні, ніж Vue
Vue	Легкий в навчанні та інтегруванні в існуючі проекти	Повільніше, ніж Svelte

Продовження таблиці 3.1

1	2	3
Реляційні СКБД		
MySQL	Легка у використанні, популярна	Повільніша зі складними запитами
PostgreSQL	Оптимізована для складних запитів	Складніша у навчанні, використанні і керуванні
ORM		
Drizzle	Сфокусована на SQL, “якщо ти знаєш SQL, ти знаєш Drizzle”, швидша, ніж Prisma, легше керувати самими запитами	Має менше вбудованих функцій
Prisma	Сфокусована на схемі бази даних, спрощує розробку запитів	Потребує додаткового навчання

3.5 Опис програмної реалізації

Основна структура розробленого Sveltekit-проекту складається з папок src і static (містить папки з ілюстраціями: pizza, burger, dessert, drink, home, а також логотип і спільний CSS-файл) (рис. 3.3)

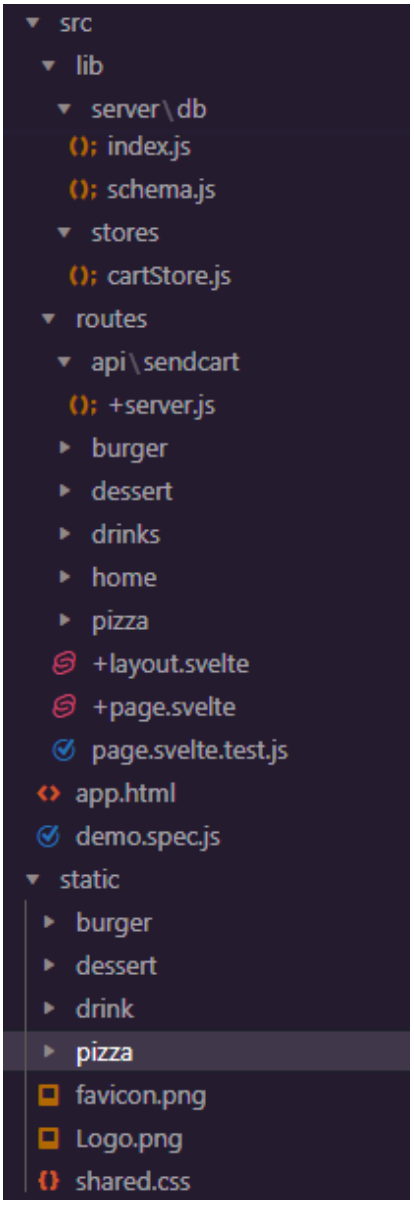

Рисунок 3.3 – Структура проекту

Кожна папка із routes містить 2 файли: +page.svelte та. Файл +page.server.js відправляє запит до бази даних, отримує результат і записує його до змінної, а +page.svelte генерує сторінку за даними, отриманими зі змінної, що доступна за допомогою руни (ключового слова) $page.data.(ім’я змінної).

Файл schema.js містить структуру таблиць бази даних, при введенні команди npx drizzle-kit push в термінал можна застосувати зміни, що були збережені у файлі. Після введення команди drizzle-kit запитає підтвердження

внесення змін, показуючи сам SQL запит. Також в цьому файлі є відносини між полями таблиць. Кожна сторінка (файл +page.server.js), яка використовує ці таблиці чи відносини має їх імпортувати.

Сторінки +page.svelte містять теги script та style, що містять відповідно код JavaScript та стилі CSS, а між ними розташований основний контент сторінки. Файл app.html містить стандартну обгортку HTML, таку як кодування документу, прив’язка до загальних стилів shared.css, іконку сторінки, імпорт шрифтів. Також вона має плейсхолдери %sveltekit.head% (містить елементи <link>, <script>, та контент <svelte:head>) та %sveltekit.body%, що містить розмітку самої сторінки.

Функція сортування за ціною використовує масив postfilter (ініціалізується створенням поверхневої копії масиву $page.data.(ім’я змінної)) для збереження порядку відсортованих результатів. Цей масив містить власні копії даних, проте для вкладених об’єктів копіює тільки їх посилання. Створення такої копії є саме те, що забезпечує реактивність, так як Svelte слідкує за зміною цих посилань.

Сторінка містить елемент fieldset, що відокремлює checkbox за групами. Це допомагає забезпечити вибір тільки одного checkbox із кожної групи. Це здійснюється функцією handleSortChange, що зазначена в параметрі onchange кожного checkbox із групи, а їх стан checked прив’язаний до змінної sort. При зміні стану checkbox масиву postfilter присвоюється значення результату сортування поверхневої копії масиву $page.data.(ім’я змінної).

Цикл {#each postfilter as p} забезпечує генерування карток піц, атрибути кожної піци можна отримати за допомогою вказування їх назви у фігурних дужках одразу після крапки та назви змінної p. Якщо ці атрибути мають атрибути, то потрібно додатково вказувати назву атрибуту. Такий синтаксис дозволено додавати у атрибути елементів HTML для забезпечення їх залежності від даних параметрів змінної і в тексті HTML для залежності його від параметрів. Також фігурні дужки можуть розраховувати результати

виразів, як при визначенні посилання на зображення. До назви зображення додається назва папки, що знаходиться в папці static, що дозволяє розподілити зображення по папкам, що відповідають їх виду.

Напої на їх сторінці згруповані за видами та розмірами, а їх картки, якщо напої мають смаки, додатково містять елементи select, що заповнюється назвами доступних смаків. У своєму скрипті сторінка визначає два об’єкти: flavou та flavouid. Одразу після визначення вони заповнюються даними із $page.data.drinksqr. Об’єкт flavou відповідає за відстеження обраних значень елементів select, і заповнення даними встановлює для кожного ідентифікатора напою ідентифікатор смаку, назви яких зберігаються в об’єкті flavouid (ключ: ідентифікатор смаку, значення: назва).

Після цього відбувається перевірка того, чи відбуваються дані дії в браузері, так як SvelteKit за замовчанням генерує HTML сторінки на сервері (SSR, Server-Side Rendering, зменшує час завантаження сторінки). Далі, щоб при зміні списку смаків, отриманих з бази даних, а також для створення запису, flavouid записується до localStorage. Це також дає можливість іншим компонентам читати назви напоїв. При зміні смаку спрацьовує функція flavouChange, якій передається ціль події (елемент, що запустив подію; у функції позначено як event.target) та значення ідентифікатору напою. Ця функція перевизначає об’єкт flavou, спочатку створюючи його поверхневу копію, а потім додаючи один з вже доступних індексів і так як всі ключі мають бути унікальними нова пара замінює стару.

Кнопка “Додати”, якщо напій має смаки, додатково передає значення об’єкту p із додаванням властивості flavid, що зберігає ідентифікатор обраного смаку.

В папці stores є файл cartStore.js, що експортує функції add(item), remove(id), clearCart() та масив cart, що є writable (надає функції запису до cart з інших компонентів). Store – об’єкт, що дозволяє доступ із реактивністю до значення. Тут було використано store для відокремлення функцій керування

кошиком від основних сторінок і компонентів. Якщо дії відбуваються в браузері, то із localStorage береться кошик. Потім йде перевірка цілісності отриманого кошику і якщо кошик не масив чи була помилка при конвертації елементу localStorage в масив, то для кошику встановлюється значення пустого масиву. Метод subscribe забезпечує оновлення cart в localStorage при внесенні змін до кошику.

Функція add викликає метод update, що спочатку читає flavouid із localStorage, а потім перевіряє, чи є предмет в кошику. Якщо він є, то map створює новий кошик, спочатку шукаючи предмет по ідентифікатору. Потім йде перевірка чи є напій соком, і якщо так, то значення предмету копіюється, а його атрибут flavour перевизначається новим об'єктом, який спочатку копіює значення об’єкту, а потім перевизначає кількість смаку з потрібним ідентифікатором. Якщо це не сік, то перевизначається тільки атрибут quant. Якщо виду соку немає в кошику, то новий створюється подібним чином. Якщо відсутній предмет не сік, то він також створюється із атрибутом quant рівному 1. Методу update повертається новий кошик, створений у результаті.

