Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему: «Розробка програмних засобів для інтерактивного обслуговування закладів харчування»

Виконав: студент II курсу

групи 1ПІ-19м

спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)

Дмитрук А.О.

(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Войтко В.В.

(прізвище та ініціали)

Рецензент: д.т.н., проф. кафедри КН Василевський О.М.

(прізвище та ініціали)

ВНТУ – 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 - «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
"17" лютого 2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дмитруку Анатолію Олександровичу

1. Тема роботи: розробка програмних засобів для інтерактивного обслуговування закладів харчування.

Керівник роботи: Войтко Вікторія Володимирівна , к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 9 березня 2021 року № 65

2. Термін подання студентом роботи: 1червня 2021року

3. Вихідні дані до роботи: операційна система додатку – Windows 7/8/8.1/10. Операційна система розробки – Windows Server 2016. Фреймворк – Wordppress. Мова програмування – C++, PHP, JavaScript, TypeScript. Середовища розробки – MVS 2017 Community, Visual Studio Code.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ; аналіз предметної області та обґрунтування доцільності розробки; розробка структури програмного Web-системи; опис програмних реалізацій платформи; тестування Web-системи; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): мета, предмет та об'єкт дослідження; актуальність розробки; розробка структури програмного продукту; алгоритм оформлення замовлення; вигляд головної сторінки Web-системи; адаптивна версія Web-системи; сторінка оформленя замовлення; сторінка ідентифікації замовлення; висновок.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	к.т.н., доц. каф. ПЗ Войтко В.В.		
5	Глущенко Л.Р., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання: 18 лютого 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області	20.02.2021–8.02.21	Виконано
3	Розробка методу, моделі та структури Web-додатку	01.03.20 – 12.03.21	Виконано
3	Розробка структури бази даних	13.03.21 – 22.03.21	Виконано
4	Розробка користувацького інтерфейсу веб-додатку	23.03.21 – 31.03.21	Виконано
5	Програмна реалізація Web-додатку	01.04.21 – 25.04.21	Виконано
6	Тестування роботи Web-додатку	01.05.21 – 16.05.21	Виконано
7	Оформлення матеріалів до захисту МКР	17.05.21–31.05.21	Виконано

Студент _____ **Дмитрук А.О.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи к.т.н., доц. каф. ПЗ Войтко В.В.

_____ _____

(підпис) (прізвище та ініціали)

Рецензент магістерської кваліфікаційної роботи **д.**т.н., проф. каф. КН _____ Василевський О.М.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі розроблено Web-систему для замовлення їжі онлайн, а також реалізована структура захисту IDS/IPS. В структурі захисту реалізована система передбачення і захисту від кібератак методом моніторингу мережі і додатків, що підключені до IDS/IPS. У Web-системі реалізовано додавання позицій у меню, можливість формування замовлення клієнтом, фільтрація по популярності продукту, інтерфейс для користувача, інтерфейс для оператора, який керує замовленнями, інтерфейс для кур'єра.

Платформу реалізовано засобами фреймворку WordPress з використанням мови програмування C++, PHP і JavaScript, відповідно. Використано інтегроване середовище розробки Microsoft Visual Studio 2017, Visual Studio Code.

SUMMARY

In the thesis, a Web-application for ordering food online was developed, as well as the IDS / IPS protection structure was implemented. The protection structure implements a system for predicting and protecting against cyberattacks by monitoring the network and applications connected to IDS / IPS. The Web-application implements the addition of menu items, the ability to generate orders by the customer, filtering by product popularity, user interface, interface for the operator who manages orders, the interface for the courier.

The platform is implemented using the WordPress framework using the C ++, PHP and JavaScript programming languages, respectively. The integrated development environment of Microsoft Visual Studio 2017, Visual Studio Code is used.

ЗМІСТ

ВСТУП

Обгрунтування вибору теми актуальності дослідження. Їжа є однією з найголовніших потреб людини. У наш час завдяки нарощуваннютемпу життя, людям не вистачає часу навіть поспати, не те що б займатись приготуванням їжі самостійно.

Сьогодні стрімкий розвиток технологій дав змогу забути ті часи коли Web-сайти складались зі статичного контенту, і приніс в сферу ресторанного бізнесу, багато інновацій, зокрема таку, як замовлення їжі через Web-додатки. Даний підхід дає змогу клієнту швидко задовольнити одну зі своїх потреб, а саме замовити їжу, зокрема додому. Тому однією з основних проблем для людей які пов'язані з ресторанним бізнесом є створення, вдосконалення Web-додатків, який мав би простий та зрозумілий інтерфейс, наділений всіма потрібними функціями для замовлення. Створення Web-додатків з кожним роком набуває все більшої популярності, адже спрямоване на полегшення роботи користувача та надає постійний доступ до потрібних даних, та дуже економить час на саму реалізацію[1]. Тому використання Web-додатку для замовлення їжі онлайн. Тому розробка і використання інтерактивного Web-додатку для замовлення їжі онлайн є актуальними.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення ефективності процесу замовлення їжі онлайн за рахунок розробки і використання спеціалізованого Web-ресурсу, що дозволить автоматизувати процеси вибору їжі, оплати, замовлення та доставки.

У відповідності до поставленої мети потрібно виконати такі задачі:

- провести аналіз інформаційного забезпечення Web-системи;
- розробити метод, модель та алгоритм роботи Web-системи;
- розробити програмну частину Web-системи;

- провести тестування Web-системи харчування.

Об’єктом дослідження є процес розробки тематичного сайту для можливості замовлення їжі онлайн.

Предметом дослідження є методи і засоби побудови Web-сайту закладу харчування.

Методи дослідження. У процесі дослідження використовувались такі методи:

- метод розробки Web-ресурсів для створення Web-сайту закладу харчування;
- методи роботи з базами даних для побудови бази даних системи для замовлення їжі онлайн;
- метод тестування для тестування Web-системи.

Наукова новизна:

1. Подальшого розвитку дістав метод підвищення інтерактивності обслуговування закладів харчування в онлайн режимі, який, на відміну від існуючих, орієнтований на створення і використання інтерактивних об’єктів інтерфейсу, що дозволяє забезпечити ефективний зворотній зв’язок з користувачем.
2. Подальшого розвитку дістали моделі реалізації Web-системи, які, на відміну від існуючих, використовуюсь розширення категоризацію об’єктів та орієнтовані на використання інтерактивної взаємодії з користувачем у процесі вибору об’єктів і формуванн кошика замовлення, що дозволяє розширити можливості Web-системи з урахуванням зворотнього зв’язку з користувачем.

Практична цінність отриманих результатів. Практична цінність результатів полягає в тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень розроблено програмні засоби для онлайн замовлення їжі в закладах харчування, що розширює перспективність практичного використання створеної Web системи.

Особистий внесок здобувача. У магістерській кваліфікаційній роботі усі результати дослідження здобуті автором даної роботи самостійно.

У публікації [2] автору належать загальна розробка методу та моделі роботи Web-системи закладу зарчування.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення й результати досліджень представлені на LXVII Міжнародна інтернет-конференції «Перспетивні напрямки наукових досліджень». м. Рівне, 31 травня 2021 р.

Публікації. За тематикою дослідження опублікована 1 наукова публікація у матеріалах LXVII Міжнародної інтернет-конференції «Перспективні напрямки наукових досліджень» [3].

Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел і додатків, у які винесено технічне завдання, лістинг програми й ілюстративний матеріал до захисту роботи.

1 АНАЛІЗ СТАНУ РОЗВИТКУ СПЕЦІАЛІЗОВАНИХ WEB-РЕСУРСІВ І ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану розвитку Web-ресурсів закладів харчування

Як відомо, одним з найпоширеніших видів інтернет-ресурсів є Web-сайти. Вони забезпечують доступ до інформації мільярдам людей по всій земній кулі. З появою всесвітньої павутини, здавалося б звичні справи, як от купівля речей чи бронювання квитків, стали можливими для людей прямо з їх помешкання. Це значно полегшило життя людей, зробило його комфортнішим, та виділило більше часу для інших справ. Розвиток онлайн замовлень є кроком для поступового переходу людської взаємодії на цифровий рівень.

Як було вже зазначено, одним з видів онлайн-замовлень, які в останній час активно розвиваються є сайти з доставки їжі. Подальшого розвитку дістала модель WEB-системи для замовлення їжі онлайн яка, на відміну від існуючих орієнтована на використання інтерактивних об’єктів в процесі розробки WEB-додатку, що підвищує зручність формування замовлення користувача.

Вони мають наступні переваги:

- ресторан зможе обслуговувати клієнтів, навіть якщо ті знаходяться на іншому кінці міста;
- економія часу та сил користувачів;
- можливість легко знайти для себе бажані страви. Адже такі сайти включають в себе можливість фільтрації;
- невисока вартість. Адже, як правило, при відвідуванні ресторану у вартість страв входить обслуговування офіціанта, жива музика, тощо;

- доступність. Більшість сервісів з доставки їжі працюють цілодобово.

Їх вид залежить напряму від формату бізнесу фірми. Це може бути лише співпраця з підприємствами, які безпосередньо виробляють страви, або ж власне виробництво їжі та її доставки [4].

Виходячи з поставленої задачі було визначено основну структуру сайту, який є безпосередньо Web-додатком з можливістю замовлення та доставки.

При розробці потрібно звернути увагу на такі особливості:

- простий дизайн, з безліччю яскравих фото їжі, що змусить користувача зробити замовлення;
- мінімум зайвої інформації, та інтуїтивний інтерфейс для легкого оформлення замовлення;
- достовірна та зрозуміла інформація про продукти, їх вартість, та умови доставки;
- буде перевагою функція онлайн-консультанта, що дасть змогу оперативно зв’язуватися з клієнтами та встановлювати зворотній зв’язок з ними;
- можливість користувачу власноруч змінювати інгрідієнти, в залежності від бажань.

У зв’язку з великою популярністю Web-додатки для замовлення їжі онлайн є актуальною проблемою для ресторанного бізнесу. Саме тому є доцільною розробка онлайн-платформи для для замовлення їжі онлайн.

1.2. Аналіз аналогів веб-додатку

Як орієнтир, проведемо порівняння декількох Web-сайтів по доставці їжі. Це: eda.ua, h-and-h.com.ua , sushi33.ua та mamafood.com.ua

Сайт eda.ua (рис. 1.1) виконаний в червоно-білих тонах. На білому фоні легко сфокусовувати увагу відвідувача.На головній сторінці, зверху присутнє

меню з вибором міста, а також пошукова стрічка. Зправа прикріплена опція онлайн-допомоги. До вибору надається декілька видів їжі: піцца, суші, бургери, шашлики, локшина та рис. При виборі одного з виду страв, ми переходимо на наступну сторінку, де зліва доступна детальна фільтрація та безпосередньо вибір закладу, звідки буде доставлятися їжа.

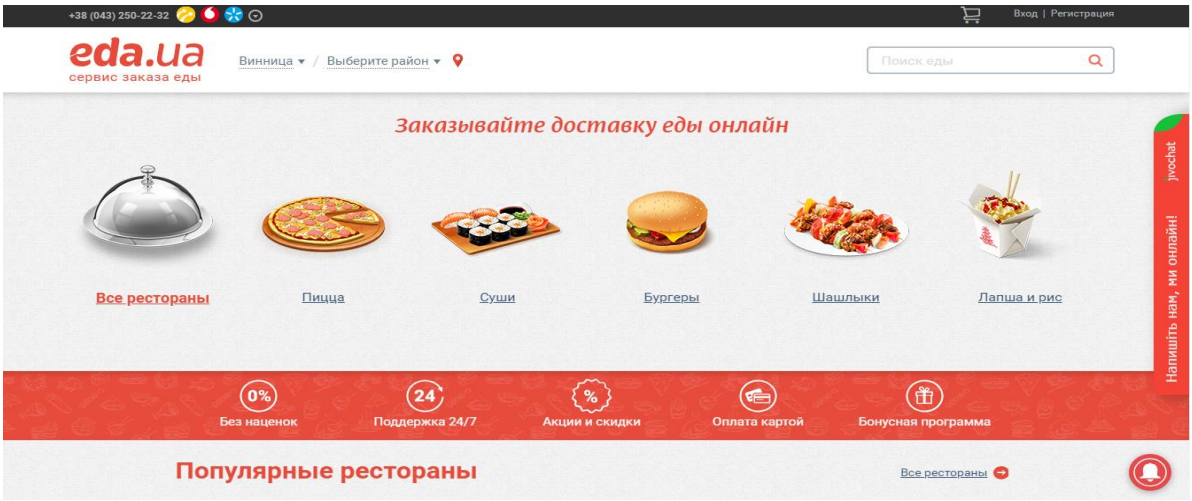

Рисунок 1.1 – Дизайн сайту eda.ua

Наступний розглянутий Web-сайт – це h-and-h.com.ua (рис. 1.2). Орієнтований на доставку здорової їжі, тому дизайн у нього зелено-білий. Зверху наведений розклад роботи, та мобільний номер для замовлень.Головне меню складається з таких пунктів: бургери, вегетеріанське меню, локшина,WOK , бізнес-ланч, десерти, салати.

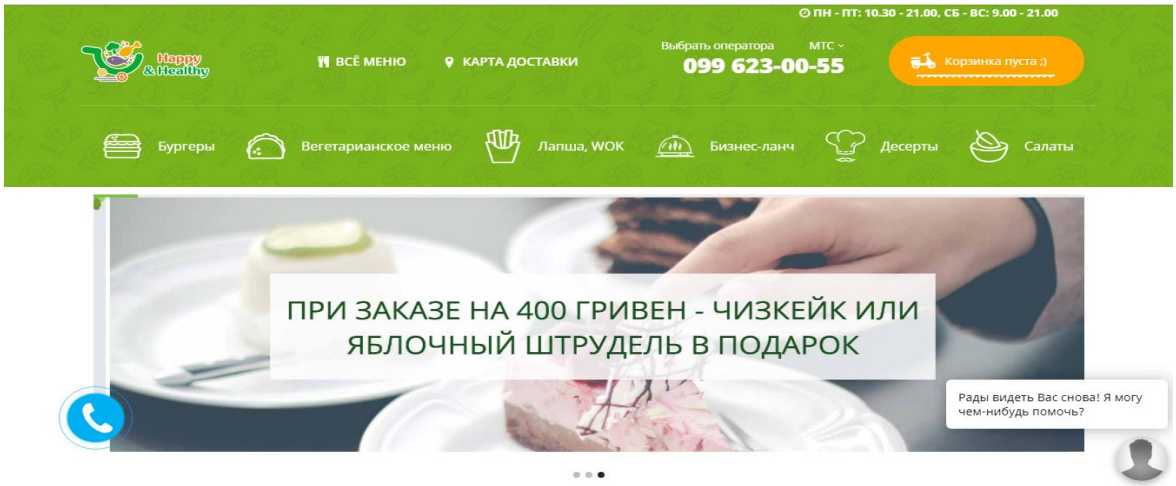

Рисунок 1.2 – Дизайн сайту h-and-h.com.ua

Третій до розгляду sushi33.ua (рис. 1.3) – має зверху контактну інформацію, вибір мови (українська, російська та англійська) Зразу під – знаходиться основне меню з різновидами продукції. Ще нижче – великі, яскраві банери з акціями, інформацією щодо покупки, тощо. Далі присутня стрічка зі закликом до приєднання в соціальних мережах.

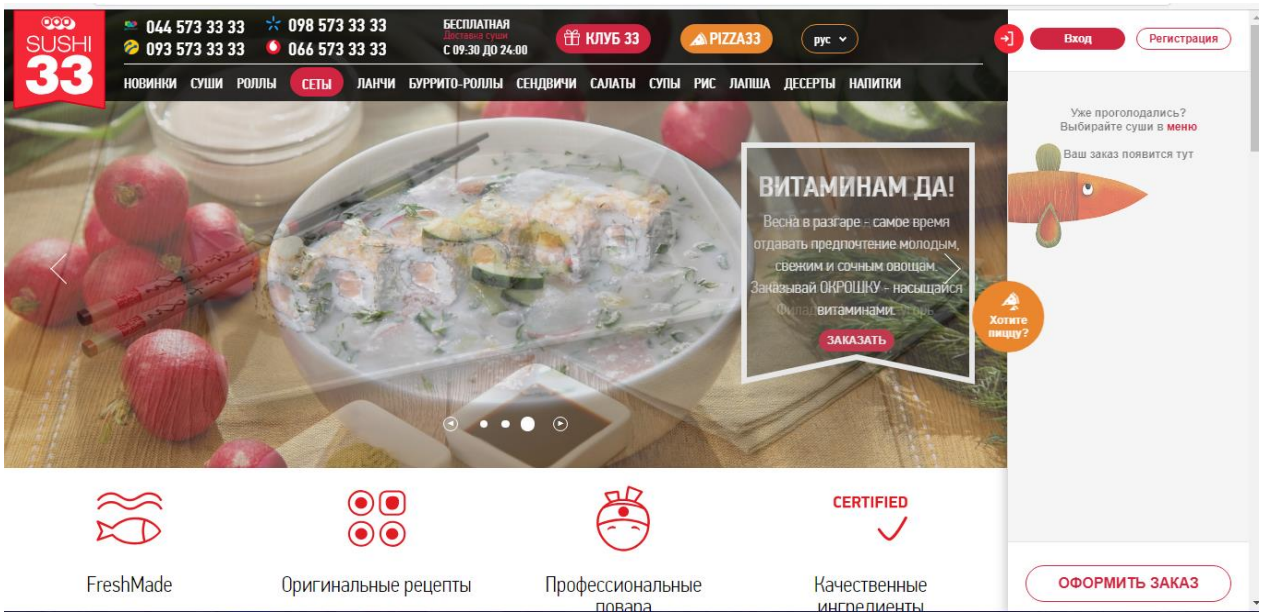

Рисунок 1.3 –Дизайн сайту sushi33.ua

Останнім є mamafood.com.ua (рис. 1.4). Він – найпростіший з точки зору дизайну, в порівнянні з вище згаданими веб-сайтами. Зверху присутня зелена стрічка з усіма контактами організації, а також годинами роботи. Знизу присутнє не зовсім якісно розміщене прозоре меню, яке закриває частину емблеми організації зліва, та частину текста розділу «Про нас». З самого низу присутні емблеми, які дозволяють інтегруватися з соцмережами. Основний функціонал зосереджено в полі «Меню» зверху. Де є можливість сортування за:

- оцінкою;
- популярністю;
- новизною;
- ціною, додатково можна вказати діапазон цін.