Функція remove відфільтровує кошик за ідентифікатором, наданим методу, а функція clearCart встановлює значення кошику порожній масив.

Файл +layout.svelte – це компонент стрічки та інтерфейсу кошику. Так як він розташований напряму в routes, він з’являтиметься на всіх сторінках. Для створення унікального компоненту layout потрібно розташувати його у папці із сторінкою.

Змінна cartVis є реактивною, що дозволяє ховати і показувати кошик. Класи navImage, navText та navFill були створені для зручнішого зображення основної інформації компонентів навігації. Імпорт зображення дозволяє називати змінну для його зображення, тому в масиві елементів навігації navImage має значення home. Елемент navFill присутній для переміщення кошику на праву частину стрічки. Саме меню є списком, видозміненим за допомогою CSS.

Властивість bind:checked у checkbox надає значення true чи false змінній drinksel. Якщо кошик порожній буде показано повідомлення “Ваш кошик порожній!”, інакше йде перевірка, чи вибрана опція “Напої”. Для напоїв, що мають смаки, їх тип позначається елементом details HTML. Справа кожного смаку є input number, що прив'язаний до кількості смаку. Замість ціни продукту вказується сума для кожного з них.

Для продуктів, які не є напоями, посилання на зображення визначається в залежності від типу.

РОЗДІЛ 4
ДОСЛІДНА ЕКСПЛУАТАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Інструкція користувача ІС

Для вибору виду сортування (спочатку дешевше чи спочатку дорожче) необхідно клацнути на відповідний напис чи чекбокс, і картки будуть одразу відсортовані у вибраному порядку. При виборі іншого методу сортування відмітка з чекбоксу, відмічений минулого разу, буде прибрана. Для прибирання сортування потрібно клацнути на вже відмічений чекбокс чи його напис і порядок за замовчуванням повернеться. Фільтри працюють схожим чином. При виборі фільтру будуть відображені тільки ті піци, що відповідають вказаному типу, і при прибиранні відміток так само встановлюється стан відображення за замовчуванням.

При додаванні продукту в кошик він буде відображений у меню, що з’являється при натисканні на значок кошику на стрічці вгорі. Кошик можна прибрати натисканням на хрестик у правому краю відображеного меню. За замовчуванням, кошик відображає тільки їжу. Змінити це можна натисканням на напис чи чекбокс “Напої”, і тоді кошик відображатиме лише напої. Кожен напій, який має вибір смаку, є розширюваним розділом. Кожен розділ відображує назву смаку та його кількість, що можна змінити введенням потрібної кількості чи використанням кнопок поряд. При встановленні кількості 0 для продукту він прибирається з кошику. Також кошик можна очистити, натиснувши на кнопку “Очистити кошик”. Це прибирає з нього всі елементи. Якщо кошик не показує повідомлення “Ваш кошик порожній!”, то це означає, що всі предмети в кошику мають тип, відмінний від відміченого в чекбоксі “Напої”. Замість ціни кожен продукт в кошику показує свою розраховану суму.

Кожен напій, що має різні смаки, має поле вибору. Список включає всі доступні смаки.

4.2 Тестування програмного забезпечення

Тестування – це дослідження ПЗ з метою отримання інформації про якість програмного продукту та містить перевірку на відповідність технічному завданню та вимогам. Результати тестування наведені в таблиці 4.1.

Таблиця 4.1 – Тестові ситуації

№	Тестова ситуація	Очікуваний результат	Статус
1	2	3	4
1	Відображення смаків соків	Відображені лише ті смаки соків, що є в наявності	Пройдено
2	Додавання в кошик соку	Якщо смак повторюється, то його кількість збільшується	Пройдено
3	Відображення продуктів в кошику	Соки і їжа рівномірно розподілені по кошику, елементи не залазять один на одного	Пройдено
4	Відображення напоїв в кошику	Кожен напій зі смаками містить розширюваний розділ з хоча б одним смаком та input для зміни кількості	Пройдено
5	Зміна кількості продукту чи смаку в кошику	Кількість залежить від відповідних input	Пройдено
6	Прибирання продукту з кошику	Продукт прибирається з кошику	Пройдено

Продовження таблиці 4.1

1	2	3	4
7	Очищення кошику	Кошику встановлюється значення пустого масиву	Пройдено
8	Прибирання смаку з кошику	Якщо смак має кількість 0, то його буде прибрано з кошику. Якщо напій не має смаків, то його буде прибрано з кошику	Пройдено
9	Фільтрація та сортування	При фільтрації продукти відображаються у відповідному порядку, а при сортуванні – тільки обрані категорії	Пройдено

4.3 Техніко-економічні показники ІС

Для об'єктивної оцінки ефективності впровадження інформаційної системи необхідно розрахувати основні техніко-економічні показники, які характеризують ефективність впровадження продукту.

4.3.1 Витрати, пов’язані з розробкою продукту.

До даних витрат належать витрати на оплату праці розробників з нарахуваннями, амортизаційні відрахування, витрати на матеріали, електроенергію та інші супутні витрати.

Витрати на оплату праці з нарахуваннями – складаються з витрат на основну заробітну плату, додаткову заробітну плату та нарахувань згідно чинного законодавства (22%).

Витрати на основну заробітну плату – визначаються виходячи з місячного посадового окладу або вартості 1 години роботи розробника та витрат часу, понесених на розробку даного продукту.

Розрахунок основної оплати праці здійснюється за формулою:

$$ЗПосн = \frac{ЗПміс}{ФРЧ} \cdot Тр; \quad (4.1)$$

де ЗПосн – основна заробітна плата , грн;

Витрати на оплату праці з нарахуваннями – складаються з витрат на основну заробітну плату, додаткову заробітну плату та нарахувань згідно чинного законодавства.

Витрати на основну заробітну плату – визначаються виходячи з місячного посадового окладу або вартості 1 години роботи розробника та витрат часу, понесених на розробку даного продукту. Розрахунок здійснюється за формулою:

$$ЗПосн = \frac{ЗПміс}{ФРЧ} \cdot Тр \quad (4.2)$$

Де ЗПосн – основна заробітна плата , грн;

ЗПміс – місячний посадовий оклад розробника, на підприємстві становить 15500 грн/міс;

ФРЧ – місячний фонд робочого часу, становить 176 (22 дні по 8 год), год./міс;

Тр – трудомісткість (затрати часу) виготовлення програмного продукту, год.

Трудомісткість (затрати часу) розробника на створення продукту визначається виходячи з кількості витрачених на розробку робочих днів та тривалості робочого дня:

Місячний фонд робочого часу (ФРЧ) розробника визначається виходячи з кількості робочих днів на місяць та тривалості робочого дня:

$$ФРЧ = Крдм \cdot Трз; \quad (4.2)$$

де Крд – кількість робочих днів на створення продукту, приймаємо 14 дн.

Трз – тривалість робочого дня (зміни), год.

Отже,

$$Тр = 14 \cdot 8 = 112 \text{ год}; \quad (4.3)$$

Таким чином:

$$ЗПосн = \frac{15500}{176} \cdot 112 = 9863{,}6 \text{ грн.};$$

У випадку, якщо при розробці продукту задіяні декілька працівників, розмір основної заробітної плати розраховується за кожним.

Додаткова заробітна плата – передбачає різноманітні види доплат за рівень кваліфікації, стаж роботи тощо. Розмір нарахувань здійснюється відповідно до норм прийнятих на підприємстві-замовника. Відсоток нарахувань додаткової заробітної плати коливається в межах 10-25%. На даному підприємстві додаткова оплата праці становить 20%.