Також можна вказати кількість страв, що будуть відображатися. Недоліком є те, що на деяких стравах відсутнє її зображене, натомість відображається головна емблема компанії.

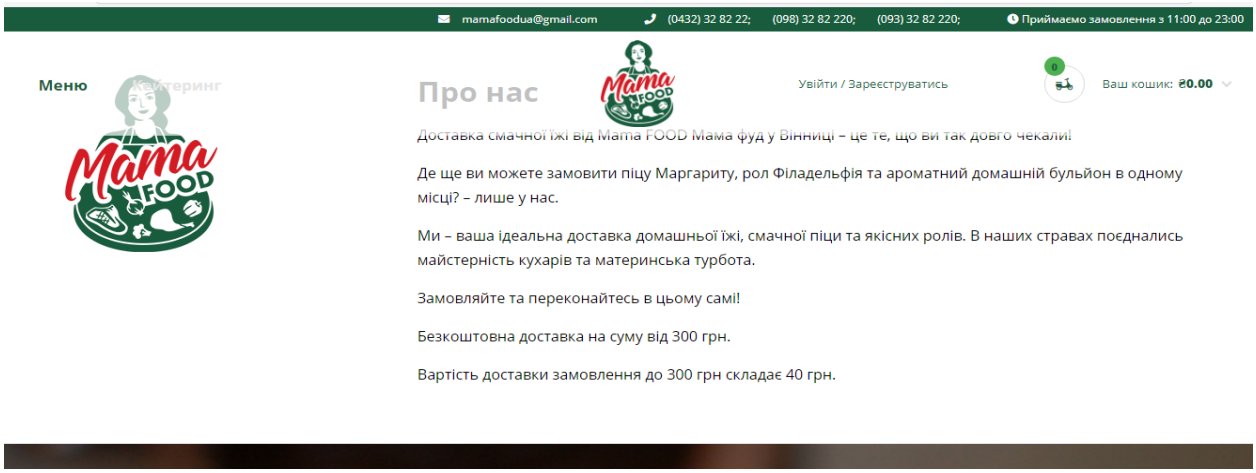

Рисунок 1.4 – Дизайн сайту mamafood.com.ua

Порівняльну характеристику вищевказаних веб-сайтів аналогів наведено у таб. 1.1.

Таблиця 1.1 – Порівняльна характеристика веб-сайтів аналогів

Критерії	**eda.ua**	**h-and-h.com.ua**	**sushi33.ua**	**mamafood.com.ua**	**Розроблений додаток**
Онлайн-помічник	+	+	-	-	+
Багатомовність	-	-	+	-	+
Інформативність	+	+	+	-	+
Наявність опції відгуків	+	+	+	-	+
Широка фільтрація	+	-	-	+	+
Дизайн	+	+	-	-	+

1.3. Теоретичні основи розробки Web-додатку

Дляє підвищення ефективності процесу замовлення їжі онлайн потрібно розробити спеціалізований Web-ресурс, що дозволить автоматизувати процеси вибору їжі, оплати, замовлення та забезпечити доставку їжі кур'єром. При розробці програми необхідно врахувати визначений функціонал системи:

- Розробити метод реалізації Web-ресурсів для створення Web-сайту закладу харчування, який має бути орієнтований на забезпечення інтерактивної взаємодії користувачів.
- Розробити базу даних Web-системи з забезпеченням категоризації об'єктів для здійснення замовлення їжі онлайн і формування кошика замовлення.
- Розробити структуру сайту з урахуванням необхідності використання об'єктів з інтерактивним функціоналом для забезпечення дієвого взаємозв'язку з віддаленими користувачами.
- Розробити користувацький інтерфейс Web-системи з розподілом функціоналу для авторизованих користувачів: для адміністраторів закладу харчування, які забезпечують наповнення й оновлення бази даних, для клієнтів, які формують кошик власного замовлення і здійснюють онлайн оплату продуктів і послуг, та для кур'єрської служби, що забезпечує доставку замовлення за вказаною адресою.
- Програмно реалізувати Web-систему закладу харчування і провести її тестування.

1.4. Постановка задач магістерської кваліфікаційної роботи

Після аналізу аналогів було вирішено скласти список завдань, необхідних для створення програмного продукту.

Головною задачею роботи є створення Web-системи, що дозволить замовити їжу онлайн, для чого необхідно виконати такі задачі:

– розробку методу і моделі роботи Web-системи для замовлення їжі;

– розробку інтерфейсу Web-системи з інтерактивною взаємодією користувачів;

– розробку програмного забезпечення циклічного вибору страв з обраних категорій меню та формування кінцевого кошика;

– проєктування архітектури бази даних веб-системи;

– налаштування серверного оточення для розміщення Web-системи;

– тестування режимів роботи розробленої Web-системи закладу харчування.

1.5. Висновки

У даному розділі визначено, на що потрібно звертати увагу при розв’язанні задачі. Проведено порівняння існуючих розв’язків на прикладі декількох Web-сайтів. А також проведено аналіз деяких існуючих методів розв'язання задачі і обгрунтовано доцільність створення Web-системи закладу харчування.

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ І АЛГОРИТМІВ РОБОТИ WEB-РЕСУРСУ

2.1 Аналіз інформаційного забезпечення системи

Для того, щоб різні інформаційні системи, з різними користувачами могли взаємодіяти між собою, в свою чергу обмінюючись власними даними, потрібно вирішити проблему їх інформаційної сумісності, шляхом створення інформаційного забезпечення. Тому його розробка є дуже важливою, для побудови повноцінної інформаційної системи.

Зазвичай цей процес [5] включає в себе створення документів, нормативної бази і реалізацію рішень щодо обсягу, розміщення і форм організації інформації, яка циркулює в системі автоматизованого оброблення економічної інформації чи в інформаційній системі.

Створення інформаційного забезпечення забезпечує єдність і зберігання інформації, необхідної для розв’язання задач, єдність інформаційних масивів для всіх задач інформаційних систем, однократність уведення інформації та її багатоцільове використання, різні методи доступу до даних, низьку вартість витрат на зберігання та використання даних, а також на внесення змін.

Основними принципами створення інформаційного забезпечення є:

- контроль та захист від несанкціонованого доступу;
- єдність і гнучкість, стандартизація та уніфікація;
- адаптивність та мінімізація помилок введення-виведення інформації.

Веб-додаток «Замовлення їжі онлайн» отримуватиме вхідні дані від користувача, шляхом введення їх з клавіатури, або натисканням кнопок у веб-інтерфейсі. Введені таким чином дані зберігаються у базі даних «MSSQL».

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

2.2 Розробка методу та моделей Web-ситеми

Платформа являє собою цілу функціональну систему, ділиться на декілька частин:

- Основна база даних – збереження головної інформації, яка обробляється платформою.
- Веб-додаток – реалізація взаємодії користувача і платформи (рис.2.1).

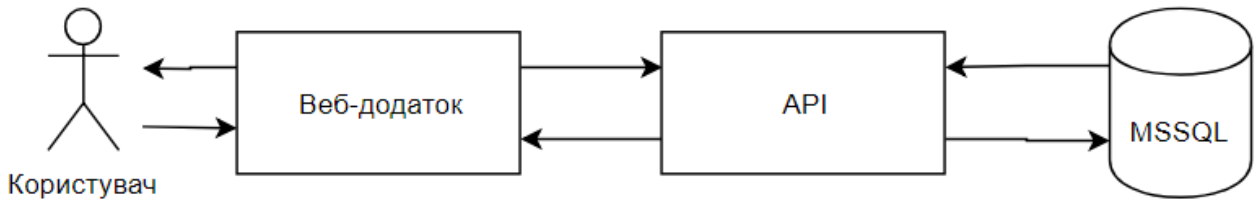

Рисунок 2.1 – Архітектура платформи

Веб-додаток – головна частина створюваної онлайн-платформи, яка бере на себе завдання керування основними функціями системи. Саме з ним взаємодіє користувач. Веб-додаток оперує з даними про замовлення, які в свою чергу зберігаються в головній базі даних платформи.

Окрім користувача та основної бази даних, веб-додаток також взаємодіє з модулем виявлення повторюваних запитань. Відокремлення модуля як окремої компоненти, замість розробки його як складової веб-

додатку, має на меті ослаблення зв’язності компонентів програмної системи. Таким чином, при внесенні змін у модуль виявлення повторюваних запитань, не потрібно вносити зміни також і у веб-додаток.

Поширеним архітектурним шаблоном, що застосовується при проектуванні та розробці веб-додатків є шаблон MVC (англ. «Model – View – Controller» – «Модель – Представлення – Контролер») [5]. Основною характерною рисою цього шаблону є поділ додатку на три складові: модель даних, що містить всю бізнес-логіку; представлення, тобто спосіб виведення результату (HTML-сторінка, XML-документ, JSON-документ тощо); модуль керування, що контролює поведінку додатку залежно від вхідних даних (рис2.2).

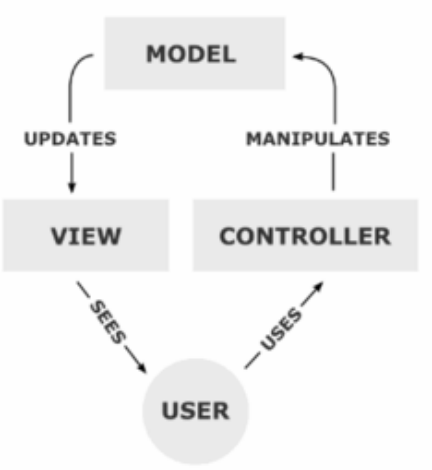

Рисунок 2.2 – Модель MVC

Перевагами цього шаблону є: можливість повторного використання коду, масштабованість (за рахунок чіткої структури додатку), а також можливість застосування модульного тестування. Саме тому при розробці веб-додатку було використано архітектурний шаблон MVC.

Структура сайту – основна дл вибудовування послідовності і форми відображення наявних даних на сайті.При правильній структурі сайту користувачам максимально зручно переходити від однієї сторінки до іншої і вивчати необхідні для них відомості.

Зовнішня структура включає в себе розташвання видимих блоків на сайті (шапка , сайдбари, футер, інформери , службові форми і інші блоки).

Внутрішня структура включає в себе приналежність матеріалів до певних категорій , акатегорій до розділів (іншими словами – рубрикацію), а також кількість посилань зв'язку сторінок. На деяких джерелах рубрикацію назиають логічною структурою.

Розробкаметоду роботи Web-системи замовлення їжі онлайн:

- користувач має авторизуватись в системі;

- забезпечення можливості створення замовлення шляхом циклічного вибору страв з обраних категорій меню та формування кнцевого кошика.При виборі страв використовується динамічні об'єкти які дають змогу створити замовленяя шляхом виділення страв, вказаної кількості в інтерактивному інтерфейсі користувача;

- забезпечити можливість дистанційної оплати замовлення ;

- забезпечити можливість доставки замовлення.

Загальна модель Web-системи акомулює взаємоз'язок базових модулів системи(рис.2.3): модуль реєстрації; модуль бази даних; модуль формування; модуль оплати; модуль замовлення.

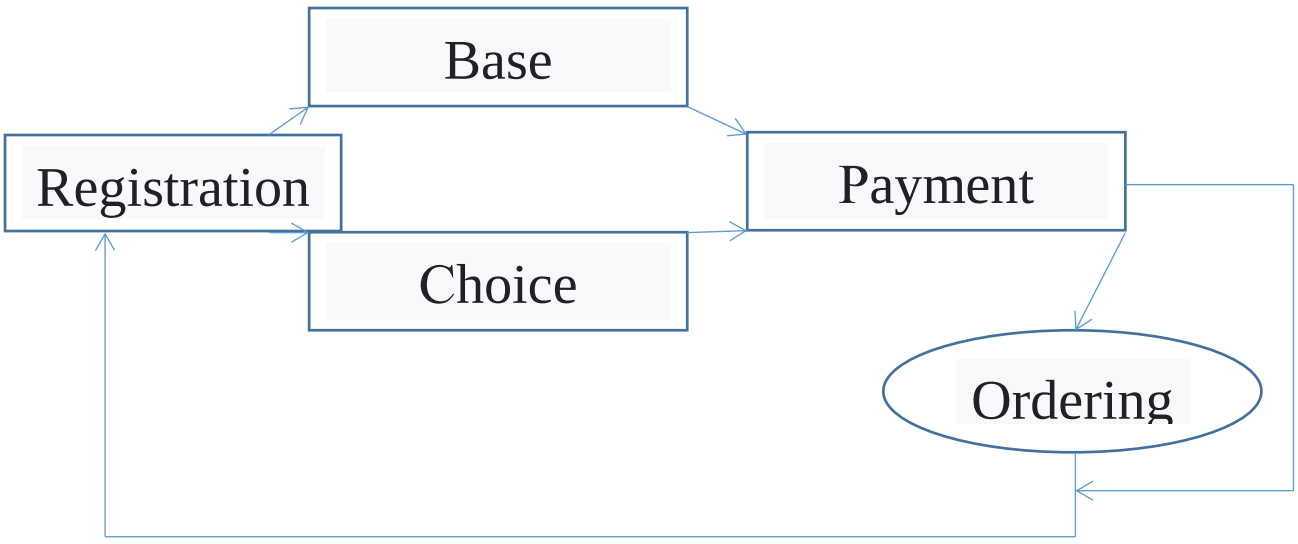

Рисунок 2.3 – Модель Web-системи

2.3 Розробка структури бази даних

Розроблювана платформа містить єдину основну базу даних, яка містить інформацію про такі сутності:

1. Products – дані про певну страву (Назва, опис, ціна);
2. ProductCategories – опис категорії продуктів;
3. OrderProduct – замовлення продукту (Ідентифікатор замовлення і продукту);
4. Orders – дані щодо замовлення (Адреса, час доставки, статус, тощо).

Описані вище сутності тісно пов’язані між собою, і їх взаємодія лежить в основі роботи веб-додатку.

Для розробки універсального відношення постає задача вибору інформа- ційних об’єктів та визначенні їх найбільш вагомих характеристик. Одне універсальне відношення містить усі найвагоміші атрибути – тобто характерна властивість інформаційного об’єкту. Також воно може містити усі дані, які в майбутньому також будуть розміщені.

При створенні універсального відношення можуть виникнути наступні проблеми:

1. аномалія оновлення;
2. аномалія видалення;
3. аномалія включення;
4. надлишковість даних.

Аномалія оновлення полягає в тому, що в разі внесення змін, які стосуються одного об’єкту, потрібно змінювати і всі дані, що дублюються у різних кортежах[6].

При видаленні кортежа, інформація яка зберігалася лише в ньому буде втрачена – це аномалія видалення.

Аномалія включення виникає при додаванні у базу даних про новий об’єкт, шляхом вставки нового кортежу, а в результаті дані про інші об’єкти ще невідомі, тому в новому кортежі будуть порожні поля.

Проблема надлишковості спричинена повторенням даних у базі. Універсальне відношення повинне містити атрибути наведені у таблиці 2.1.

Таблиця 2.1 – Список атрибутів універсального відношення

№	Назва поля	Коментар
1	ID Order	Номер замовлення
2	Customer Name	Ім’я користувача
3	Customer Phone Number	Телефон користувача
4	Customer Email	Пошта користувача
5	Address	Адреса доставки замовлення
6	Delivery Date	Дата доставки
7	Status	Статус замовлення
8	Comment	Особливі побажання
9	ID Product	Ідентифікатор страви
10	Product Name	Назва страви
11	Price	Ціна страви
12	Product Description	Опис страви
13	Category ID	Ідентифікатор категорії продуктів
14	Category Name	Назва категорії
15	Category Description	Опис категорії

ER-модель (сутність - зв'язок) – описує концептуальні схеми за допомогою узагальнених конструкцій блоків. В такій моделі існує можливість виділяти найменші вузли та встановлювати між ними зв'язки.

Базовим поняттям є ступінь зв'язку – те число сутностей, яке асоціюється через набір зв'язків з іншою сутністю.

Розрізняють такі види зв'язків:

1. Один до одного (1:1) – сутність, яка має одну роль, завжди відповідає інша сутність з однією роллю
2. Один до багатьох (1:N) – сутність, яка має одну роль відповідає довільне число сутностей з іншою роллю.
3. Багато до багатьох (M:N) – кожна з асоційованих сутностей може мати у відповідність будь-яку кількість сутностей з іншою роллю.

Проаналізувавши інформаційні об’єкти в базі даних (Рис. 2.4), виділено такі сутності разом з їх ключами та атрибути:

1. Продукти(Id продукту, Назва, Ціна, Опис, Id категорії);
2. Категорія продукту (Id категорії, Назва, Опис);
3. Замовлення (Id категорії, Ім’я замовника, Мобільний номер замовника, Електронна пошта замовника, Адреса , Дата доставки, Статус, Коментар);
4. Замовлення родукту (Id замовлення, Id продукту).