Сума додаткової заробітної плати розраховується від розміру основної:

$$ЗПдод = ЗПосн \cdot \frac{\%дод.}{100}. \quad (4.3)$$

де %дод – відсоток нарахувань додаткової заробітної плати, %

$$ЗПдод = 9863{,}6 \cdot \frac{20}{100} = 1972{,}7 \text{ грн.}$$

Нарахування на заробітну плату здійснюють від суми основної та додаткової зарплати. Розмір нарахувань становить 22%.

$$Нзп = (ЗПосн + ЗПдод) \cdot \frac{\%нар}{100} \quad (4.4)$$

де %нар – відсоток нарахувань заробітної плати, %

$$Нзп = (9863{,}6 + 1972{,}7) \cdot \frac{22}{100} = 2604 \text{ грн.}$$

Розрахунок загальної суми витрат на оплату праці з нарахуваннями розробника наведемо у вигляді таблиці 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці з нарахуваннями розробників програмного продукту

Найменування посади	Основна заробітна плата, грн.	Додаткова заробітна плата, год.	Розмір нарахувань на оплату праці, грн.	Всього витрат на оплату праці з нарахуваннями, грн.
Інженер-програміст	9863,6	1972,7	2604	14440,3

Розрахунок суми амортизаційних відрахувань обладнання, яке використовується при розробці продукту здійснюється найбільш поширеним прямолінійним методом. При створенні продукту використовувався ноутбук та принтер. Відповідно до податкового кодексу України електронно-обчислювальні машини (ПК) відносять до 4 групи, з терміном корисного використання 2 роки. Відповідно річна норма амортизації становить 50%. Балансова вартість ПК (або ноутбуку) на момент придбання (розрахунку) становить 18000 грн, принтера - початкова вартість 10000 грн, приймаємо річну норму амортизації в розмірі 20% (термін використання 5 років).

Розрахунок річної суми амортизаційних відрахувань:

$$\text{Аріч} = \text{БВобл} \cdot \frac{\%\text{аморт.}}{100}; \quad (4.5)$$

де БВобл. - ціна придбання, або балансова вартість комп’ютера на момент придбання, грн.

%аморт – річний відсоток нарахувань додаткової заробітної плати, %

$$\text{Аріч. комп.} = 18000 \cdot \frac{50}{100} = 9000\text{грн.}$$

$$\text{Аріч. принт.} = 10000 \cdot \frac{20}{100} = 2000\text{грн.}$$

Сума амортизаційних відрахувань, що увійде до собівартості програмного продукту розраховується с врахуванням часу використання ноутбуку та принтера при розробці продукту.

$$А = Аріч \cdot \frac{Твик(днів)}{365} \quad (4.6)$$

де Твик. - термін використання персонального комп'ютера та принтера при розробці програмного продукту, міс. або дні.

Таким чином, загальна сума амортизаційних відрахувань, становитиме:

$$А = (9000 + 2000) \cdot \frac{13днів}{365} = 391{,}8 \text{ грн.}$$

Витрати на матеріали – визначаються відповідно до кількості витрачених матеріалів на розробку продукту та ціни на даний матеріал, на момент розрахунку. Витрати на матеріали, що були використані на розробку продукту наведені в таблиці 4.3.

Таблиця 4.3 – Розрахунок матеріальних витрат на створення продукту

Найменування матеріалу	Ціна за одиницю матеріалу, грн./од	Витрачено, од	Вартість витрачених матеріалів, грн.
Чорнила, банка	320	3	960
Папір, уп.	200	2	400
Разом витрат			1060

Витрати на електроенергію – з потужності обладнання, що використовується при розробці продукту (П=0,44), коефіцієнту використання потужності (Квп=0,95), фактичного часу (трудомісткості) на розробку та ціни електроенергії, що склалась на момент розрахунку (Ц1кВт=4,32 грн.):

$$Вел. = П \cdot Квп \cdot Тр \cdot Ц1кВт = 0{,}44 \cdot 0{,}95 \cdot 112 \cdot 4{,}32 = 202{,}2 \text{ грн.}; \quad (4.7)$$

Розрахунок суми витрат на розробку відображені даними таблиці 4.4

Таблиця 4.4 – Розрахунок витрат на створення продукту

Найменування витрат	Вартість, грн.
Витарти на оплату праці з нарахуваннями, грн.	14440,3
Амортизаційні відрахування, грн	391,8
Витрати на матеріали, грн	1060
Витрати на електроенергію, грн	202,2
Разом витрат	15034,4

4.3.2. Витрати на розміщення продукту.

Для публічного розміщення сайту піцерії також необхідно врахувати вартість хостингу та домену. Річна вартість хостингового обслуговування становить орієнтовно 200 грн/ міс, або 2400 грн/рік), а річна вартість доменного імені — приблизно 400 грн.

Таким чином, загальні витрати на розміщення інформаційної системи наведено 2800 грн

4.3.3. Розрахунок собівартості одиниці копії програмного продукту

Сукупні витрати на створення та розміщення програмного продукту складаються з загальної суми витрат:

$$Соб.прод = Вроз. + Врозм. + Врек + Вінш. \quad (4.8)$$

де Врозр. – витрати на розробку програмного продукту, грн;

Врозм.с. - витрати, пов’язані з розміщенням інформації на сервері та вартість хостингу сайту, грн;

Врек – витрати на рекламу, приймаємо на рівні - 2600 грн.

Вінш – інші витрати, які не включені до вищеперелічених витрат, коливаються в межах 10-30% , приймаємо на рівні 25%.

$$\text{Вінш.} = (15034{,}4 + 2800 + 2600) \cdot \frac{25}{100} = 5108{,}6\text{грн.}$$

Таким чином, витрати на створення продукту складатиме:

$$\text{Вствор.} = 15034{,}4 + 2800 + 2600 + 5108{,}6 = 25543 \text{ грн.}$$

Оскільки, програмний продукт приймається та впроваджується за завданням замовника, а також може реалізовуватись іншим покупцям відповідно доцільно визначити ціну продажу. Ціна реалізації залежить від кількості реалізованого продукту. В нашому випадку плануємо реалізацію 10 од продукту. Таким чином, витрати на створення продукту розподіляються на кількість запланованих продажів та становлять собівартість одиниці продукту для продажу. Таким чином, собівартість одиниці продукту становить:

$$\text{Спрод.} = \frac{25543}{10} = 2554{,}3 \frac{\text{грн}}{\text{од}}.$$

4.3.4.Розрахунок ціни реалізації проектного рішення

Ціна реалізації визначається виходячи з рівня запланованої прибутковості (рентабельності) задля задоволення цільової аудиторії та прибутковості замовника. Нами було проведено аналіз ринкової вартості данного продукту, відповідно до результатів, рівень рентабельності за згодою замовника може коливається в межах 40-80%. Приймаємо рівень прибутковості 60%, рівень податку на додану вартість – згідно чинного законодавства становить 20%.

Таким чином, ціна реалізації продукту розраховується за формулою:

$$\text{Цр} = \text{Спрод} \cdot \left(1 + \frac{\text{Р}}{100}\right) \cdot \left(1 + \frac{\text{ПДВ}}{100}\right) \quad (4.9)$$

де Р– прийнятий норматив рентабельності, 60%.

ПДВ – ставка податку на додану вартість, приймається на рівні 20%.

$$\text{Цр} = 2554{,}3 \cdot \left(1 + \frac{60}{100}\right) \cdot \left(1 + \frac{20}{100}\right) = 4904{,}3$$

4.3.5.Розрахунок прибутку від створення та продажу програмного продукту.

Розрахунок чистого прибутку від реалізації запланованого обсягу програмного продукту здійснюється з врахуванням річного прибутку від реалізації та податку на прибуток, що існує на момент розрахунку. При розрахунку прибутку ціна реалізації приймається без ПДВ та становить ціну реалізації власника:

$$\text{Цр власн.} = 2554{,}3 \cdot \left(1 + \frac{60}{100}\right) = 4086{,}9 \text{ грн.}$$

Прибуток від продажу розраховується за формулою:

$$\text{П} = (\text{Цр власн.} - \text{Спрод}) \cdot \text{Опрод} \quad (4.10)$$

де Опрод. – кількість одиниць продажу програмного продукту, од;

Таким чином загальна сума прибутку від продажу становить:

$$\text{П} = (4086{,}9 - 2554{,}3) \cdot 10 = 15326 \text{ грн.}$$

Відповідно чистий прибуток за відрахуванням податку на прибуток становитиме:

$$\text{ЧП} = \text{П} - \left(\text{П} \cdot \frac{\%\text{ПП}}{100}\right) \quad (4.11)$$

де %ПП – відсоток податку на прибуток, приймається на рівні 18%

$$\text{ЧП} = 15326 - \left(15326 \cdot \frac{18}{100}\right) = 12567{,}3 \text{ грн}$$

4.3.6. Термін окупності.