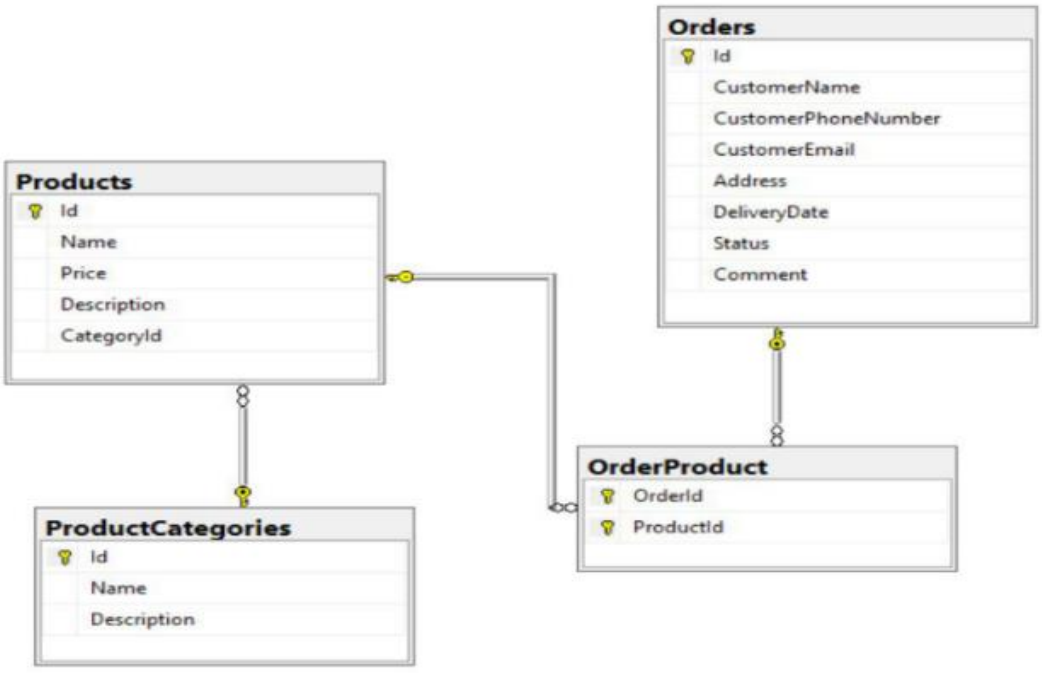

Рисунок 2.4 – Взаємодія між компонентами БД

2.4. Розробка користувацького інтерфейсу Web-додатку

Для платформи, яка розроблюється необхідно створити наступні сторінки:

1. Головна.
2. Категорії страв – відображення страв.
3. Корзина.
4. Сторінка оформлення замовлення.
5. Сторінка оператора – список необроблених замовлень.
6. Сторінка кур’єра.

Кожна сторінка ділитиметься на 2 частини (рисунок 2.5):

1. Шапка (header).
2. Тіло (body).

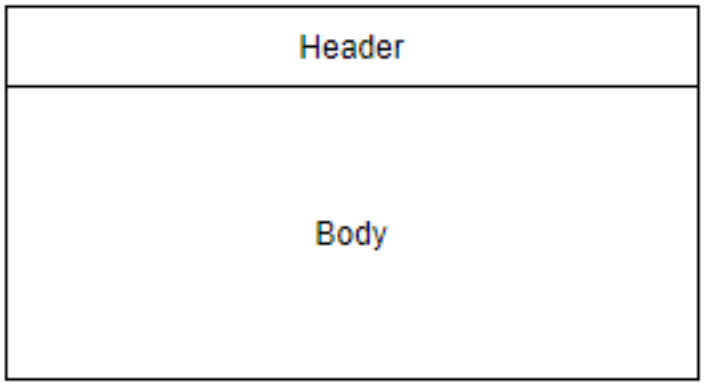

Рисунок 2.5 – Взаємне розміщення частин сторінки

Дизайн сайту створено у мінімалістичному стилі. Мінімалізм у веб-дизайні передбачає знищення всього зайвого, візуального і текстового, зосереджуючи увагу на простому та інтуїтивно зрозумілому представленні інформації (в текстовій чи візуальній формі), тобто спрощення як дизайну так і коду. Принцип «Чим менше, тим краще» вимагає від дизайнерів

використання якомога меншої кількості елементів, які можуть захаращувати веб-сайт. Це стосується і функціональності сайту: надмірність заплутує користувача, ускладнюючи його роботу на сайті [7].

Блок Header міститиме такі елементи, розташування яких наведено на рис. 2.6:

1. Емблема веб-додатку з його назвою;
2. Кнопка «Корзини».

Рисунок 2.6 – Структурна схема шапки веб-додатку

В блоці Body, розміщуватиметься основна інформація про можливі замовлення, і він міститиме в собі наступні блоки:

1. Список категорій страв;
2. Меню з варіантами страв, описом до них;
3. Кнопка додавання в корзину страв.

Структурна схема початкового блоку Body наведена на рис. 2.7.

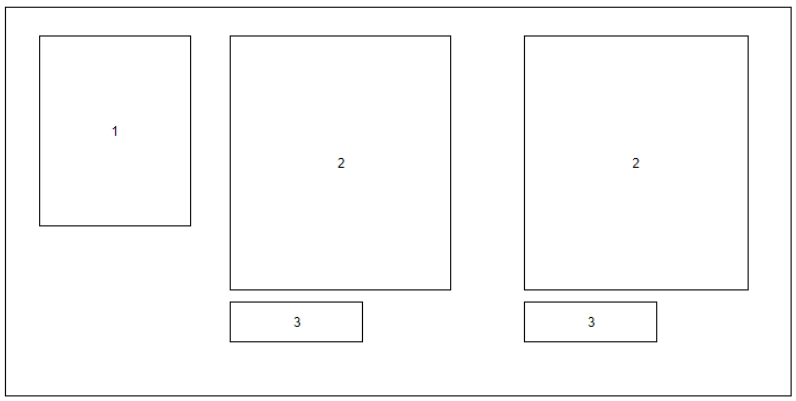

Рисунок 2.7 – Структурна схема тіла веб-додатку початкової сторінки

При додаванні страв у корзину, та натисканні кнопки «Корзина» у шапці сайту, блок Body дещо зміниться: над меню з’явиться інформація про поточний стан корзини (кількість доданих страв та ціна) – блок 4, а також кнопка очищення корзини – блок 5. А з самого низу з’явиться кнопка підтвердження замовлення –блок 6. Структурна схема після додавання страв у корзину зображена на рис. 2.8.

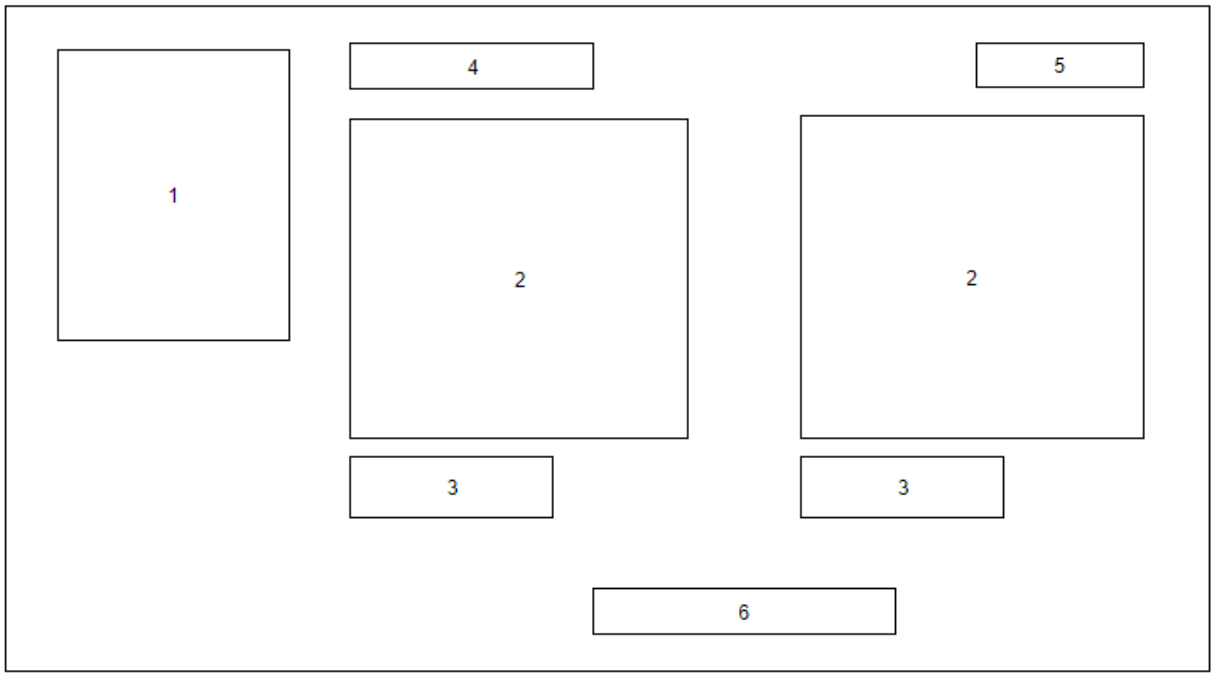

Рисунок 2.8 – Тіло головної сторінки веб-додатку, після додавання в корзину

Після нажаття кнопки підтвердження замовлення сторінка зміниться наступним чином: весь доступний простір займе поле в якому користувач повинен заповнити усі потрібні контактні дані, для прийому замовлення – блок 8.

Остаточно підтвердити замовлення користувач зможе кнопкою, що розташована з самого низу – блок 7 (рисунок 2.8).

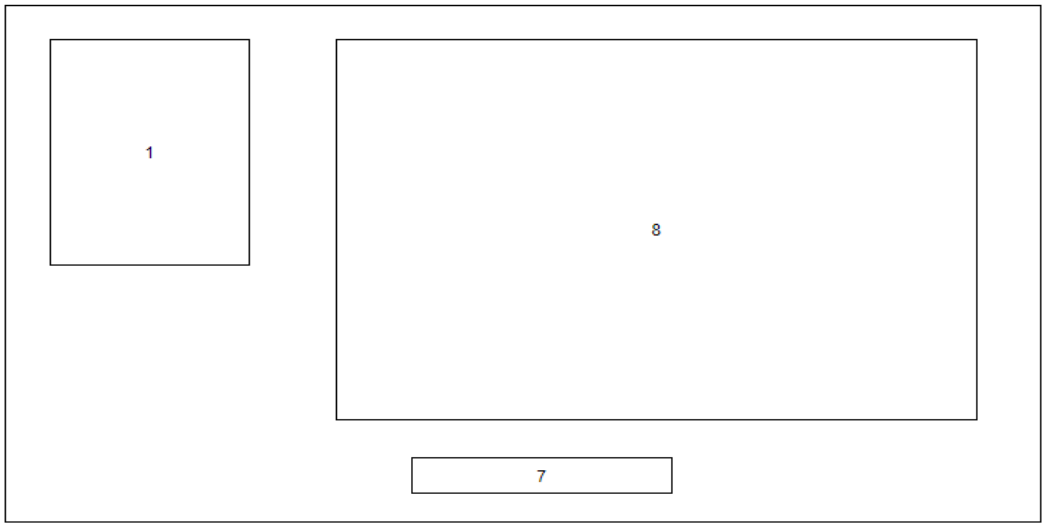

Рисунок 2.9 – Структура сторінки заповнення контактних даних

2.5 Розробка алгоритмів роботи програмного додатку та базових модулів

Алгоритм — набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій; система правил виконання дискретного процесу, яка досягає поставленої мети за скінченний час. Для візуалізації алгоритмів часто використовують блок-схеми.

Властивості алгоритму — це вимоги, які повинен задовольняти алгоритм.

Алгоритм складається із сукупності відокремлених одна від одної команд, кожна з яких виконується за кінцевий час. Тільки закінчивши виконання однієї команди, виконавець переходить до виконання іншої.

Однозначність. Кожна команда алгоритму однозначно визначає дії виконавця і не допускає їх подвійного тлумачення. Однозначним є й порядок виконання операцій.

Формальність. Будь-який виконавець, що володіє заданою системою команд виконавця, може, не знаючи суті завдання, виконати алгоритм й отримати такий самий результат.

Масовість. Алгоритм повинен передбачати можливість розв'язання групи типових задач, зміни вхідних (початкових) даних в деяких допустимих межах.

Скінченність. Алгоритм складається зі скінченної кількості дій, колена з яких виконується за скінченний проміжок часу. Виконання алгоритму не може закінчуватися невизначеною ситуацією або зовсім не закінчуватися.

Результативність. Виконання алгоритму при допустимих вхідних даних приведе до очікуваного результату [8].

Важливою складовою розроблюваної платформи, є модуль, який реалізує додавання страв до корзини, менеджмент замовлення та оформлення замовлення. Блок-схема алгоритм оформлення замовлення наведено на рисунку 2.10.

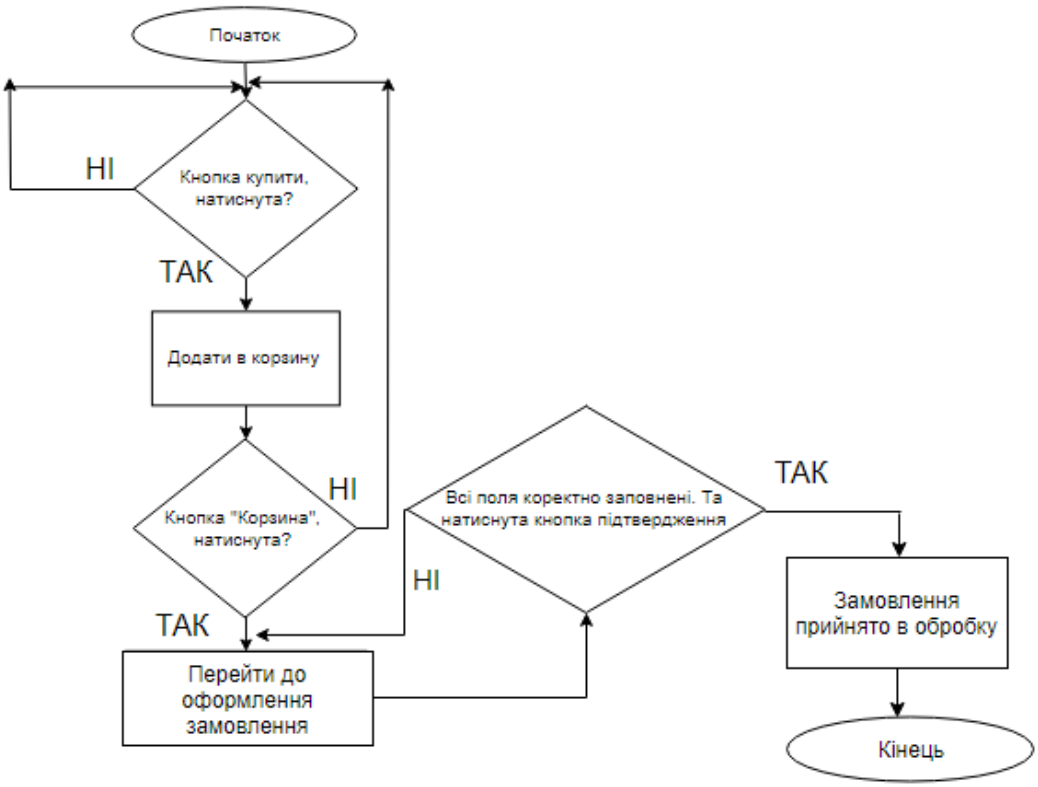

Рисунок 2.10 – Алгоритм оформлення замовлення

Таким чином, алгоритм — це однозначно визначена послідовність окремих команд, формальне виконання яких за кінцеве число кроків приводить до розв'язання задачі.

2.6 Висновки

У другому розділі роботи була розроблено метод, модель, структуру Web-системи закладу харчування, базу даних, яка буде основним сховищем оброблюваних даних. Також створено інтерфейс користувача Web-додатку та алгоритм оформлення замовлення. В результаті розробки основної бази даних, створене універсальне відношення та ER-модель предметної області.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-СИСТЕМИ ЗАКЛАДУ ХАРЧУВАННЯ

В цьому розділі розглянуто технології та підходи, які використовуються при розробці Web-системи закладу харчвання, а також представлені безпосередньо ті, які використовувались при розробці Web-додатку для замовлення їжі онлайн і обгрунтовано доцільність їх використання. Для порівняння описано технології, які застосовуються при розробці Web-клієнтів і при створенні серверних частин Web-додатків.

3.1 Варіантний аналіз і обґрунтування вибору програмних засобів реалізації Web-системи

На початку розробки Web-системи необхідно дослідити доступні програмні засоби та підібрати такі, що найкраще відповідають розробці.

Вибір програмних засобів буде відбуватися для таких складових платформи:

1. Веб-додаток.
2. API.
3. Основна база даних.

Для розробки веб-додатку та API буде проведено аналіз мов програмування, а саме: C++, C# та Java[21].

C++ – компільована мова програмування високого рівня, розроблена Б’ярне Страуструпом. Перша версія мови була випущена у 1979 році. Найактуальніша версія мови C++ 17 вийшла у 2017 році. C++ успадкувала синтаксис і основні аспекти мови C. По суті, C++ є надмножиною мови C, на що також вказує початкова назва – «C with classes» – «Сі з класами»[9]. Мова C++ підтримує парадигму об’єктно-орієнтованого програмування, характеризується високою швидкодією за рахунок того, що код програми компілюється у машинний код, а також має велику кількість сторонніх бібліотек, що спрощують процес розробки. До недоліків мови відноситься відсутність автоматичного керування пам’яттю («збір сміття» – «garbage

collection»), що є потенційно небезпечним, адже допускає існування помилок, що призводять до втрат пам’яті (явище, коли частина пам’яті, виділеної для виконання програми, не вивільняється після завершення її використання). Натомість, ручне керування пам’яттю надає розробнику більше контролю над роботою програми. Мова C++ не є кросплатформною. Програма, написана мовою C++ та скомпільована для операційної системи Windows, не працюватиме під керуванням операційної системи Linux – для цього її потрібно скомпілювати заново з допомогою спеціалізованого компілятора [10].