Термін окупності характеризує період часу протягом якого окупляться витрати на розробку програмного продукту замовником.

Розрахунок періоду окупності здійснюється за формулою:

$$\text{Ток} = \frac{\text{Вствор.}}{\text{ЧП}}; \quad (4.12)$$

$$\text{Ток} = \frac{225543}{12567{,}3} = 2{,}032 \text{ роки.}$$

Таким чином, створення та подальший продаж розробленого програмного продукту є достатньо ефективним для замовника. Витрати на створення продукту становлять майже 25,5 тис. грн. При плануванні обсягу подальшого продажу на рівні 10 од., розмір отриманого прибутку для замовника становить 12,6 тис. грн., а період окупності при даних показниках – 2 роки. Слід зауважити, що показники ефективності розраховувались за рівнем прибутковості нижчим ринкового показника, також при розрахунках використовували мінімальну кількість проданого продукту, відповідно на перспективу продаж даного продукту є достатньо ефективним

4.4 Охорона праці

Характеристика приміщення і виконуваних робіт: площа приміщення 20 м^2, у приміщенні операторів є два комп'ютери. Комп'ютери об'єднані в локальну мережу. До складу основного обладнання входять персональні ПК та друкуючий пристрій типу Canon S300. У приміщенні є один вхід. Обладнання розміщується таким чином, щоб був забезпечений вільний прохід до всіх робочих місцях. Робота в основному пов'язана з ПК.

Приміщення, в яких експлуатуються ПК повинні відповідати вимогам з пожежної безпеки, зазначеними у наступних нормативно-правових документах:

– Правила пожежної безпеки в Україні;

– НПАОП 40.1-1.32-01 (ДНАОП 0.00-1.32-01) Правила будови електроустановок. Електрообладнання спеціальних установок”;

– Правил безпечної експлуатації електроустановок споживачів;

- ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою;

– Вимогам нормативно-технічної та експлуатаційної документації заводу-виробника ПК.

Будівлі і ті їх частини, в яких розташовуються ПК, повинні мати не нижче II ступеня вогнестійкості. Приміщення для обслуговування, ремонту та налагодження ПК повинні належати за пожежовибухобезпекою до категорії В відповідно до ДСТУ Б В.1.1-36:2016, а за класом приміщення – до П-IIа за ПБЕ.

Категорія пожежної безпеки приміщення офісу з комп'ютерами, згідно з українськими нормами, визначається як категорія В (пожежовибухонебезпечна), а за класом приміщення – як П-IIа (помірно небезпечна за горючістю). Це зумовлено наявністю горючих матеріалів, таких як папір, пластик, та електронного обладнання, а також потенційною небезпекою короткого замикання.

В приміщенні з ПК знаходяться горючі речовини та матеріали: папір, пластик, ізоляційні матеріали обладнання, дерев'яні меблі, які можуть утворювати вибухонебезпечні суміші з повітрям або, при пожежі, виділяти значну кількість теплоти. Горючі матеріали в приміщенні не утворюють вибухонебезпечних сумішей. Будь-яке електрообладнання може стати джерелом пожежі через перегрів, коротке замикання або несправність.

Для приміщення з ПК обрано порошкові вогнегасники, які підходять для гасіння електрообладнання, що знаходиться під напругою.

Розмір та кількість вогнегасників обирається в залежності від площі приміщення. Згідно МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ НАКАЗ 15.01.2018 № 25, 1 кг. гасячої речовини на 25 м² площі об'єкта. Для площі 20м^2 достатньо вогнегасника з 1 кг порошка, обираємо вогнегасник типу ВП-1.

Вогнегасники слід встановлювати у легкодоступних та видних місцях, а також у пожежонебезпечних місцях, де найбільш вірогідна поява осередків пожежі. При цьому необхідно забезпечити їх захист від потрапляння прямих сонячних променів та дії опалювальних та нагрівальних приладів.[2]

Також, згідно правил пожежної безпеки, в приміщенні має бути план евакуації та інструкція щодо дій персоналу під час пожежі.

У разі виявлення ознак пожежі (горіння):

– вимкнути електропостачання;

– вжити (за можливості) заходів щодо евакуювання людей, гасіння (локалізації) пожежі первинними засобами пожежогасіння та збереження матеріальних цінностей;

– повідомити про пожежу керівника;

– у разі необхідності викликати інші аварійно-рятувальні служби.

ВИСНОВКИ

Було розроблено інформаційну систему для автоматизації роботи піцерії зі зручним інтерфейсом.

При виконанні проекту було порівняно аналоги системи, що дозволило виявити їх сильні та слабкі сторони, що дало ідеї щодо функціоналу, який покращить використання системи. Було сформоване технічне завдання. Після цього функціональні вимоги було проілюстровано на діаграмі варіантів використання, що дозволило зручний візуальний доступ до перегляду їх структури. Далі на етапі створення діаграми IDEF0 було розглянуто і зображено процеси функціонування інформаційної системи.

Створення діаграми класів дозволило полегшити процес розробки системи, так як вона зручно відображає структуру даних, що дозволяє забезпечити правильний доступ до даних об’єктів.

ER-діаграма є результатом створення бази даних, і дозволяє швидко зрозуміти її структуру і допомагає в її редагуванні, забезпечуючи меншу кількість помилок.

Було описано функціонування системи у підрозділах опису програмної реалізації та інструкції користувача. В підрозділі тестування було описано проблеми, які виникли при тестуванні системи і вказано статус їх виконання.

В розділі розрахунку техніко-економічних показників було розраховано витрати на розробку системи. В розділі охорони праці було обрано вогнегасник та вказано правила пожежної безпеки.

СПИСОК ЛІТЕРАТУРИ

1. НПАОП 40.1-1.32-01 (ДНАОП 0.00-1.32-01). Правила будови електроустановок. Електрообладнання спеціальних установок.
2. ДСТУ Б В.1.1-36:2016. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою
3. Джеймс Моралес. Що таке діаграма класів UML і найкращий творець діаграм класів UML: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/#part1
4. Зінов'єва О. Г., Шаров С. В. Проектування інформаційних систем: конспект лекцій. Запоріжжя, 2024. 148 с.
5. Методичні рекомендації до виконання та захисту кваліфікаційної роботи бакалавра спеціальності 122 «Комп'ютерні науки» / уклад.: С.В. Шаров, Д.В. Лубко, О.Г. Зінов'єва. Запоріжжя, ТДАТУ, 2025. 63 с.
6. Методологія IDEF0. URL: https://stud.com.ua/87184/ekonomika/metodologiya_idef0
7. Основи створення web-сайтів. URL:: http://intkonf.org/sitnik-di-tudihata-ayunavchalniy-kurs-osnovi-stvorennya-web-saytiv/
8. Петренко В.С., Карнаушенко А.С. Сучасний стан та перспективи розвитку доставки продуктів харчування в Україні/*Економіка та управління підприємствами*. № 1(18).2020. С. 132-148
9. Популярність піци в Україні. URL: https://lviv.cx.ua/populiarnistpitsy-v-ukraini-trendy-ta-statystyka/
10. Посібник ALM. URL: https://visuresolutions.com/uk/посібник-з-відстеження-управління-вимогами/функціональні-вимоги/
11. Чайка, І. М., Дністрянська, Н. І. Системи автоматизації ресторанного бізнесу в Україні та їхні маркетингові можливості. *Індустрія туризму і гостинності в Центральній та Східній Європі*, (12), 2025, с.63-67.

12. Швець О. Занурення у патерни проектування. URL: https://refactoring.guru/uk/design-patterns/adapter
13. CSS.IN.UA. URL: https://css.in.ua/article/shcho-take-html_10
14. How is CSS pseudo content treated by screen readers? URL: https://accessibleweb.com/question-answer/how-is-css-pseudo-content-treated-by-screen-readers/
15. Laudon K. C., Laudon J. P. Management Information Systems: Managing the Digital Firm. Pearson Education, 2021. 656 p.
16. Martin Hjalm. Digital transformation in the food & beverage industry: How new technologies are serving innovation. URL: https://www.vaimo.com/blog/digital-transformation-in-the-food-beverage-industry-how-new-technologies-are-serving-innovation
17. MySQL Stored Procedure Advantages URL: https://www.tutorialspoint.com/What-are-theadvantages-and-disadvantages-of-using-MySQL-stored-procedures
18. Online Food Delivery – Worldwide. URL: https://www.statista.com/outlook/emo/online-food-delivery/worldwide
19. UML діаграми для моделювання процесів інформаційних систем URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html
20. Virtual DOM and Internals. URL: https://uk.legacy.reactjs.org/docs/faq-internals.html
21. Zero-config deployments. URL: https://svelte.dev/docs/kit/adapter-auto

ДОДАТОК А

Файл shared.css:

```
body{
    margin:0px;
    font-family: 'Noto Sans', sans-serif;
    font-size: calc(15px + 0.390625vw);
    background: #FFFFC7;
    color: #473335;
}

.Flex {
        background-color: #FBE790;
        margin: 5px;
        padding: 1em;
        text-align: center;
}

.Cont {
    width: 100%;
    display: grid;
    grid-template-columns: repeat(4, minmax(0, 1fr));;
    padding: auto;
}

.pix {
    /* object-fit: contain; */
    /* width: 30px; */
    width: 100%;
}

button {
    padding: 10px;
    margin-top: 8px;
    border-style: none;
    border-radius: 3px;
    cursor: pointer;
```

```
}

.button {
   margin-left: 15px;
   background-color:#ff8000;
   color: white;
}

.button:hover, .button:focus {
   background-color: #fcac5d;
}

@media (max-width: 900px) {
   .Cont {
      grid-template-columns: repeat(3, minmax(0, 1fr));
   }
}
@media (max-width: 660px) {
   .Cont {
      grid-template-columns: repeat(2, minmax(0, 1fr));
   }
}
```

ДОДАТОК Б

Файл +layout.svelte:

```
<script>
   let { children } = $props();
   import cartImg from '$lib/assets/5.svg'
   import home from '$lib/assets/Home.svg'
   import x from '$lib/assets/x.svg'

   import { cart, add, remove, clearCart, removeFlav, flavChange } from
'$lib/stores/cartStore';
   import { browser } from '$app/environment'
   import { onMount } from 'svelte';

   let cartVis = $state(false)

class navImage {
   constructor(src, alt, href) {
      this.src=src;
      this.alt=alt;
      this.href=href;
   }
}

class navText {
   constructor(text, href) {
      this.text=text;
      this.href=href;
   }
}

class navFill {
   constructor(){

   }
}
```

```
const nav = [
   new navImage(home, 'Home', '/home'),
   new navText('Піцци', '/pizza'),
   new navText('Бургери', '/burger'),
   new navText('Десерти', '/dessert'),
   new navText('Напої', '/drinks'), new navFill(),
]

   let drinksel = $state(false)
   let flavouid
   if (browser) {
      flavouid = JSON.parse(localStorage.getItem('flavouid'))
   }

   const total = $derived($cart.reduce((accumulator, currentObject) => {
      return accumulator + currentObject.price*currentObject.quant
   }, 0))


   let cartt=[]; let totall=0; let sending=$state(false); let success=false; let
message=$state('')

   async function sendCart() {
      if($cart.length>0) {
         sending=true
         success=false
      }
      const cartData = { items: $cart, total: total}
      try {
         const response = await fetch('/api/sendcart', {
            method: 'POST',
            headers: {
               'Content-Type': 'application/json'
            }, body: JSON.stringify(cartData)
         })
         const result = await response.json();
         if(result.success) {
```

```
            success=true; message='Order placed successfully! Your order ID is: '+result.id
            clearCart()
          } else {
            success=false
            message='Error:'+result.message
          }
        } catch (error) {
          success=false
          message='This is catch. '+error.message
        } finally {
          sending=false
        }
    }

</script>

<header>
    <ul class="main-menu">
        {#each nav as item}
        <li style={item instanceof navFill ? 'flex-grow: 1;' : ''}>
            {#if !(item instanceof navFill)}
            <a href={item.href} draggable="false">
            {#if item instanceof navText}
            {item.text}
            {:else if item instanceof navImage}
            <img src={item.src} alt={item.alt}>
            {/if}
            </a>
            {/if}
        </li>
        {/each}
        <li><button class="imgbutton" onclick={() => (cartVis=true)}><img src={cartImg} alt="Cart"></button></li>
    </ul>

</header>
```

```
{#if cartVis}
<aside class="right">
  <div style="display: flex;"><div style="flex-grow: 1;"><h4 style="padding: 0;">Кошик</h4></div>
  <button class="imgbutton" onclick={() => (cartVis=false)}><img src={x} alt="x_close" draggable="false"></button></div>
  <label><input type="checkbox" bind:checked={drinksel}>Напої</label><span style="float: right;">Всього: {total} грн</span>

  <hr width="100%">

  <div class="cartcont">
    {#if $cart.length === 0}
    Ваш кошик порожній!
    {:else if drinksel}
    {#each $cart as item (item.id)}
    {#if item.type.includes('drink')}
    {#if item.flavour}
    <details>
      <summary>{item.name}, {item.quant*item.price} грн</summary>
      <dl>{#each Object.entries(item.flavour) as [idd, flav]}

        <dd>{flav.name}</dd>
        <dt><input type="number" min="0" value={flav.quant}
          oninput={(e) => {console.log("rfwefd"); flavChange(item.id, idd, parseInt(e.target.value))}}>

      </dt>{/each}</dl>
    </details>
    {:else}
    <!-- regular items -->
    <div style="display:flex;">
      <div class="Flex">

        <img src={'/drink/'+item.imgLink} alt={item.name} class="pix"></div>
        <div style="margin-top: 1em; width: 20vw;">
        <h4>{item.name}</h4>
```

```
            {item.price*item.quant} грн
            <input type="number" min="0" bind:value={item.quant}
            onchange={()=> {if(item.quant===0) {remove(item.id)}}}>

          </div></div>
        {/if}{/if}{/each}
        {:else}
        {#each $cart as item (item.id)}
        {#if !item.type.includes('drink')}
        <div style="display:flex;">
          <div class="Flex">
            <!-- everything else -->
            <!-- <div class="jeifvnu"> -->
             <!-- {#if item.type === "dessert"}{path="/dessert/"}
             {:else}{path="/pizza/"}{/if} -->
            <img src={(item.type === 'dessert' ? '/dessert/'
            : item.type === 'burger' ? '/burger/'
            : '/pizza/')+item.imgLink} alt={item.name} class="pix"></div>
            <div style="margin-top: 1em; width: 20vw;">
            <h4>{item.name}</h4>
            <!-- {#if item.foodcont.cont}<em>{item.foodcont.cont}</em><br>{/if} -
->
            <!-- <span style="color: #FE5F55"></span> -->
            {item.price*item.quant} грн
            <input type="number" min="0" bind:value={item.quant}
            onchange={()=> {if(item.quant===0) {remove(item.id)}}}>
            <!-- <button onclick={cart.add}>Додати</button> -->
          </div></div>{/if}
        {/each}
        {/if}
    </div>
    <button class="cartContr" onclick={clearCart}>Очистити</button>
    <button class="cartContr" onclick={sendCart}>{sending ? 'Processing...'
:'Відправити'}</button><br>{message}
</aside>
{/if}

{@render children()}
```

```
<style>
    .imgbutton {
        border: none;
        background-color: transparent;
        cursor: pointer;
    }
    .cartContr {
        background-color: #351431;
        padding: 10px;
        margin-top: 8px;
        border-style: none;
        color: white;
        border-radius: 3px;
        cursor:pointer;
    }
    .right {
        color: #251101;
        width: 360px;
        background: #847996;
        float: right;
        position: fixed;
        top: 0;
        bottom: 0;
        right: 0px;
        z-index: 999;

        padding: 1em;
    }
    .cartcont {
        overflow-y: auto;
        max-height: calc(100vh - 100px);
    }
    dl {
        margin: 0;
    }
    dd {
        float: left;
```

```
    margin: 0;
    clear: both;
    width:70%;
    display: block;
}
dt {
    width:30%;
    display: block;
    float: right;
}
details {
    padding-top: 10px;
    padding-bottom: 5px;
    display: block;
    cursor: pointer;
}

h4 {
    padding: 0;
    margin: 0;
}
hr {
    color: #251101;
}
input {
    width: 60px;
    font-size: calc(11px + 0.390625vw);
    font-family: 'Noto Sans', sans-serif;
    margin-right: 15px;
}

header {
    border:none;
    position: sticky;
    top: 0px;
    width: 100%;
    background: #FCAA67;
```

```
    }

.main-menu {
    margin: 0;
    padding: 0;
    list-style: none;
    display: flex;
    flex-direction: row;
}

.main-menu > li {
    display: flex;
    white-space: nowrap;
    -user-drag: none;
    user-select: none;
}

.main-menu a {
    display: block;
    margin-left: -1px;
    padding: 8px 10px;
    text-decoration: none;
    color: #473335;
    -user-drag: none;
    user-select: none;
}

.main-menu a:hover {
background-color: #FED597;
}

img {
    width:calc(20px + 0.390625vw);
    user-select: none;
}
</style>
```