C# – мова програмування високого рівня, розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде в корпорації Microsoft. Перша версія мови була випущена у 2002 році. Найактуальніша версія мови C# 7.0 вийшла 7 березня 2017 року. Мова програмування C# була розроблена під впливом C++ та Java, має C-подібний синтаксис. Підтримує парадигму об’єктно-обієнтованого програмування. Є кросплатформною, оскільки код програми компілюється в IL (Intermediate Language), інша назва CIL (Common Intermediate Language) або MSIL (Microsoft Intermediate Language), що інтерпретується віртуальною машиною CLR (Common Language Runtime). Має дещо нижчу швидкодію, ніж мова C++, проте вищу, ніж Java. Характеризується наявністю великої кількості сторонніх бібліотек, автоматичного керування пам’яттю та вбудованою реалізацією деяких шаблонів проектування – наприклад, шаблон Спостерігач (Observer) реалізується мовною конструкцією event [11].

Java – мова програмування високого рівня, випущена компанією «Sun Microsystems» у 1995 році. Найактуальніша версія мови Java Standard Edition 10 вийшла 20 березня 2018 року. Має C-подібний синтаксис. Підтримує парадигму об’єктно-орієнтованого програмування. Є кросплатформною, оскільки код програми компілюється в байт-код, що при виконанні інтерпретується віртуальною машиною для конкретної платформи. Має дещо гіршу швидкодію у порівнянні з мовами C++ та C#. Характеризується

наявність великої кількості сторонніх бібліотек. Java використовує автоматичний збирач сміття для керування пам'яттю під час життєвого циклу об'єкта. Програміст вирішує, коли створювати об'єкти, а віртуальна машина відповідальна за звільнення пам'яті після того, як об'єкт стає непотрібним. Коли до певного об'єкта вже не залишається посилань, збирач сміття може автоматично прибирати його із пам'яті [12].

Результати порівняльного аналізу мов програмування наведено у таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз мов програмування

Критерій	C++	C#	Java
Підтримка об’єктно-орієнтованого програмування	+ (1.0)	+ (1.0)	+ (1.0)
Автоматичне керування пам’яттю	- (0.0)	+ (1.0)	+ (1.0)
Кросплатформність	- (0.0)	+ (1.0)	+ (1.0)
Висока швидкодія	+ (1.0)	+ (1.0)	+/- (0.5)
Наявність та доступність сторонніх бібліотек	+ (1.0)	+ (1.0)	+ (1.0)
Сума балів	3	5	4.5

У результаті проведеного аналізу для розробки було обрано мову програмування C#.

Веб-розробка мовою програмування C# здійснюється за допомогою фреймворка ASP.NET. Найактуальнішою версією фреймворку ASP.NET є ASP.NET Core 2.0, випущена у жовтні 2017 року. Основною перевагою перед ASP.NET MVC є кросплатформність, що надає гнучкості при виборі хостингу. Саме тому для розробки веб-додатку була використана ASP.NET Core 2.0.

Для розробки клієнтської частини додатку було обрано фреймворки Angular та Bootstrap. Перевагою Bootstrap є використання готових стилів та функцій що дозволяє прискорити процес розробки.

Хостинг веб-сервісів, розроблених за допомогою фреймворку ASP.NET Core 2.0, може відбуватися на комп'ютері під керуванням як операційної системи Windows, так і операційної системи Linux. Для хостингу веб-сервісу під керуванням операційної системи Linux використовується веб-сервер Apache. Для хостингу веб-сервісу під керуванням операційної системи Windows використовується веб-сервер Internet Information Services (IIS). Для хостингу розроблюваного веб-сервісу було обрано саме операційну систему Windows, оскільки подібний підхід спрощує процес відлагодження розроблюваного проекту на етапі розробки.

У якості системи керування основною базою даних було обрано Microsoft SQL Server 2012, оскільки Microsoft SQL Server вирізняється наявністю великої кількості допоміжних інструментів, що спрощують процес розробки та підтримки системи. Вибір версії 2012 року обумовлений вищою надійністю «старіших» версій продукту.

3.2 Вибір середовища розробки

Для ефективної розробки програмного забезпечення застосовуються інтегровані середовища розробки (IDE – Integrated Development Environment). Інтегроване середовище розробки – це комплексне програмне рішення для розробки програмного забезпечення, що складається з редактора вихідного

коду, інструментів для автоматизації компіляції та відлагодження програм. Більшість сучасних середовищ розробки мають вбудовану функцію автодоповнення коду.

З метою вибору середовища розробки для порівняння було обрано середовища Visual Studio 2017, Project Rider [13].

Результати порівняльного аналізу середовищ розробки зведено у таблиці 3.2.

Таблиця 3.2 – Порівняльний аналіз середовищ розробки

Критерій	Visual Studio 2017	Project Rider
Вбудована підтримка модульного тестування	+ (0,5)	+ (0,5)
Вбудована підтримка контролю версій	+ (0,6)	+ (0,6)
Вбудована підтримка nuget	+ (0,5)	+ (0,5)
Автодоповнення коду	+ (0,9)	+ (0,9)
Можливість додання розширень	+ (0,5)	+ (0,5)
Наявність безкоштовної версії	+ (0,9)	- (0)
Сума	3,9	3,0

Microsoft Visual Studio – серія продуктів корпорації Microsoft, що підтримують розробку для .NET, а також розробку на C++, Python, Node.js, JavaScript/TypeScript. Microsoft Visual Studio надає засоби для розробки консольних додатків, програм з графічним інтерфейсом, зокрема з підтримкою технологій Windows Forms та WPF, а також веб-сайти, веб-додатки тощо.

Project Rider – продукт компанії JetBrains для розробки для .NET. Як і інші середовища розробки від компанії JetBrains, характеризується потужним механізмом автодоповнення та статичного аналізу коду.

У результаті аналізу було прийнято рішення про використання інтегрованого середовища розробки Microsoft Visual Studio 2017, оскільки воно вирізняється наявністю безкоштовної версії – Microsoft Visual Studio 2017 Community, а також має зручний інтерфейс для роботи з nuget.

Для роботи з Microsoft SQL Server 2012 використовуватиметься SQL Server Management Studio 11, оскільки ця утиліта є офіційним засобом для конфігурування, керування та адміністрування компонентів Microsoft SQL Server.

Для роботи з фреймворком Angular викоирстовувався Visual Studio Code.

3.3 Розробка програмних модулів системи

Для реалізації використаємо змішаний принцип розробки, переважним буде Top-Down, коли спочатку будується загальний «скелет» програми, що доповнюється функціоналом у порядку зменшення важливості [13]. Після впровадження нової функціональної особливості виконується тестування системи та виправляються недоліки. Розробка відбуватиметься у наступному порядку:

1. Базові функції веб-додатку.
2. Мікросервіс оновлення службової бази даних.
3. Решта функцій веб-додатку.

Такий порядок розробки дає можливість виконувати тестування та зневадження одразу після розробки певної функції.

Ключовим елементом веб-додатку є рівень доступу до даних (Data Access Layer), що реалізується у класі ApplicationDbContext з використанням фреймворку Entity Framework Core.

При розробці основної бази даних було обрано підхід Code First. При застосуванні цього підходу модель даних задається через програмний код, у класі StoreDbContext з його методами [15]. На рис. 3.1 наведено фрагмент реалізації класу StoreDbContext у якому задається опис таблиці OrderProduct.

```
public class StoreDbContext : DbContext
{
    public StoreDbContext(DbContextOptions options) :
base(options) { }

    public DbSet<Order> Orders { get; set; }
    public DbSet<Product> Products { get; set; }
    public DbSet<ProductCategory> ProductCategories { get;
et; }

    protected override void OnModelCreating(ModelBuilder
odelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<OrderProduct>()
            .HasOne(x => x.Order)
            .WithMany(x => x.OrderProducts)
            .HasForeignKey(pt => pt.OrderId);

        modelBuilder.Entity<OrderProduct>()
            .HasOne(x => x.Product)
            .WithMany(x => x.OrderProducts)
            .HasForeignKey(x => x.ProductId);
    }
```

Рисунок 3.1 – Опис таблиці OrderProduct

На рис. 3.2 показано реалізацію методу UpdateEntity, який дає змогу додавати, оновлювати позиції у меню.

```
public Product UpdateEntity(Product entity,
StoreServiceBase<ProductCategory> categoryService)
    {
        if (entity is null)
        {
            throw new ArgumentNullException(nameof(entity));
        }

        if (Category != null)
        {
            entity.Category = categoryService.Get(Category.Id);
        }

        entity.Description = Description;
        entity.Name = Name;
        entity.Price = Price;
        entity.ImageUrl = ImageUrl;

        return entity;
    }
```

Рисунок 3.2 – Опис методу UpdateEntity

На рис 3.3 представлено реалізацію методу UpdateEntity для ProductCategory, що дає змогу виконати додавання або оновлення категорій страв у меню.

```
public ProductCategory UpdateEntity(ProductCategory
entity)
    {
        if (entity is null)
        {
            throw new ArgumentNullException(nameof(entity));
        }

        entity.Description = Description;
        entity.Name = Name;

        return entity;
    }
```

Рисунок 3.3 – UpdateEntity для ProductCategory

Для прикладу на рис. 3.4 представлено заповнення таблиці Product.

```
public class Product
{
    [Key, Required]
    public int Id { get; set; }

    [Required, StringLength(150)]
    public string Name { get; set; }
    [Required]
    public decimal Price { get; set; }

    [Required, StringLength(3000)]
    public string Description { get; set; }
    [Required]
    public string ImageUrl { get; set; }

    [ForeignKey("Category")]
    public int CategoryId { get; set; }
    public virtual ProductCategory Category { get; set; }

    public virtual ICollection<OrderProduct> OrderProducts {
get; set; }
  }
```

Рисунок 3.4 – Заповнення таблиці Product

На рисунку 3.5 зображений програмний код для опису сутності замовлення, де ми вказузуємо унікальний ідентифікатор та усі інші атрибути, які будуть в базі даних.

```
public class OrderProduct
{
    [Key]
    public int Id { get; set; }

    public int OrderId { get; set; }
    public virtual Order Order { get; set; }

    public int ProductId { get; set; }
    public virtual Product Product { get; set; }
}
```

Рисунок 3.5 – Клас замовлення продукту

Аналогічно до опису замовлення ми описуємо категорію. Програмний код наведено на рисунку 3.6.

```
public class ProductCategory
{
    [Key, Required]
    public int Id { get; set; }

    [Required, StringLength(150)]
    public string Name { get; set; }
    [StringLength(3000)]
    public string Description { get; set; }
}
```

Рисунок 3.6 – Клас категорії продукту

Точка входу для програми відображена на рисунку 3.7 Сворює веб-хост, та задає потрібні налаштування сервісу.

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[]
args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseUrls("http://*:5000")
            .UseStartup<Startup>();
}
```

Рисунок 3.7 – Точка входу для програми

Програмний код моделі замовлення зображено на рисунках 3.8. та 3.9. Вона має свої власні поля, та можливість оновлювати саму себе.

```
public class OrderModel
{
   public int Id { get; set; }

   public string CustomerName { get; set; }
   public string CustomerPhoneNumber { get; set; }
   public string CustomerEmail { get; set; }
   public IEnumerable<ProductModel> Products { get; set; }

   public string Address { get; set; }
   public DateTime DeliveryDate { get; set; }
   public int Status { get; set; }
   public decimal TotalPrice { get; set; }

   public string Comment { get; set; }
```

Рисунок 3.8 – Опис моделі замовлення

```
    public Order UpdateEntity(Order entity,
StoreServiceBase<Product> productService)
    {
      if (entity is null)
      {
        throw new ArgumentNullException(nameof(entity));
      }

      entity.Address = Address;
      entity.Comment = Comment;
      entity.CustomerEmail = CustomerEmail;
      entity.CustomerName = CustomerName;
      entity.CustomerPhoneNumber = CustomerPhoneNumber;
      entity.DeliveryDate = DeliveryDate;
      entity.Status = Status;

      // temporary.
      // this solution will not work in all cases.
      if (entity.OrderProducts is null)
      {
        entity.OrderProducts = Products.Select(x => new
OrderProduct() { Product = productService.Get(x.Id) })
          .ToList();
      }

      return entity;
    }
```

Рисунок 3.9 – Опис моделі замовлення (продовження)

Сервіс замовлення звертається до бази даних, та повертає значення за ідентифікатором. Його програмний опис наведено на рисунку 3.10.

```
public class OrdersService : StoreServiceBase<Order>
{
    public OrdersService(StoreDbContext context) : base(context) {
}

    public override Order Get(int id)
    {
        return this.context.Set<Order>()
            .Include("OrderProducts")
            .Include("OrderProducts.Product")
            .FirstOrDefault(x => x.Id == id);
    }
}
```

Рисунок 3.10 – Опис сервісу замовлення

3.4 Висновки

У третьому розділі було розглянуто найбільш популярні технології, що застосовуються при створенні Web-додатків, а саме: засоби розширення функціональності браузерів, такі як скриптові мови, елементи управління ActiveX, Java-аплети і додатки Macromedia Flash, а також технології створення серверних Web-додатків, такі як CGI, ISAPI , ASP, JSP, PHP, ASP .NET.

Описано технології та засоби що використовувались безпосередньо при розробці даного Web-додатку: ASP.NET Core, OData, Angular, TypeScript, СУБД MSSQL, EntityFramework, Bootstrap. Представлено реалізацію важливих методів Web-додатку.

4. ТЕСТУВАННЯ WEB-СИСТЕМИ ЗАКЛАДУ ХАРЧУВАННЯ

4.1Аналіз методів тестування

Тестування веб-сайту – це процес, який полягає в перевірці відповідності сайту заявленим характеристикам, вимогам експлуатації в різних середовищах, з різними навантаженнями, вимогам по зручності використання .

Залежно від спрямованості тестування, перевіряється та чи інша особливість веб-сайту: корпоративного сайту, інтернет-магазину, сайту-візитки. Як правило, процес тестування документується у вигляді тестового плану і тест-кейсів. Тестовий план описує стратегію тестування, методи і засоби тестування, порядок якого і інші його особливості. Тест-кейси описують послідовні покрокові операції перевірки функціоналу програми або веб-сайту.

Як правило, тестуванням сайту займається декілька людей, які проводять такі види тестування:

1. Функціональне тестування. Передбачає перевірку роботи головних функцій сайту, коректність посилань, тестування роботи користувацьких форм, системи авторизації, тощо [16].

2. Тестування інтерфейсу. За цього виду тестування перевіряється відповідність елементів на сайті з їх розміщенням в макеті. Також перевіряється кросбраузерність і адаптивність сайту.

3. Тестування зручності. Під тестуванням зручності розуміють процес визначення того, наскільки веб-сайт є зручним для користувача. Основним завданням є перевірка того, чи може користувач швидко знайти те, що він шукав, і що йому при цьому нічого не заважає.

4. Тестування безпеки. Цей вид тестування полягає у перевірці захищеності сайту від різноманітних видів атак, а також надійність його роботи.

5. Тестування продуктивності. Проводиться з ціллю перевірки того, як швидко завантажуються сторінки сайту і які навантаження він здатен витримувати.

За способом проведення, тестування поділяють на ручне, коли воно проводиться безпосередньо самим розробником, та автоматизоване, коли воно проводиться без участі програміста.

Важливими вимогами до сучасних сайтів є їх кросбраузерність та адаптивність. Під кросбраузерністю сайту розуміють коректне відображення елементів сторінки та їх робота в усіх браузерах.

4.2 Тестування Web-системи

З метою перевірки відповідності між реальною поведінкою програми і очікуваннями здійснюють тестування програмних продуктів. Найбільш поширеними методами тестування вважають тестування «білого» та «чорного ящика», їх розрізняють за доступом розробника тестів до вихідного коду. Одним із способів вивчення поставленого питання є дослідження стратегії «чорної скриньки», тестування з керуванням за даними, або тестування з керуванням по входу-виходу. При використанні цієї стратегії програма розглядається як чорна скринька. Таке тестування має на меті з'ясування обставин, в яких поведінка програми не відповідає її специфікації. Тестові ж дані використовуються тільки у відповідності зі специфікацією програми (без урахування знань про її внутрішню структуру)[17].

Стратегія «білої скриньки», або стратегія тестування, керованого логікою програми, дозволяє досліджувати внутрішню структуру програми. В цьому випадку тестувальник отримує тестові дані шляхом аналізу логіки програми При використанні даної стратегії часто не використовується специфікація програми.

Для тестування програмного додатку було обраного стратегію «чорної скриньки», тестування відбувалось на основі браузера Google Chrome.

На рис. 4.1 і рис. 4.2 відображено тест адаптивності дизайну, з якого можна побачити, що дизайн додатку повністю адаптивний під розміри вікна браузера.

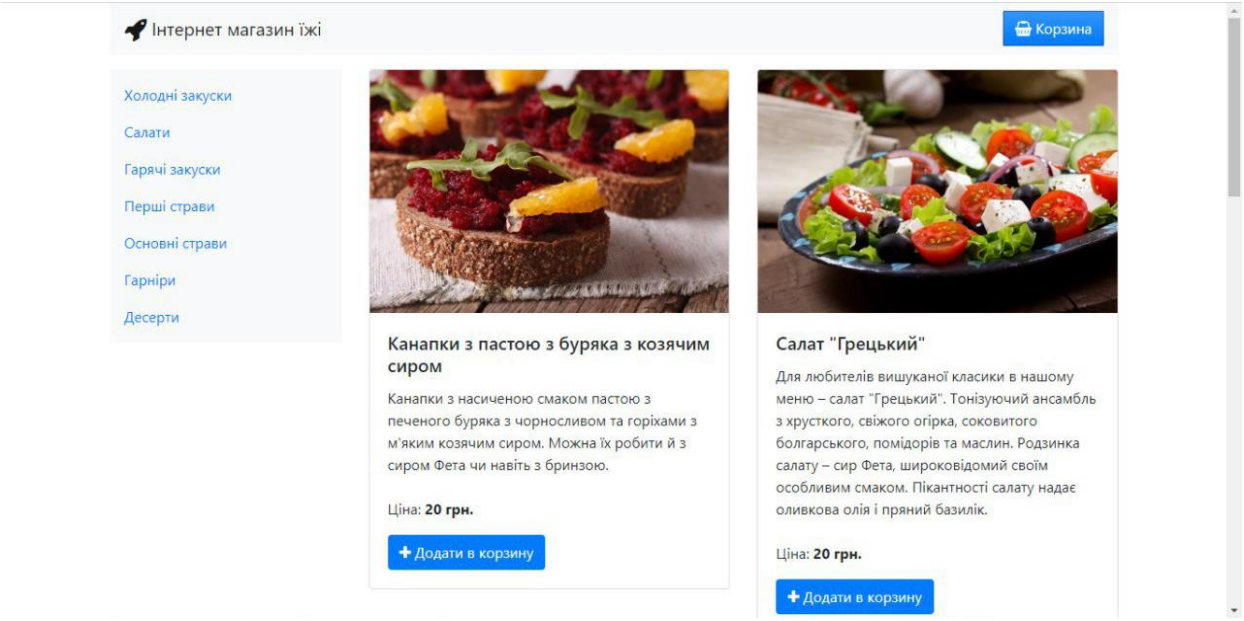

Рисунок 4.1 – Тестування адаптивності дизайну

Наявність адаптивної версії дає можливість легко здійснювати покупки онлайн власникам мобільних телефонів. Якість юзабіліті сайту безпосередньо пов’язана з показниками конверсії: чим зручніший та зрозуміліший ресурс, тим простіше відвідувачу здійснити важливі для бізнесу дії. Сьогодні можна шукати товари та послуги прямо у смартфоні. Адаптивна версія сайту – один зі способів продемонструвати відвідувачам готовність надавати сервіс високої якості та увагу до своєї аудиторії. Саме тому впродовж багатьох років Google встановлює певні стандарти для веб-ресурсів, які хочуть потрапити до Топу пошукової видачі[20]. Сайт ресторану є пристосованим до всіх видів девайсів у різних браузерах (Chrome, Firefox, Edge, Opera) та операційних систем, таких як Windows та MacOs [18].

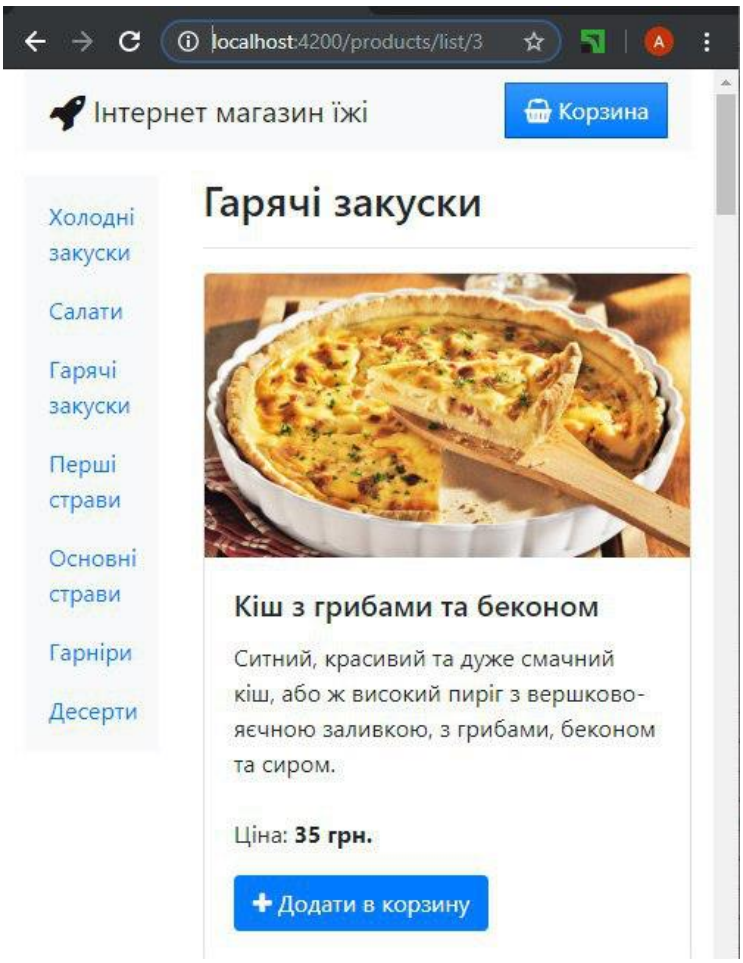

Рисунок 4.2 –Тестування адаптивності дизайну

Головна сторінка має наступний вигляд як на рис. 4.3.

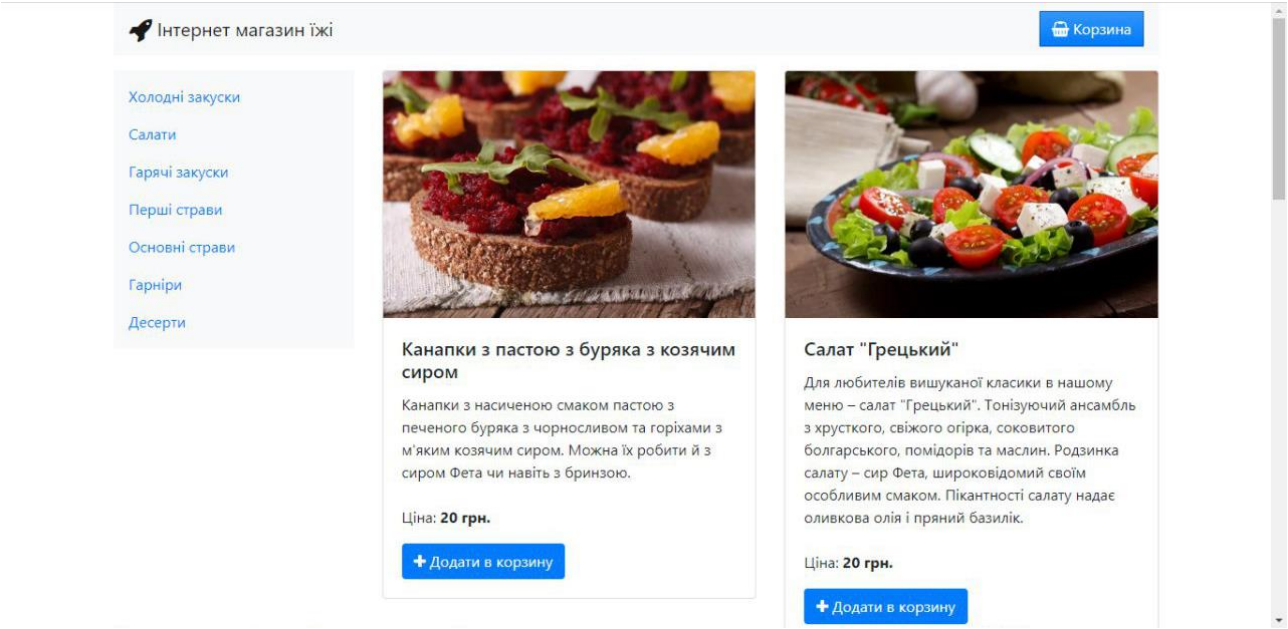

Рисунок 4.3 – Головна сторінка Web-додатку

У верхньому лівому кутку бачимо надпис «Інтернет магазин їжі», та емблему космічної ракети, що надає асоціацію користувачу про дуже швидку доставку. Одразу знизу розташоване меню, з усіма доступними розділами їжі для замовлення. Основну площу сторінки займає безпосередньо відображення страв. Кожна страва займає окремий блок, який складається з: зображення страви, назвою, коротким описом з інгрідієнтами, ціною вказаною у гривнях, та кнопкою «Додати в корзину», яка складає основу для функції замовлення. Перейти безпосередньо до корзини наших замовлень ми можемо перейти нажавши кнопку, яка розташована в правому верхньому кутку сторінки. Відображення страв на головній сторінці відбувається за принципом найпопулярнішого. Тобто на сторінці будуть відображатися страви, кількість замовлень яких – була найбільшою, ігноруючи розділи їжі.

Нажимаючи на один з розділів (рис. 4.4) їжі ми переходимо на сторінку з саме цим типом страв. Відображення тут також відбувається за критерієм популярності замовлень, але відображаються страви лише з цього розділу [19].

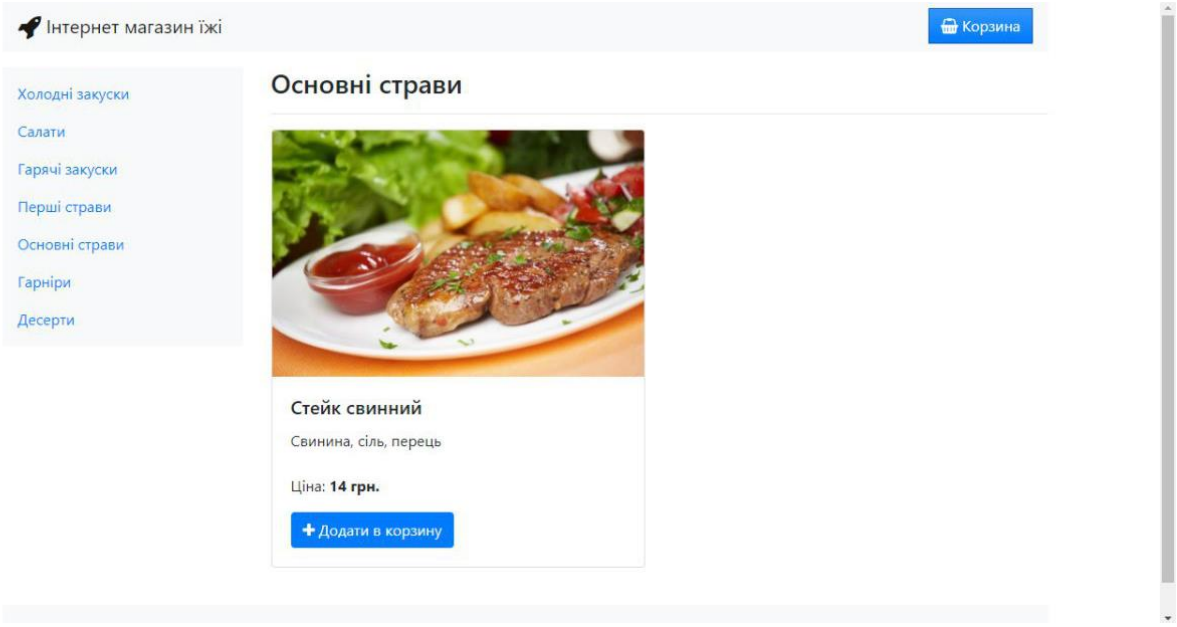

Рисунок 4.4 – Сторінка для розділу страв

Після того, як користувач додав усі бажані страви у корзину та перейшов у неї, відбувається відображення наступних сторінок (рис. 4.5, рис. 4.6).

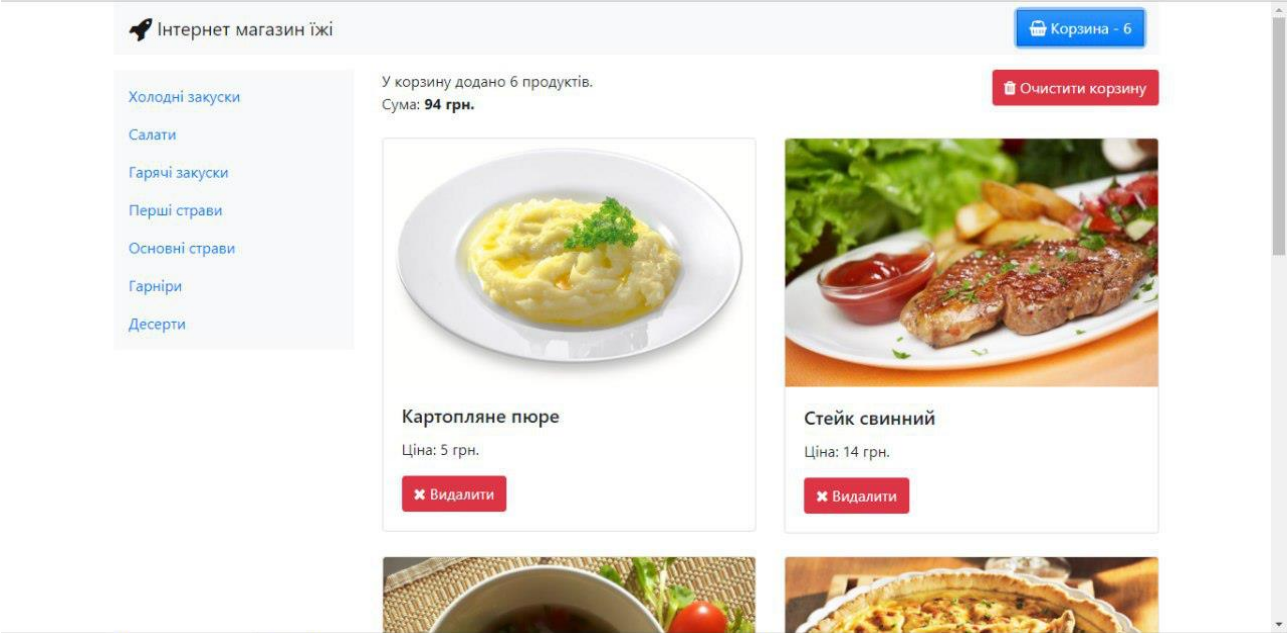

Рисунок 4.5 – Верхня частина сторінки корзини

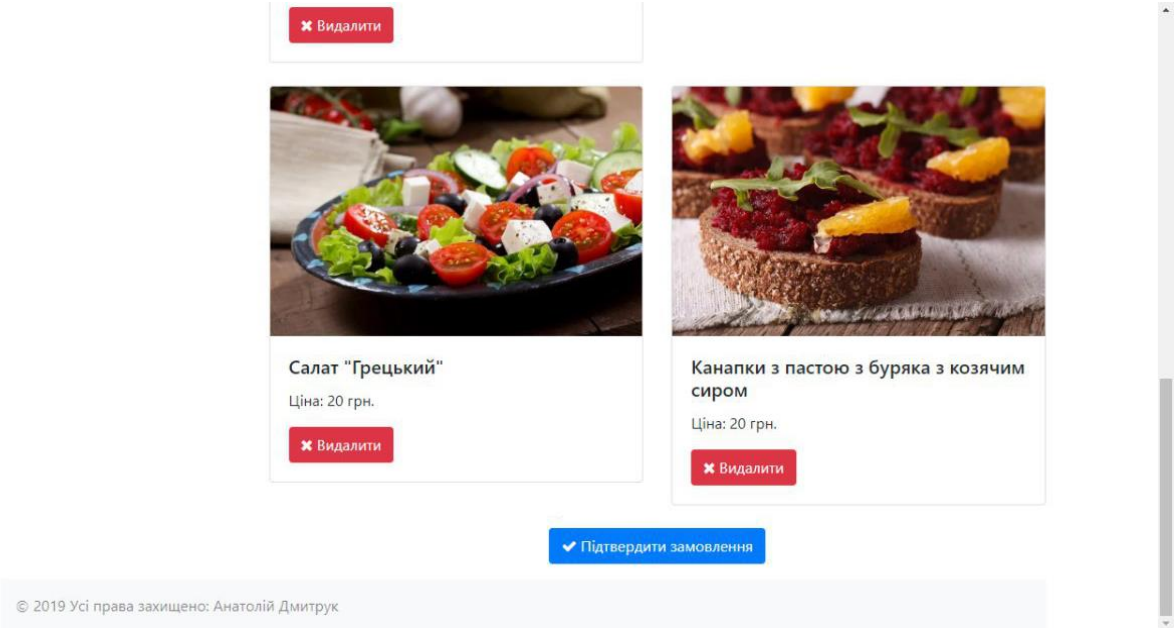

Рисунок 4.6 – Нижня частина сторінки корзини

На сторінці корзини ми можемо побачити усі додані користувачем раніше страви зі зображенням зовнішнього вигляду, назвою, ціною та

кнопкою «Видалити», яка реалізує функцію видалення страви з корзини. Також є можливість видалити усі страви за допомогою кнопки «Очистити корзину», яка розташована у правому верхньому кутку. Відображається і загальна кількість страв, та сумарна ціна за них – зверху зліва. Якщо користувача все влаштовує, він може перейти на безпосереднє замовлення, де потрібно вказати контактні дані, нажавши на кнопку «Підтвердити замовлення», яка розташована знизу сторінки .

Далі користувач повинен заповнити свої дані (рис. 4.7), для того, щоб оператор зміг зв’язатися з ним у наступних полях: «Ваше ім’я», «Номер телефону», «Поштова адреса», «Адреса доставки», «Дата доставки», та поле «Коментар» [20], яке є опціональним, і заповнюється лише за бажанням користувача.

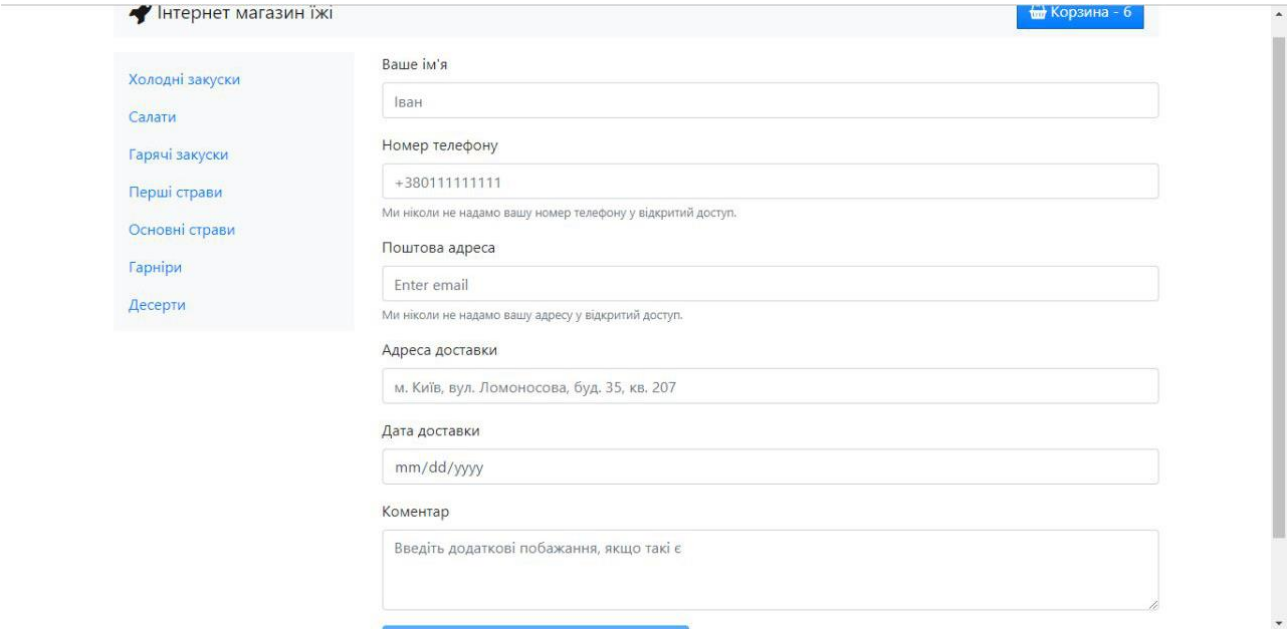

Рисунок 4.7 – Сторінка оформлення замовлення

У разі коректної заповнення форми, ми перейдемо на сторінку індикації успішного замовлення (рис. 4.8), де користувача буде проінформовано про те що найближчим часом зателефонує оператор, для остаточного підтвердження. Сторінка оформлення замовленн викона в зрозумілому для користувача форматі для зручного замовлення.

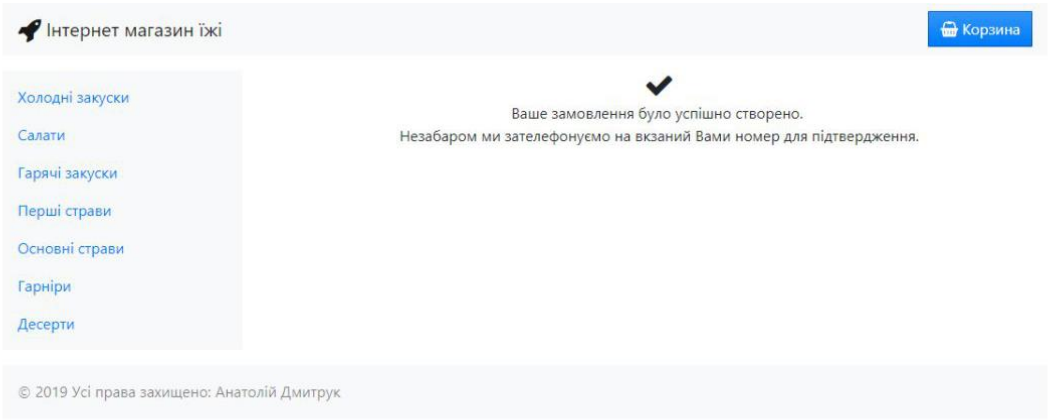

Рисунок 4.8 – Сторінка індикації замовлення

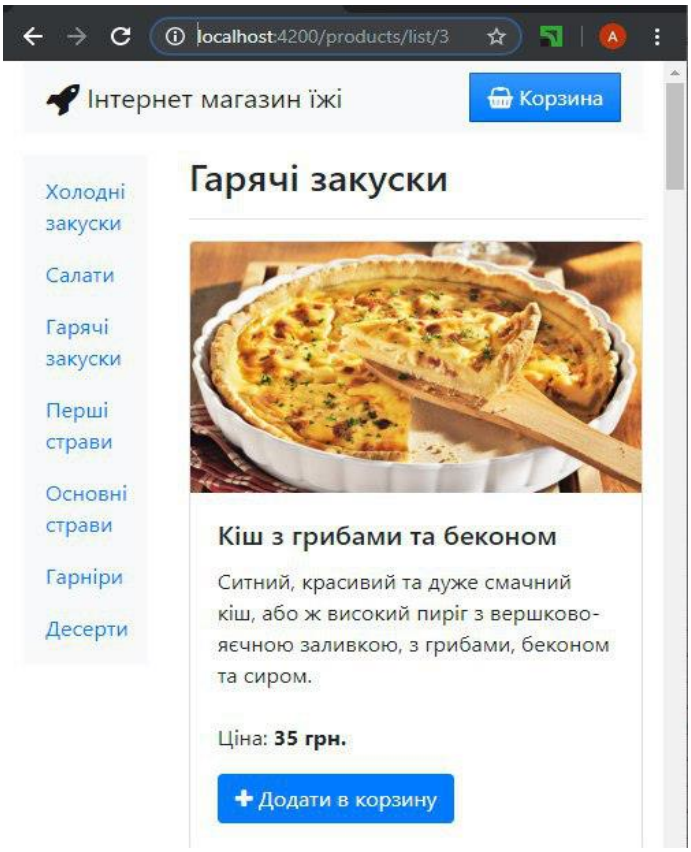

Рисунок 4.9 – Тестування адаптивності дизайну

Спробуємо додати страви у корзину, для подальшого оформлення замовлення. Як видно на рис. 4.10, після додавання страв до корзини відображається їх кількість, після вилучення зберігається коректна кількість страв у корзині (рис.4.11).

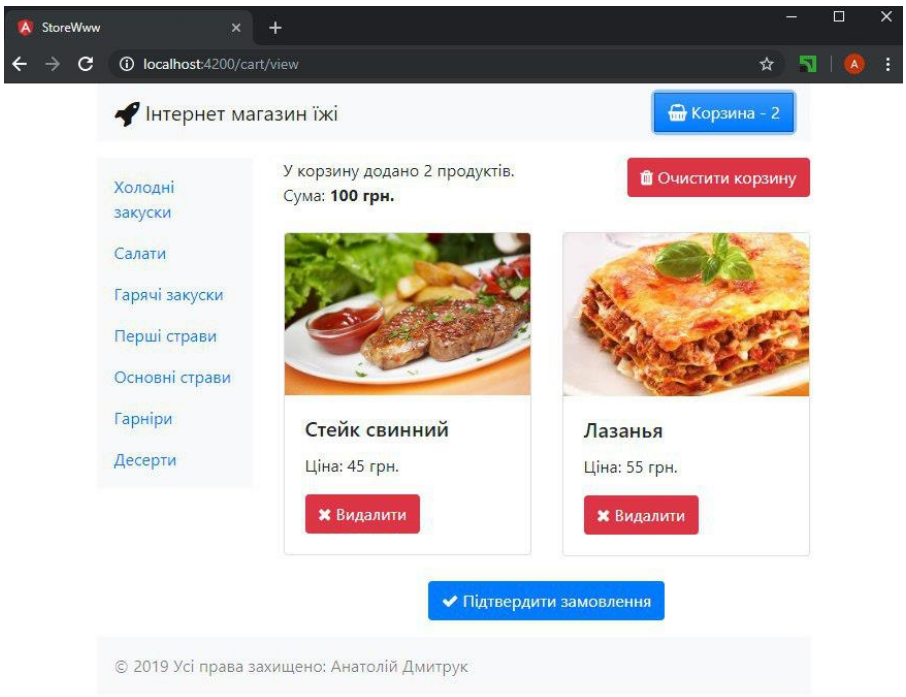

Рисунок 4.10 – Корзина з двома стравами

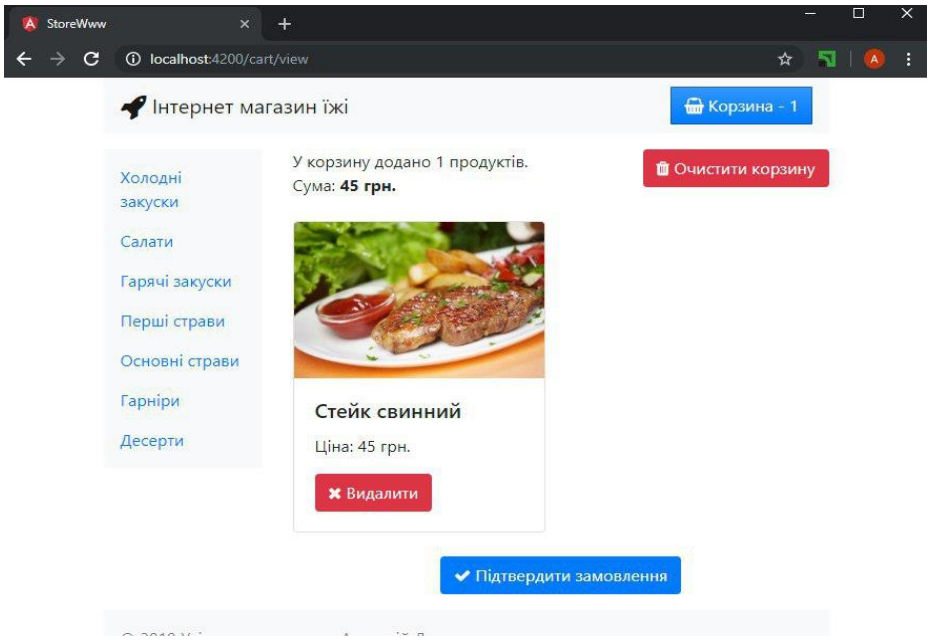

Рисунок 4.11 – Корзина після видалення страви

Як видно з наведених вище рисунків, вміст додатку коректно відображається у браузері, функціонує адаптивний дизайн.

4.3. Висновки

У четвертому розділі роботи було розглянуто види та методики тестування, обрано методику тестування «чорної скриньки, проведено опис роботи різних компонентів, а також тестування основних параметрів, таких як коректність відображення інформації та адаптивність дизайну.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Войтко Вікторія Володимирівна (к.т.н., доц. кафедри ПЗ ВНТУ), Черноволик Галина Олександрівна (к.т.н., доц. кафедри ПЗ ВНТУ).

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Проведено оцінювання комерційного потенціалу за критеріями, наведеними в таблиці 5.1.

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер.	0	1	2	3	4
3	Ціна продукту значно вища за ціни	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам	Ціна продукту дещо нижче за ціни	Ціна продукту значно нижче за ціни

	аналогів		аналогів	аналогів	аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна Конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне значне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні Незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Войтко В. В.	2. Черноволик Г.О.
	Бали, виставлені експертами:	
1	3	4
2	3	2
3	2	3
4	4	4
5	2	3
6	3	4
7	4	3
8	4	2

9	3	3
10	2	3
11	4	4
12	2	2
Сума балів	$СБ_1 = 36$	$СБ_2 = 37$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 36,5$	

Отже, з отриманих даних таблиці 5.2 видно, що нова розробка має достатній рівень комерційного потенціалу.

. Web-сайт на тему «Доставка їжі онлайн». Основне використання даного продукту проводиться в ресторанах швидкого приготування, також звичайних харчових закладах. Серед аналогів існує «Сушия», «Квадрат», «Dominos».

Переваги даного продукту:

- приємний UI;

- панель керування сайту зручна для адміністрацій харчових закладів;

- оплата онлайн чи готівкою;

- розумні фільтри;

Про даний продукт дізнаються звичайні люди, а також харчові заклади, так як є можливість використовувати даний продукт як «дошку оголошень» для їжі.

З технічних переваг є: відома майже всім CMS WordPress, яка є легка в вивченні та використанні; база данних Mysql, яка також всім відома і проста.

Також використана мова програмування PHP, що є вельми простою у вивченні і користуванні.

Даний продукт є дуже цікавий в нинішній час, так як є купа різних компаній по доставці їжі, які тримають свої лідерські місця, що значить, що даний тип продуктів програмного забезпечення є цікавий людям і широко використовується. І затребуваний як звичайними людьми, так і харчовими закладами. Він дає можливість закаладам харчування викладати свій товар у нас, а потребувачам купляти продукти. Де дозволяє зробити процес замовлення набагато легшим і доступним для всіх, а закладач швидкого харчування розширитись і отримати більший прибуток за рахунок розміщення свого товару на даній площадці.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.3.

Таблиця 5.3 – Розрахунки основної заробітної плати

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	30000	1363,63	5	6818,15