ДОДАТОК В

Файл /pizza/+page.svelte:

```
<script>
   import { page } from '$app/stores'
   import { add, cart } from '$lib/stores/cartStore';

   let postfilter = [...$page.data.pizzasqr]
   let filterr = null; let cheese = null; let sort = null;
   function handleFilterChange() {
      if (event.target.checked) {
         filterr = event.target.value;
      } else {
         if (filterr === event.target.value) {
            filterr = null;
         }
      }
      switch (filterr) {
         case "nomeat":
            postfilter=$page.data.pizzasqr.filter(item => item.type==='pizza_nomeat'); break;
         case "meat":
            postfilter=$page.data.pizzasqr.filter(item => item.type==='pizza_meat'); break;
         case "fish":
            postfilter=$page.data.pizzasqr.filter(item => item.type==='pizza_fish'); break;
         default:
         postfilter=[...$page.data.pizzasqr]; break;
      }
   }

   function handleSortChange() {
      if (event.target.checked) {
         sort = event.target.value
      } else {
         if (sort === event.target.value) {
```

```
            sort = null
        }
    }
    switch (sort) {
        case "cheaper":
            postfilter=[...$page.data.pizzasqr].sort((a,b) => a.price-b.price); break;
        case "expensive":
            postfilter=[...$page.data.pizzasqr].sort((a,b) => b.price-a.price); break;
        default:
        postfilter=[...$page.data.pizzasqr]; break;
    }
}
</script>

<svelte:head>

    <title>Pizzas</title></svelte:head>

    <h2 align="center">Піцци</h2>
    <main>
        <fieldset>
            <label><input type="checkbox" name="nomeat" value="nomeat"
                checked={filterr === "nomeat"} onchange={handleFilterChange}>Без
м'яса</label>
            <label><input type="checkbox" name="meat" value="meat"
                checked={filterr === "meat"}
onchange={handleFilterChange}>М'ясна</label>
            <label><input type="checkbox" name="fish" value="fish"
                checked={filterr === "fish"}
onchange={handleFilterChange}>Рибна</label>
        </fieldset>
        <fieldset>
            <label><input type="checkbox" name="cheaper" value="cheaper"
                checked={sort === "cheaper"}
onchange={handleSortChange}>Спочатку дешевше</label>
            <label><input type="checkbox" name="expensive" value="expensive"
                checked={sort === "expensive"}
onchange={handleSortChange}>Спочатку дорожче</label>
```

```
      </fieldset>
   <div class="Cont">
      {#each postfilter as p}
         <div class="Flex">
            <img src={"/pizza/"+p.imgLink} alt={p.name} class="pix">
            <h3>{p.name}</h3>
            <em>{p.foodcont.cont}</em><br>
            <span class="Price">{p.price}</span> грн<button class="button"
onclick={add(p)}>Додати</button>
         </div>
      {/each}
   </div>
</main>
```

ДОДАТОК Г

Файл /pizza/+page.server.js:

```
import { db } from '$lib/server/db'
import { foodcont, products, productrels, contrels } from '$lib/server/db/schema'
import { eq } from "drizzle-orm"

export async function load() {
	try{
		const pizzasqr = await db.query.products.findMany({
			where: or(eq(products.type, 'pizza_meat'),
eq(products.type, 'pizza_nomeat'), eq(products.type, 'pizza_fish')),
			with: {
				foodcont: true
			}
		})

	return {
		pizzasqr
	};
	} catch {

	}
}
```

ДОДАТОК Д

Файл /main/+page.svelte:

```
<script>
   import { page } from '$app/stores'
   import { add, cart } from '$lib/stores/cartStore';
   import logo from '$lib/assets/Logo.png'
   import pic from '$lib/assets/pic.png'

</script>

<svelte:head>

   <title>Home page</title></svelte:head>

   <main>
   <h3 align="center">Піцерія "Ваша піца"</h3>
   <img src={logo} alt="Logo" class="logo">
   <div style="float: rigth;">
   Ми працюємо для вас!<br><br>

   В нашому меню:
   <ul>
      <li>Піци🍕</li>
      <li>Бургери🍔</li>
      <li>Десерти🍰</li>
      <li>Напої🍵</li>
   </ul><br></div>
   <img src={pic} alt="Pizza illustration" class="illustr">
   Є можливість онлайн-замовлень.<br>
   Контакти: м. Мелітополь, пр. 50-р. Перемоги, 35<br><br>

   Графік роботи: Пн-Нд 10:00 - 22:00<br>
   тел. +38-068-797-83-98<br><br>

   Ми готові до співпраці зі службами доставки.
```

```
</main>

<style>
   main{
      padding: 10px;
   }
   ul {
      margin-top: 0;
   }
   li {
      margin-left: 10px;
   }
   .logo {
      width: 20vw;
      float: left;
      margin-right: 20px;
   }
   .illustr {
      float: right;
      width: 30vw;
   }

</style>
```

ДОДАТОК Е

Файл /burger/+page.svelte:

```
<script>
   import { page } from '$app/stores'
   import { add, cart } from '$lib/stores/cartStore';

   let sort = null; let postfilter = [...$page.data.burgersqr]
   function handleSortChange() {
      if (event.target.checked) {
         sort = event.target.value;
      } else {
         if (sort === event.target.value) {
            sort = null;
         }
      }
      switch (sort) {
         case "cheaper":
            postfilter=[...$page.data.burgersqr].sort((a,b) => a.price-b.price); break;
         case "expensive":
            postfilter=[...$page.data.burgersqr].sort((a,b) => b.price-a.price); break;
         default:
         postfilter=[...$page.data.burgersqr]; break;
      }
   }
</script>

<svelte:head>

   <title>Burgers</title></svelte:head>

   <h2 align="center">Бургери</h2>
   <main>
      <fieldset>
         <label><input type="checkbox" name="cheaper" value="cheaper"
            checked={sort === "cheaper"}
onchange={handleSortChange}>Спочатку дешевше</label>
```

```
        <label><input type="checkbox" name="expensive" value="expensive"
          checked={sort === "expensive"}
onchange={handleSortChange}>Спочатку дорожче</label>
      </fieldset>
   <div class="Cont">
      {#each postfilter as p}
        <div class="Flex">
          <img src={"/burger/"+p.imgLink} alt={p.name} class="pix">
          <h3>{p.name}</h3>
          <span class="Price">{p.price}</span> грн<button class="button"
onclick={add(p)}>Додати</button>
        </div>
      {/each}
   </div>
</main>
```