Frontend-програміст	25000	1136,36	20	22727,2
Всього:				29545,35

Розрахуємо додаткову заробітну плату:

$$Здод = 0{,}1 \cdot 29545{,}35 = 2954{,}535\ (грн.)$$

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми їхньої основної та додаткової заробітної плати (вираз 5.2):

$$Нзп = (Зо + Зр) \cdot \frac{\beta}{100}, \qquad (5.2)$$

$$Нзп = (29545{,}35 + 2954{,}535) \cdot \frac{22}{100} = 7149{,}97\ (грн.).$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою (вираз 5.3):

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \qquad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.4 – Розрахунок амортизаційних відрахувань

Найменування	Ціна, грн.	Норма амортизації, %	Термін використання, м.	Сума амортизації
ПК + прилади керування	10000	20	3	500
Прилади маніпуляції ПК	1000	20	3	50
Всього	550			

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n \text{Н}_\text{і} \cdot \text{Ц}_\text{і} \cdot \text{К}_\text{і},$$

(5.4)

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.5 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	200	1	200
Фліпчарт	шт.	350	1	350
Маркери	шт.	25	4	100
Всього з урахуванням транспортних витрат				750

Витрати на силову електроенергію розраховуються за формулою (вираз 5.5):

$$В_е = В \cdot П \cdot Ф \cdot К_п ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=9,5 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=250 год.);

$К_п$ – коефіцієнт використання потужності ($К_п$< 1, $К_п = 0,8$).

$$Ве = 9,5 \cdot 0,6 \cdot 250 \cdot 0,8 = 1140 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто(вираз 5.6):

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (29545,35 + 2954,535) = 32\,499,885 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

$$В = 29545,35+2954,535+7149,97+550+750+1140+32\,499,885 = 74\,589,74 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою(вираз 5.7):

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{74\ 589{,}74}{1} = 74\ 589{,}74$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою(вираз 5.8):

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{74\ 589{,}74}{0{,}9} = 82\ 877{,}48(грн.)$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою(вираз 5.9):

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення програмного продукту зменшаться на 60 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 60 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 450 користувачів, протягом другого року – на 400 користувачів, протягом третього року – 350 користувачів. Реалізація програмного продукту до впровадження результатів наукової розробки складала 50 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 100 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_{2020} = 60 \cdot 50 + (100 + 60) \cdot 450 = 75000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_{2021} = 60 \cdot 50 + (100 + 60) \cdot (450 + 400) = 139000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_{2022} = 60 \cdot 50 + (100 + 60) \cdot (450 + 400 + 350) = 195000 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою(вираз 5.10):

$$Е_{абс} = (ПП - PV), \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

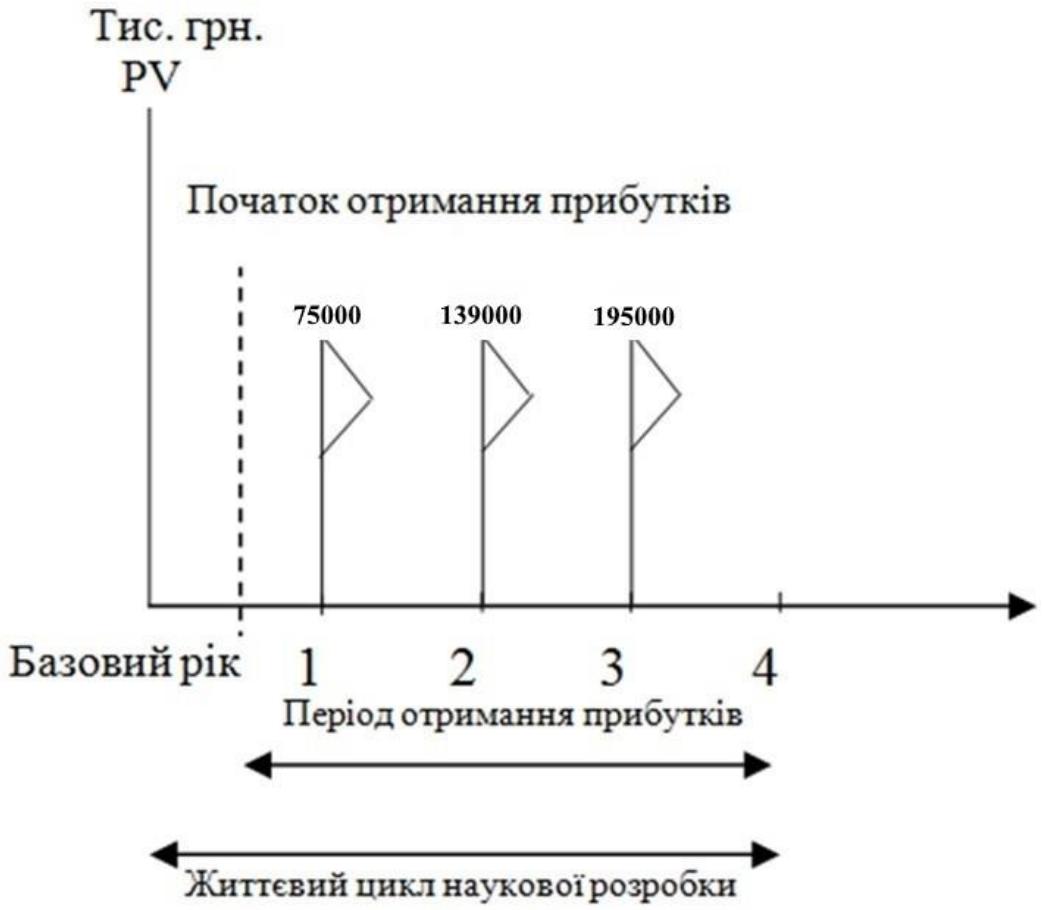

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою(вираз 5.11):

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

Т – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{82\,877,48}{(1+0,1)^0} + \frac{75000}{(1+0,1)^2} + \frac{139000}{(1+0,1)^3} + \frac{195000}{(1+0,1)^4} = 382481,32(грн.)$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 382481,32 - 82\,877,48 = 299\,603,84грн.$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою(вираз 5.12):

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{299\,603,84}{82\,877,48}} - 1 = 0,58 \text{ або } 58\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 58% > 30%, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проєкту інвестицій. Термін окупності вкладених у реалізацію наукового проєкту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{0{,}44} = 2{,}27 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

Під час оцінки комерційного потенціалу розробки можна зробити висновок, що розробка має досить високий рівень комерційного потенціалу. В ході оцінки було розраховано витрати на виконання науково-дослідної та конструкторсько-технологічної роботи, а також показники збільшення чистого прибутку протягом трьох років.

Таким чином, обрахунок терміну окупності розробки показав, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У магістерські кваліфікаційній роботі розроблено Web-системи для замовлення їжі онлайн. Проведено аналіз сучасних web-технологій та обґрунтовано здійснений вибір засобів реалізації сайту. Здійснено детальний аналіз предметної області та визначено актуальність питання, проаналізовані основні конкуренти сайту.

Також у роботі проаналізовано принципи реалізації Web-системи та розроблено функціональну структуру додатку. У роботі було розглянуто основні принципи розробки Web-систем. Проведений варіантний аналіз мов програмування та СКБД, що використовуються при розробці Web-систем.

Проведено повноцінне тестування системи.

Web-система працює коректно, має адаптивний дизайн та інтерфейс. Має високу швидкість завантаження сторінки.

Розвязано такі задачі:

1. Розглянуто основні можливості застосування Web-систем у даній галузі.

2. Проаналізувано існуючі засоби для розробки Web-систем.

3. Розроблено метод, модель, структуру та дизайн Web-системи.

4. Розроблено і систематизувано контент.

5. Розроблено та протестовано Web-система для замовлення їжі онлайн.

Було розглянуто види тестування, проведено тестування веб-системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фримен А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание / А. Фримен – Москва: «Вильямс», 2013 – 688 с.

2. Войтко В. В. До розробки програмного забезпечення Веб-системи закладу харчування / В. В. Войтко, Н.Є. Барчук, А.О. Дмитрук // матеріали LXVII Міжнародна інтернет-к конференції «Перспективні напрямки наукових досліджень». м. Рівне, 31 травня 2021 р. [Електронний ресурс]. – Режим доступу: https://el-conf.com.ua/%d0%b0%d1%80%d1%85%d1%96%d0%b2-%d0%ba%d0%be%d0%bd%d1%84%d0%b5%d1%80%d0%b5%d0%bd%d1%86%d1%96%d0%b9/

3. Інформаційні системи і технології в статистиці: Навч. посібник / За ред. д-ра екон. наук, проф. В. Ф. Ситника. — К.: КНЕУ, 2003. — 267 с.

4. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP: разработка динамических веб-приложений. – СПб.: Символ- Плюс, 2007. – 336 с., ил. 3.

5. Troelsen A. C# 6.0 and the .NET 4.6 Framework, 7th edition / A. Troelsen, Ph. Japikse – Apress, 2015 – 1625 ст.

6. Кузнецов С.Д. Основы баз данных, 2-е издание / С.Д. Кузнецов – Москва: «БИНОМ», 2007 – 484 ст.

7.Майданюк В.П. Інтерфейс"Користувач-комп'ютер": Навчальний посібник / В.П. Майданюк, А.М. Пєтух. - Вінниця: ВДТУ, 1999. - 66 с.

8. Романюк О.Н. Веб-дизайн і комп'ютерна графіка: навч. посіб. для студ. напр. підгот. «Прогр. інженерія» всіх спец. / О. Н. Романюк, Д. І. Катєльніков, О.П. Косовець; Вінницьк. нац. техн. ун-т. - Вінниця: ВНТУ, 2007. - 141 с.

9. Алгоритми. Побудова і аналіз / Т. Кормен, Ч. Лайзерсон, Р. Ривест, К. Штайн. – Москва: И.Д.Вильямс, 2013. – 1328 с. – (3).

10. Angular [Електронний ресурс]. – Режим доступу: https://angular.io/

11. TypeScript [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/

12. Visual Studio Code [електронний ресурс] - Режим доступу: https://css.in.ua/article/shcho-take-html_10 – назва з екрану

13. ODATA [Електронний ресурс] – Режим доступуhttps://www.odata.org/

14. Visual Studio Code [електронний ресурс] - Режим доступу: https://css.in.ua/article/shcho-take-html_10 – назва з екрану

15. Вильямсон Х. УніверсальнийDynamic HTML / Бібліотека програміста – СПб.:ПИТЕР, 2001

16. Продизайн [електронний ресурс] - Режим доступу: https://prodesign.in.ua/2011/12/yak-stvoryty-dyzajn-sajtu-scho-orijentovanyj-na-mobilni-prystroji/ – назва з екрану

17. Тестування [електронний ресурс] // Режим доступу: http://www.victoria.lviv.ua/html/wp/l-testing.html – Назва з екрану.

18. Степанченко И.В. Методы тестирования программного обеспечения : учебное пособие / И.В. Степанченко. – Волгоград : ВолгГТУ, 2006. – 74с..

19. Ideyne [електронний ресурс] - Режим доступу: https://ideyne.com/ua/article/krossbrauzernost_saita – назва з екрану

20. Troelsen A. C# 6.0 and the .NET 4.6 Framework, 7th edition / A. Troelsen, Ph. Japikse – Apress, 2015 – 1625 ст.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О.Н.

“17” лютого 2021 року

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка програмних засобів для інтерактивного обслуговування закладів харчування»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

к.т.н., доцент Войтко В.В.

"18" лютого 2021 р.

Виконав:

студент гр. 1ПІ-19м Дмитрук А.О.

"18" лютого 2021 р.

Вінниця – 2021 року

1. Галузь застосування.

Платформа застосовується у процесі використання веб-системи для замовлення їжі.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 65 ректора ВНТУ про закріплення тем МКР від 9 березня 2021 року.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності процесу замовлення їжі онлайн за рахунок розробки і використання спеціалізованого Web-ресурсу, що дозволить автоматизувати процеси вибору їжі, оплати, замовлення та доставки.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP: разработка динамических веб-приложений. – СПб.: Символ- Плюс, 2007. – 336 с., ил. 3Норман Д. Дизайн привычных вещей. Москва, 2018. 380 с.

2. Google Material Design. URL: https://material.io/design

3. Кузнецов С.Д. Основы баз данных, 2-е издание / С.Д. Кузнецов – Москва: «БИНОМ», 2007 – 484 с.

4. Інформаційні системи і технології в статистиці: Навч. посібник / За ред. д-ра екон. наук, проф. В. Ф. Ситника. — К.: КНЕУ, 2003. — 267 с.

5. Troelsen A. C# 6.0 and the .NET 4.6 Framework, 7th edition / A. Troelsen, Ph. Japikse – Apress, 2015 – 1625 ст.

4. Технічні вимоги.

Оскільки система складається з 2-х частин (клієнт, сервер), розглянемо вимоги до кожної з них.

1. Клієнт. Оскільки клієнт виконаний в форматі сайту і не має важких обчислень на стороні клієнта, варто звертати увагу лише на вимоги браузера. Розглянемо мінімльні системні характеристики необхідні для роботи з сайтом: Windows 7, Windows 8, Windows 8.1, Windows 10 або пізнішої версії. Процесор Intel Pentium 4 або більш пізніша версія з підтримкою SSE3.
2. Сервер. Мінімальні вимоги: 1 VCPU, 2 GB RAM, 20 GB DISK LOCAL
3. DOCKER. Серверна частина працює на ОС Linux засобами Docker. Якщо маються на увазі апаратні вимоги, то докер сам по собі не має мінімальних апаратних вимог. Це лише утиліта для віртуалізації. Мінімальні системні вимоги залежать від програми, яка буде розвернута в контейнерах. З апаратних вимог лише одна – підтримка віртуалізації. Що стосується системних (програмних) вимог, то для систем на базі ОС Linux потрібно лише: бітність 64, ядро не старіше 3.10.
4. NODE. Якщо маються на увазі апаратні вимоги - все залежить від стороніх бібліотек які використовуються в самому додатку. Розроблюваний додаток не має важких залежностей.

5. Конструктивні вимоги.

Система повинна бути зручною у використанні та обслуговуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до МКР;

- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки.

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання етапів проекту (роботи)
1	Аналіз предметної області	20.02.2021–8.02.21
3	Розробка методу, моделі та структури Web-додатку	01.03.20 – 12.03.21
3	Розробка структури бази даних	13.03.21 – 22.03.21
4	Розробка користувацького інтерфейсу веб-додатку	23.03.21 – 31.03.21
5	Програмна реалізація Web-додатку	01.04.21 – 25.04.21
6	Тестування роботи Web-додатку	01.05.21 – 16.05.21
7	Оформлення матеріалів до захисту МКР	17.05.21–31.05.21

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком .

ДОДАТОК Б Лістинг вихідного коду платформи

Лістинг Order Controllers

```
using System;
using System.Linq;
using System.Linq.Expressions;
using Store.Controllers.Base;
using Store.Entities;
using Store.Models;
using Store.Services.Base;

namespace Store.Controllers
{
   public class OrdersController : StoreControllerBase<Order, OrderModel>
   {
      public OrdersController(StoreServiceBase<Order> service, StoreServiceBase<Product> productsService) : base(service)
      {
         this.productsService = productsService;
      }

      private readonly StoreServiceBase<Product> productsService;

      protected override Expression<Func<Order, OrderModel>> SelectExpression => x => new OrderModel()
      {
         Address = x.Address,
         Comment = x.Comment,
         CustomerEmail = x.CustomerEmail,
         CustomerName = x.CustomerName,
```

```
            CustomerPhoneNumber = x.CustomerPhoneNumber,
            DeliveryDate = x.DeliveryDate,
            Id = x.Id,
            Products = x.OrderProducts == null ? null : x.OrderProducts.Select(y => new ProductModel()
            {
                Price = y.Product.Price,
                Category = new ProductCategoryModel()
                {
                    Id = y.Product.Category.Id,
                    Name = y.Product.Category.Name,
                    Description = y.Product.Category.Description
                },
                Description = y.Product.Description,
                Id = y.Product.Id,
                Name = y.Product.Name,
                ImageUrl = y.Product.ImageUrl
            }),
            Status = x.Status,
            TotalPrice = x.OrderProducts == null ? 0 : x.OrderProducts.Sum(y => y.Product.Price)
        };


        protected override Order ConvertModelToEntity(OrderModel model, Order entity = null)
        {
            return model.UpdateEntity(entity ?? new Order(), this.productsService);
        }
    }
}
```

Лістинг Product Categories Controller

```
using Store.Controllers.Base;
using Store.Entities;
using Store.Models;
using Store.Services.Base;
using System;
using System.Linq.Expressions;

namespace Store.Controllers
{
   public class ProductCategoriesController : StoreControllerBase<ProductCategory, ProductCategoryModel>
   {
      public ProductCategoriesController(StoreServiceBase<ProductCategory> service) : base(service) { }

      protected override Expression<Func<ProductCategory, ProductCategoryModel>> SelectExpression => x => new ProductCategoryModel
      {
         Description = x.Description,
         Id = x.Id,
         Name = x.Name
      };

      protected override ProductCategory ConvertModelToEntity(ProductCategoryModel model, ProductCategory entity = null)
      {
         return model.UpdateEntity(entity ?? new ProductCategory());
      }
```

```
    }
}
```

Лістинг ProductsController

```
using System;
using System.Linq;
using System.Linq.Expressions;
using Store.Controllers.Base;
using Store.Entities;
using Store.Models;
using Store.Services.Base;

namespace Store.Controllers
{
    public class ProductsController : StoreControllerBase<Product, ProductModel>
    {
        public ProductsController(StoreServiceBase<Product> service, StoreServiceBase<ProductCategory> categoriesService) : base(service)
        {
            this.categoriesService = categoriesService;
        }

        private readonly StoreServiceBase<ProductCategory> categoriesService;

        protected override Expression<Func<Product, ProductModel>> SelectExpression => x => new ProductModel()
        {
            Category = x.Category == null ? null : new ProductCategoryModel()
            {
                Description = x.Category.Description,
```

```
            Id = x.Category.Id,
            Name = x.Category.Name
        },
        ImageUrl = x.ImageUrl,
        Description = x.Description,
        Name = x.Name,
        Id = x.Id,
        Popularity = x.OrderProducts == null ? 0 : x.OrderProducts.Count(),
        Price = x.Price
    };

    protected override Product ConvertModelToEntity(ProductModel model, Product entity = null)
    {
        return model.UpdateEntity(entity ?? new Product(), this.categoriesService);
    }
  }
}
```

Лістинг класів:

```
        using System;
        using System.Collections.Generic;
        using System.ComponentModel.DataAnnotations;

        namespace Store.Entities
        {
            public class Order
            {
                [Key, Required]
```

```
        public int Id { get; set; }

        [Required, StringLength(100)]
        public string CustomerName { get; set; }
        [Required, StringLength(20)]
        public string CustomerPhoneNumber { get; set; }
        [Required, StringLength(300)]
        public string CustomerEmail { get; set; }

        [Required, StringLength(1000)]
        public string Address { get; set; }
        [Required]
        public DateTime DeliveryDate { get; set; }
        [Required]
        public int Status { get; set; }

        [StringLength(1500)]
        public string Comment { get; set; }

        public virtual ICollection<OrderProduct> OrderProducts { get; set; }
    }
}

using System.ComponentModel.DataAnnotations;

namespace Store.Entities
{
    public class OrderProduct
    {
        [Key]
```

```
        public int Id { get; set; }

        public int OrderId { get; set; }
        public virtual Order Order { get; set; }

        public int ProductId { get; set; }
        public virtual Product Product { get; set; }
    }
}

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Store.Entities
{
    public class Product
    {
        [Key, Required]
        public int Id { get; set; }

        [Required, StringLength(150)]
        public string Name { get; set; }
        [Required]
        public decimal Price { get; set; }

        [Required, StringLength(3000)]
        public string Description { get; set; }
        [Required]
        public string ImageUrl { get; set; }
```

```
        [ForeignKey("Category")]
        public int CategoryId { get; set; }
        public virtual ProductCategory Category { get; set; }

        public virtual ICollection<OrderProduct> OrderProducts { get; set; }
    }
}

using System.ComponentModel.DataAnnotations;

namespace Store.Entities
{
    public class ProductCategory
    {
        [Key, Required]
        public int Id { get; set; }

        [Required, StringLength(150)]
        public string Name { get; set; }
        [StringLength(3000)]
        public string Description { get; set; }
    }
}

using Store.Entities;
using Store.Services.Base;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
namespace Store.Models
{
    public class OrderModel
    {
        public int Id { get; set; }

        public string CustomerName { get; set; }
        public string CustomerPhoneNumber { get; set; }
        public string CustomerEmail { get; set; }
        public IEnumerable<ProductModel> Products { get; set; }

        public string Address { get; set; }
        public DateTime DeliveryDate { get; set; }
        public int Status { get; set; }
        public decimal TotalPrice { get; set; }

        public string Comment { get; set; }

        public Order UpdateEntity(Order entity, StoreServiceBase<Product>
productService)
        {
            if (entity is null)
            {
                throw new ArgumentNullException(nameof(entity));
            }

            entity.Address = Address;
            entity.Comment = Comment;
            entity.CustomerEmail = CustomerEmail;
```

```
                entity.CustomerName = CustomerName;
                entity.CustomerPhoneNumber = CustomerPhoneNumber;
                entity.DeliveryDate = DeliveryDate;
                entity.Status = Status;

                // temporary.
                // this solution will not work in all cases.
                if (entity.OrderProducts is null)
                {
                    entity.OrderProducts = Products.Select(x => new OrderProduct()
{ Product = productService.Get(x.Id) })
                        .ToList();
                }

                return entity;
            }
        }
    }

    using Store.Entities;
    using System;

    namespace Store.Models
    {
        public class ProductCategoryModel
        {
            public int Id { get; set; }

            public string Name { get; set; }
            public string Description { get; set; }
```