ДОДАТОК Є

Файл /burger/+page.server.js:

```
import { db } from '$lib/server/db'
import { foodcont, products, productrels, contrels } from '$lib/server/db/schema'
import { eq } from "drizzle-orm"

export async function load() {
	try{
		const burgersqr = await db.query.products.findMany({
			where: eq(products.type, 'burger')
		})
	return {
		burgersqr
	};
	} catch {

	}
}
```

ДОДАТОК Ж

Файл /dessert/+page.svelte:

```
<script>
   import { page } from '$app/stores'
   import { add, cart } from '$lib/stores/cartStore';

   let sort = null; let postfilter = [...$page.data.dessertsqr]
   function handleSortChange() {
      if (event.target.checked) {
         sort = event.target.value;
      } else {
         if (sort === event.target.value) {
            sort = null;
         }
      }
      switch (sort) {
         case "cheaper":
            postfilter=[...$page.data.dessertsqr].sort((a,b) => a.price-b.price); break;
         case "expensive":
            postfilter=[...$page.data.dessertsqr].sort((a,b) => b.price-a.price); break;
         default:
         postfilter=[...$page.data.dessertsqr]; break;
      }
   }
</script>

<svelte:head>

   <title>Desserts</title></svelte:head>

   <h2 align="center">Десерти</h2>
   <main>
      <fieldset>
         <label><input type="checkbox" name="cheaper" value="cheaper"
            checked={sort === "cheaper"}
onchange={handleSortChange}>Спочатку дешевше</label>
```

```
        <label><input type="checkbox" name="expensive" value="expensive"
          checked={sort === "expensive"}
onchange={handleSortChange}>Спочатку дорожче</label>
      </fieldset>
   <div class="Cont">
      {#each postfilter as p}
        <div class="Flex">
          <img src={"/dessert/"+p.imgLink} alt={p.name} class="pix">
          <h3>{p.name}</h3>
          <span class="Price">{p.price}</span> грн<button class="button"
onclick={add(p)}>Додати</button>
        </div>
      {/each}
   </div>
</main>

<style>

</style>
```

ДОДАТОК З

Файл /dessert/+page.server.js:

```
import { db } from '$lib/server/db'
import { foodcont, products, productrels, contrels } from '$lib/server/db/schema'
import { eq } from "drizzle-orm"

export async function load() {
	try{
		const dessertsqr = await db.query.products.findMany({
			where: eq(products.type, 'dessert')
		})

	return {
		dessertsqr
	};
	} catch {

	}
}
```

ДОДАТОК И

Файл /drinks/+page.server.js:

```
<script>
   import { page } from '$app/stores'
   import { add, cart } from '$lib/stores/cartStore';
   import { browser } from '$app/environment'

   let sort = null; let postfilter = [...$page.data.drinksqr]
   function handleSortChange() {
      if (event.target.checked) {
         sort = event.target.value;
      } else {
         if (sort === event.target.value) {
            sort = null;
         }
      }
      switch (sort) {
         case "cheaper":
            postfilter=[...$page.data.drinksqr].sort((a,b) => a.price-b.price); break;
         case "expensive":
            postfilter=[...$page.data.drinksqr].sort((a,b) => b.price-a.price); break;
         default:
         postfilter=[...$page.data.drinksqr]; break;
      }
   }
   let flavou={};let flavouid={}//init the array with 1st values of all elements of
drinks_juice type
   for (const item of $page.data.drinksqr) {

      if (item.juicesflavours) {

         flavou[item.id]=item.juicesflavours[0].flavid

         for (const flavoue of item.juicesflavours) {
            flavouid[flavoue.flavid]=flavoue.flavour
         }
```

```
        }
    }
    if(browser) {
        localStorage.setItem('flavouid', JSON.stringify(flavouid))
    }

    function flavouChange(event, drinkId) {
        flavou = {...flavou, [drinkId]: parseInt(event.target.value) }
    }
</script>

<svelte:head>

    <title>Drinks</title></svelte:head>

    <h2 align="center">Напої</h2>
    <main>
        <fieldset>
            <label><input type="checkbox" name="cheaper" value="cheaper"
                checked={sort === "cheaper"}
onchange={handleSortChange}>Спочатку дешевше</label>
            <label><input type="checkbox" name="expensive" value="expensive"
                checked={sort === "expensive"}
onchange={handleSortChange}>Спочатку дорожче</label>
        </fieldset>

    <div class="Cont">
        {#each postfilter as p (p.id)}
            <div class="Flex">

                <img src={"/drink/"+p.imgLink} alt={p.name} class="pix">
                <h3>{p.name}</h3>
                <span class="Price">{p.price}</span> грн
                {#if p.juicesflavours}

                <select name="flavour" value={flavou[p.id]}
onchange={(e)=>flavouChange(e, p.id)}>
```

```
            {#each p.juicesflavours as f }

               <option value={f.flavid}>{f.flavour}</option>
            {/each}</select><button class="button" onclick={() =>
            add({ id: p.id, name: p.name, imgLink: p.imgLink, price: p.price, type:
p.type, quant: 1,
            flavid: flavou[p.id] }

            )}>Додати</button>
            {:else}<button class="button" onclick={add({ name: p.name, imgLink:
p.imgLink, price: p.price, type: p.type})}>Додати</button>
          {/if}</div>
       {/each}
    </div>
</main>
```

ДОДАТОК І

Файл /drinks/+page.server.js:

```
import { db } from '$lib/server/db'
import { foodcont, products, productrels, contrels, juicesflavours, jfrels } from '$lib/server/db/schema'
import { eq } from "drizzle-orm"

export async function load() {
	try{
		const drinksqr = await db.query.products.findMany({
			where: eq(products.type, 'drink_juice'),
			with: {
				juicesflavours: {
					where:
					eq(juicesflavours.avaliable,true),
				}
			}
		})

	return {
		drinksqr
	};
	} catch(e) {
		return {
			status: 500,
			error: { message: "Failed to load data", details: e.message || 'Unknown error'}
		}
	}
}
```

ДОДАТОК Й

Файл schema.js:

```
import { mysqlTable, smallint, int, char, varchar, boolean, json } from 'drizzle-orm/mysql-core';
import { relations } from 'drizzle-orm';

export const products = mysqlTable('products', {
   id: smallint('id', { unsigned: true}).primaryKey().autoincrement(),
   imgLink: char('imgLink', {length: 30}).notNull().unique(),
   name: char('name', { length: 50}).notNull().unique(),
   price: smallint('price', { unsigned: true}).notNull(),
   type: char('type' , { length: 20 }).notNull()
})

export const foodcont = mysqlTable('foodcont', {
   id: smallint('id', { unsigned: true}).references(() => products.id, {onDelete: 'cascade'}).notNull().unique(),
   cont: char('cont', { length: 70}).notNull(),
})

export const orders = mysqlTable('orders', {
   id: varchar('id', {length: 255}).primaryKey(),
   total: smallint('total', {unsigned: true}).notNull(),
   complete: boolean('complete').default(0)
})

export const carts = mysqlTable('carts', {
   id: varchar('id', { length: 255}).references(() => orders.id, {ondelete: 'cascade'}).notNull(),
   prodid: smallint('prodid', {unsigned: true}).notNull(),
   name: char('name', {length: 50}).notNull(),
   quant: smallint('quant', {unsigned: true}).notNull(),
   price: smallint('price', { unsigned: true}).notNull(),
   flavour: json('flavour')
})
```

```
export const juicesflavours = mysqlTable('juicesflavours', {
   id: smallint('id', { unsigned: true}).notNull().references(() => products.id, {onDelete: 'cascade'}).notNull(),
   flavour: char('flavour', { length: 20 }).notNull(),
   flavid: smallint('flavid', { unsigned: true}).notNull().primaryKey().autoincrement(),
   avaliable: boolean('avaliable').default(1)
})

export const productrels = relations(products, ({one, many}) => ({
   foodcont: one(foodcont, {
      fields: [products.id],
      references: [foodcont.id],
   }),
   juicesflavours: many(juicesflavours),
}))

export const jfrels = relations(juicesflavours, ({one, many}) => ({
   products: one(products, {
      fields: [juicesflavours.id],
      references: [products.id]
   }),
}))

export const orderrels = relations(orders, ({many}) => ({
   carts: many(carts)
}))
export const cartrels = relations(carts, ({one}) => ({
   orders: one(orders, {
      fields: [carts.id],
      references: [orders.id]
   })
}))

export const contrels = relations(foodcont, ({one}) => ({
   products: one(products, {
      fields: [foodcont.id],
      references: [products.id],
```

```
    })
})))
```

ДОДАТОК К

Файл index.js:

```
import { drizzle } from 'drizzle-orm/mysql2';
import mysql from 'mysql2/promise';
import * as schema from './schema.js';
import { env } from '$env/dynamic/private';

if (!env.DATABASE_URL) throw new Error('DATABASE_URL is not set');

const client = await mysql.createConnection(env.DATABASE_URL);

export const db = drizzle(client, { schema, mode: 'default' });
```

ДОДАТОК Л

Файл cartStore.js:

```
import { writable } from 'svelte/store'
import { browser } from '$app/environment'

export const cart = writable([])

if (browser) {
   try {
      const storedCart =localStorage.getItem('cart')
      if (storedCart) {
         const parsedCart = JSON.parse(storedCart)
         if (Array.isArray(parsedCart)) {
            cart.set(JSON.parse(storedCart))
         } else {
            console.warn("Cart was not an array.")
            cart.set([]);
         }
      }

   } catch(e) { console.error("Error parsing from localStorage: ",e); cart.set([]) }

   cart.subscribe(value => { console.log(value)
      localStorage.setItem('cart', JSON.stringify(value))
   })
}

export function add(smth) {
   cart.update(c => {
      let n
      let flavouid = JSON.parse(localStorage.getItem('flavouid'))
      console.log('flavouid', flavouid)
      if (c.find(it => it.id === smth.id)) {

         if(smth.flavid) { //drinks with flavour
```

```
            n = c.map(item => item.id === smth.id ? smth.flavid in item.flavour ?
//just to be safe
                {...item, quant: item.quant+1, flavour: {...item.flavour, [smth.flavid]:
                   {          name:          flavouid[smth.flavid],          quant:
item.flavour[smth.flavid].quant+1} }}:
                {...item, quant: item.quant+1, flavour: {...item.flavour, [smth.flavid]:
                   { name: flavouid[smth.flavid], quant: 1} }} : item
            )
         } else { //other items
            n = c.map(item => item.id === smth.id ? { ...item, quant: item.quant+1 }
: item)
         }

      } else {
         if(smth.type === 'drink_juice') {

            n = [...c, { ...smth, flavour: {...smth.flavour,
               [smth.flavid]: { name: flavouid[smth.flavid], quant: 1 }}
            }]

         } else {
            n = [...c, { ...smth, quant: 1}]
         }
      }
      return n
   })
}

export function remove(smthID) {
   cart.update(c => c.filter(item => item.id !== smthID))
}

export function flavChange(smthID, flavId, q) {
   const newquant = q
   let qq

   cart.update(c => {
      const n = c.map(item => {
```

```
            if(item.id === smthID && item.flavour) {
                const flav = {...item.flavour}
                qq = flav[flavId].quant
                if(q===0) {
                    delete flav[smthID]
                } else {
                    flav[flavId] = {...flav[flavId], quant: q}
                }
                if(Object.keys(flav).length===0) {
                    return null;
                } else {
                    return {...item, quant: (item.quant-qq+q), flavour: flav}
                }
            } else {
                return item
            }

        }).filter(item => item !== null)
        return n
    })
}

export function removeFlav(smthID, flavid) {
    cart.update(c => {
        const n=[]
        for (const item of c) {
            if(item.id === smthID && item.flavour) {
                const newfl = {...item.flavour}
                delete newfl[flavid]
                if(Object.keys(newfl).length>0) {
                    n.push({...item, flavour: newfl })
                } else {
                    n.push(item)
                }
            }
        }
        return n
    })
```

```
    console.log('cart:', cart)
}

export function clearCart() { cart.set([]) }
```

ДОДАТОК М

Файл .env:

```
DATABASE_URL="mysql://root:ewfuyerfinnefju@localhost:3306/pizza_schema
"
```

ДОДАТОК Н

Тези у збірнику наукових праць

ІНФОРМАЦІЙНА ОНЛАЙН-СИСТЕМА ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ПІЦЦЕРІЇ

Ціль сталого розвитку №9 Інновації та інфраструктура

Зайцева А. М. azaitseva938@gmail.com
Таврійський державний агротехнологічний університет імені Дмитра Моторного

У сучасному світі успішний бізнес у сфері громадського харчування неможливо уявити без цифрової трансформації. Через значні переваги до конкурентоспроможності, багатьом компаніям необхідно мати власний сайт. І саме такі сайти утворюють сферу електронної торгівлі, або e-commerce Для піццерій, як і для інших закладів швидкого обслуговування, критично важливою є швидкість прийняття замовлень, точність обліку й ефективна логістика доставки. Метою роботи є розробка веборієнтованої інформаційної системи, яка дозволяє автоматизувати ключові бізнес-процеси піццерії. Більшість готових рішень на ринку або надто дорогі, або не дозволяють повноцінно адаптуватися під потреби малого бізнесу. Тому актуальним та перспективним є створення власної онлайн-системи з відкритим кодом, яка дозволить автоматизувати замовлення, вести облік, керування меню та аналітику для піццерії.

В результаті розроблений проєкт буде представляти собою сайт, за допомогою якого користувачі зможуть без зайвих зусиль здійснювати замовлення, переглядати меню та оновлення в закладі. Розроблюваний сайт повинен мати сучасний дизайн та зручну навігацію. Він має надавати можливість користувачу, за необхідності, зв'язатися з закладом, переглянути його розміщення на карті міста; переглядати меню, здійснювати онлайн замовлення, має відображати актуальні акції та пропозиції.

Середовищем розробки для онлайн-системи був обраний фреймфорк SvelteKit. SvelteKit - це потужний фреймворк для створення веб-додатків, який поєднує у собі простоту та продуктивність фреймворку Svelte з додатковими можливостями. Це офіційний фреймворк для створення програм на основі Svelte. SvelteKit заснований на принципах Svelte, компонентного фреймворку, який компілює код вашої програми у високоефективний JavaScript-код у процесі складання. Однією з основних переваг SvelteKit є його здатність генерувати статичні сайти, що покращує час завантаження сторінок та забезпечує SEO-

оптимізацію.. SvelteKit також підтримує рендеринг на стороні сервера (SSR), що дозволяє значно покращити взаємодію користувача з програмою.

Для поєднання до бази даних MySQL була використана система керування базами даних Drizzle. Drizzle - система керування реляційними базами даних, з відкритим вихідним кодом.. Drizzle розроблений з урахуванням сучасних вимог до розробки додатків та дозволяє легко інтегруватися з різними фреймворками та бібліотеками. Управління даними за допомогою Drizzle спрощує завдання, пов'язані зі зберіганням та обробкою інформації про замовлення, клієнтів та запаси.

Функціонал системи

1) Клієнтський інфтерфейс: каталог піц, конструктор замовлення
2) Система бронювання часу доставки.
3) Реєстрація та авторизація користувачів, накопичувальні бонуси
4) Адмін-панель: управління меню,перегляд замовлень.

Запропонована система дозволяє автоматизувати роботу піццерії, мінімізувати помилки, прискорити обробку замовлень, отримати аналітику та поліпшити якість обслуговування

Список використаних джерел

1. Методи розробки сайтів URL: https://webstudio2u.net/ua/webdesign/354-sitedevelop-methods.html (дата звернення 27.04.2025).
2. What’s the deal with SvelteKit? URL: https://svelte.dev/blog/whats-the-deal-with-sveltekit (дата звернення 27.04.2025).

Науковий керівник: *Зінов’єва О.Г., ст. викладач кафедри КН Таврійського державного агротехнологічного університету імені Дмитра Моторного*