```
        public ProductCategory UpdateEntity(ProductCategory entity)
        {
            if (entity is null)
            {
                throw new ArgumentNullException(nameof(entity));
            }

            entity.Description = Description;
            entity.Name = Name;

            return entity;
        }
    }
}

using Store.Entities;
using Store.Services.Base;
using System;

namespace Store.Models
{
    public class ProductModel
    {
        public int Id { get; set; }
        public int Popularity { get; set; }

        public string Name { get; set; }
        public decimal Price { get; set; }
```

```
        public string Description { get; set; }
        public string ImageUrl { get; set; }

        public ProductCategoryModel Category { get; set; }

        public Product UpdateEntity(Product entity,
StoreServiceBase<ProductCategory> categoryService)
        {
            if (entity is null)
            {
                throw new ArgumentNullException(nameof(entity));
            }

            if (Category != null)
            {
                entity.Category = categoryService.Get(Category.Id);
            }

            entity.Description = Description;
            entity.Name = Name;
            entity.Price = Price;
            entity.ImageUrl = ImageUrl;

            return entity;
        }
    }
}

using System.Linq;
using Microsoft.EntityFrameworkCore;
```

```
using Store.Entities;
using Store.Services.Base;

namespace Store.Services
{
    public class OrdersService : StoreServiceBase<Order>
    {
        public OrdersService(StoreDbContext context) : base(context) { }

        public override Order Get(int id)
        {
            return this.context.Set<Order>()
                .Include("OrderProducts")
                .Include("OrderProducts.Product")
                .FirstOrDefault(x => x.Id == id);
        }
    }
}

using Microsoft.AspNet.OData.Builder;
using Microsoft.OData.Edm;
using Store.Models;

namespace Store
{
    public static class EdmModelBuilder
    {
        public static IEdmModel Build()
        {
            var builder = new ODataConventionModelBuilder();
```

```
            builder.EntitySet<OrderModel>("Orders");
            builder.EntitySet<ProductModel>("Products");
            builder.EntitySet<ProductCategoryModel>("ProductCategories");

            return builder.GetEdmModel();
        }
    }
}

using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;

namespace Store
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        public static IWebHostBuilder CreateWebHostBuilder(string[] args)
=>
            WebHost.CreateDefaultBuilder(args)
                .UseUrls("http://*:5000")
                .UseStartup<Startup>();
    }
}
```

```
using Microsoft.AspNet.OData.Extensions;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Store.Entities;
using Store.Services;
using Store.Services.Base;

namespace Store
{
   public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      public void ConfigureServices(IServiceCollection services)
      {
         services.AddDbContext<StoreDbContext>(options =>
options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")))
;
         services.AddOData();
```

```
services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
            services.AddCors(options =>
            {
                options.AddPolicy("AllowAll",
                    builder =>
                    {
                        builder
                            .AllowAnyOrigin()
                            .AllowAnyMethod()
                            .AllowAnyHeader()
                            .AllowCredentials();
                    });
            });

            services.AddScoped<StoreServiceBase<Order>, OrdersService>();
            services.AddScoped<StoreServiceBase<Product>>();
            services.AddScoped<StoreServiceBase<ProductCategory>>();
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment
env)
        {
            if (env.IsDevelopment())
                app.UseDeveloperExceptionPage();

            app.UseCors("AllowAll");
            app.UseMvc(routes =>
            {
```

```
                routes.MapRoute(name: "default", template:
"{controller}/{action=Index}/{id?}");

routes.Select().Expand().Filter().OrderBy().MaxTop(null).Count();
                });
                app.UseOData("odata", "odata", EdmModelBuilder.Build());
            }
        }
    }


    using Microsoft.EntityFrameworkCore;
    using Store.Entities;

    namespace Store
    {
        public class StoreDbContext : DbContext
        {
            public StoreDbContext(DbContextOptions options) : base(options) { }

            public DbSet<Order> Orders { get; set; }
            public DbSet<Product> Products { get; set; }
            public DbSet<ProductCategory> ProductCategories { get; set; }

            protected override void OnModelCreating(ModelBuilder
modelBuilder)
            {
                base.OnModelCreating(modelBuilder);

                modelBuilder.Entity<OrderProduct>()
```

```
            .HasOne(x => x.Order)
            .WithMany(x => x.OrderProducts)
            .HasForeignKey(pt => pt.OrderId);

        modelBuilder.Entity<OrderProduct>()
            .HasOne(x => x.Product)
            .WithMany(x => x.OrderProducts)
            .HasForeignKey(x => x.ProductId);
    }
  }
}
```

Додаток В – Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доц. кафедри ПЗ _____ В.В. Войтко

Рецензент, д.т.н., проф. кафедри КН _____ О.М. Василевський

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____ В.В. Войтко

Виконавець, студентка групи 1ПІ–19м _____ А. О. Дмитрук

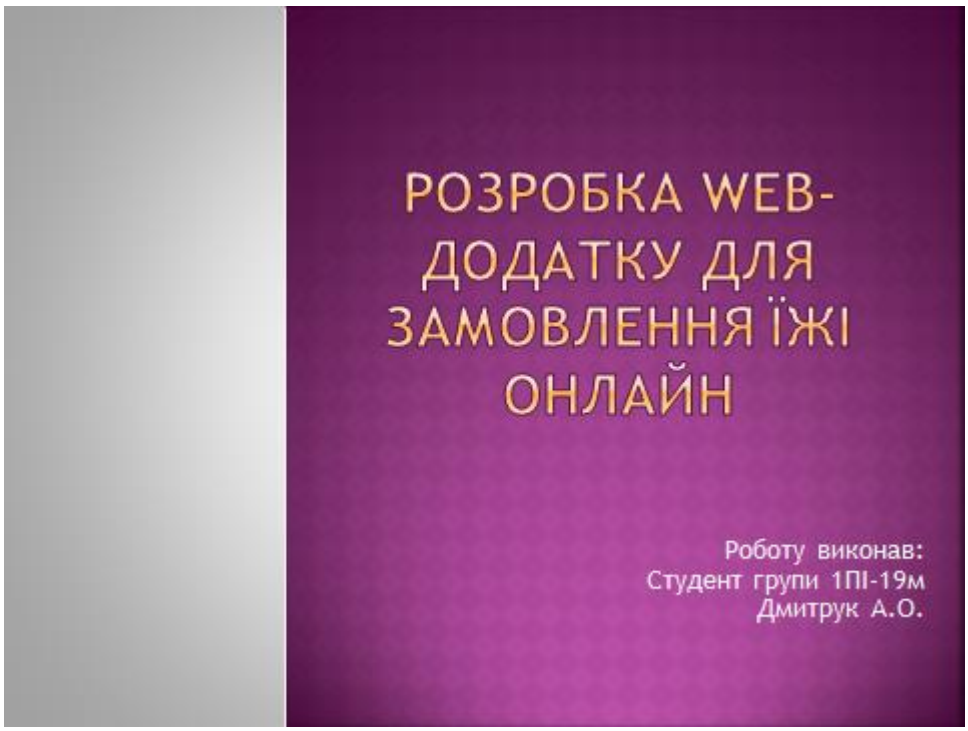

Рисунок В.1- Титульна сторінка

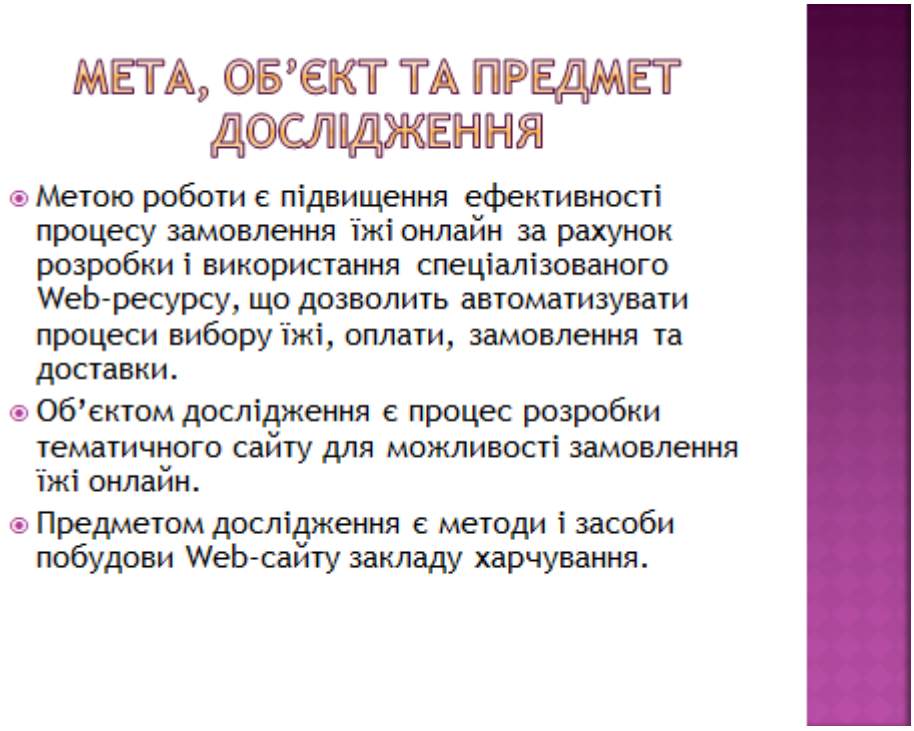

Рисунок В.2 – Мета, об'єкт та предмет дослідження

НАУКОВА НОВИЗНА

- Подальшого розвитку дістав метод підвищення інтерактивності обслуговування закладів харчування в онлайн режимі,який, на відміну від існуючих орієнтований на створення і використовування інтерактивних об'єктів інтерфейсу, що дозволяє забезпечити ефективний зворотній зв'язок з користувачем.
- Подальшого розвитку дістали моделі реалізації Web-системи , які навідміну від існуючих, використовуюсь розширення категоризацію об'єктів та орієнтовані на використання інтерактивної взаємодії з користувачем у процесі вибору об'єктів і формуванн кошика замовлення, що дозволяє розширити можливості Web-системи з урахуванням зворотнього зв'язку з користувачем.

Рисунок В.3- Наукова новизна

АНАЛІЗ АНАЛОГІВ

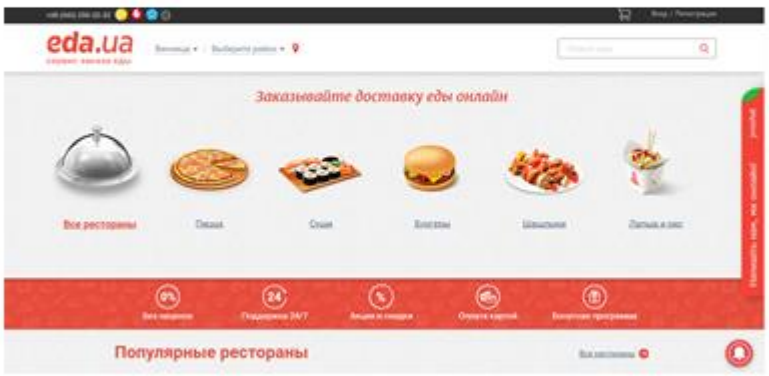

Рисунок В.4- Аналіз аналогів

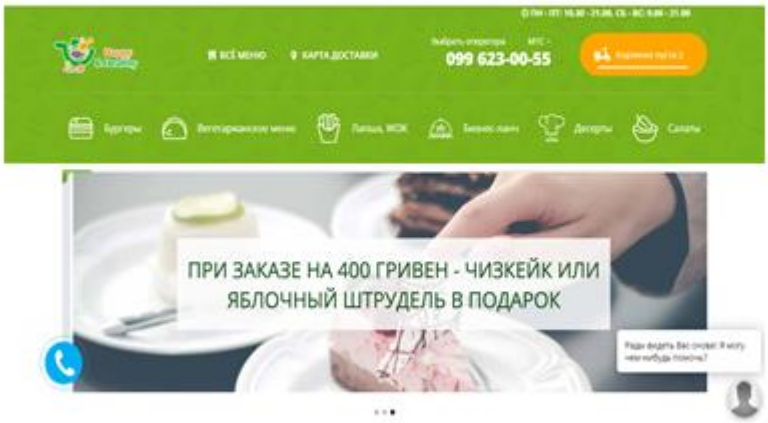

Рисунок В.5- Аналоги

Рисунок В. 6- Взаємодія між компонентами БД

Рисунок В.7-Алгоритм заповнення замовлення

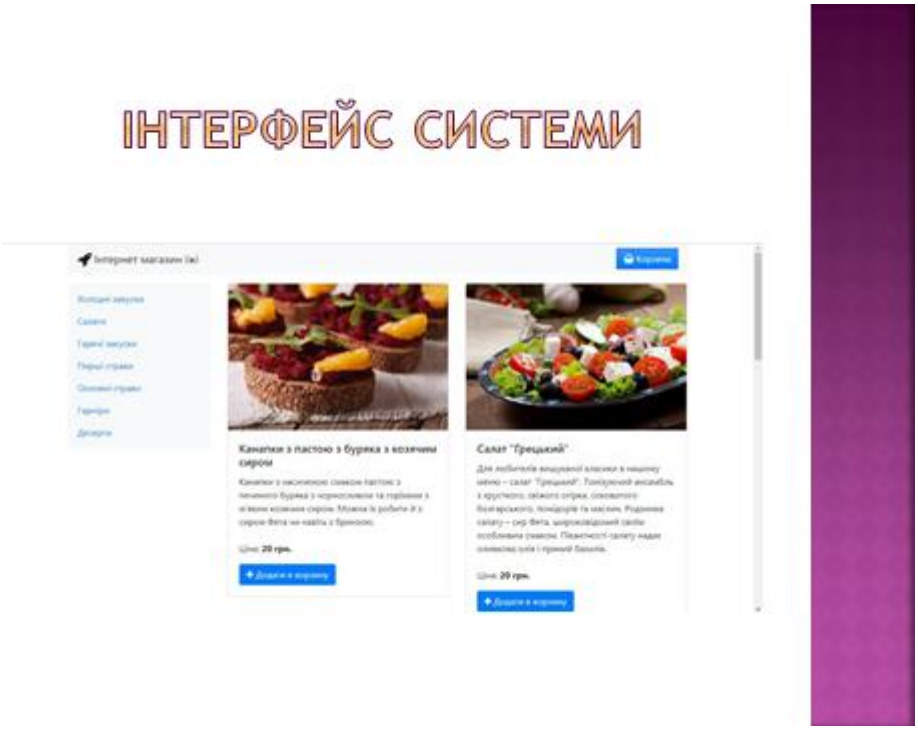

Рисунок В.8- Інтерфейс системи

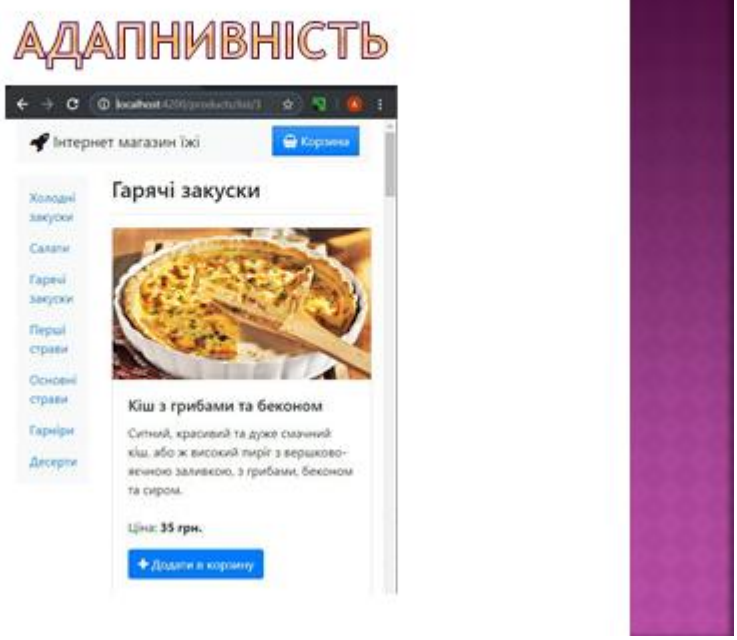

Рисунок В. 9- Адаптивність

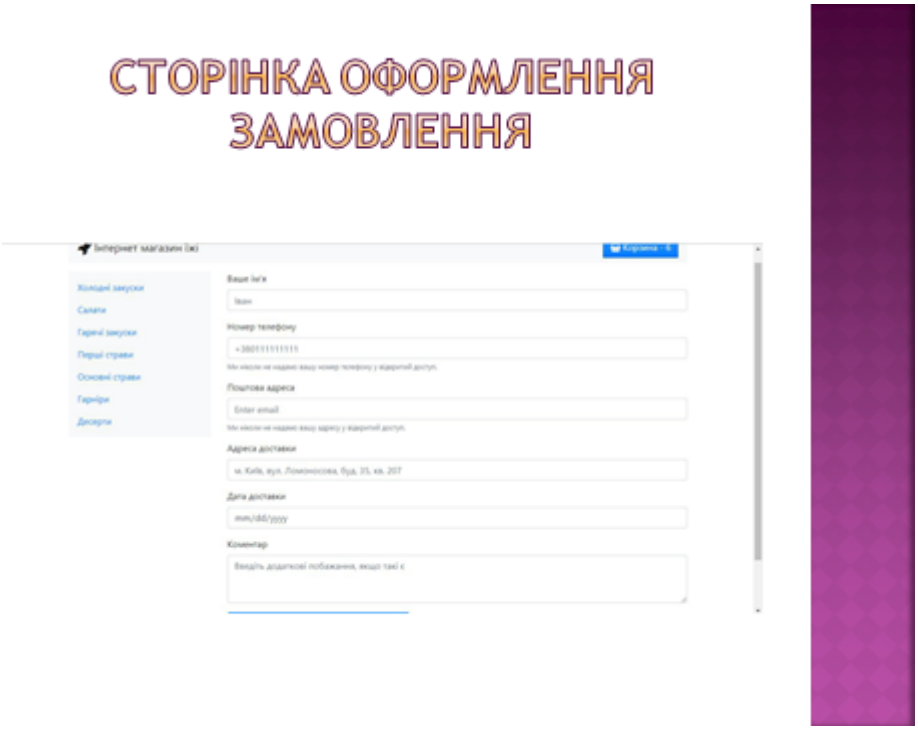

Рисунок В.10- Сторінка оформлення замовлення

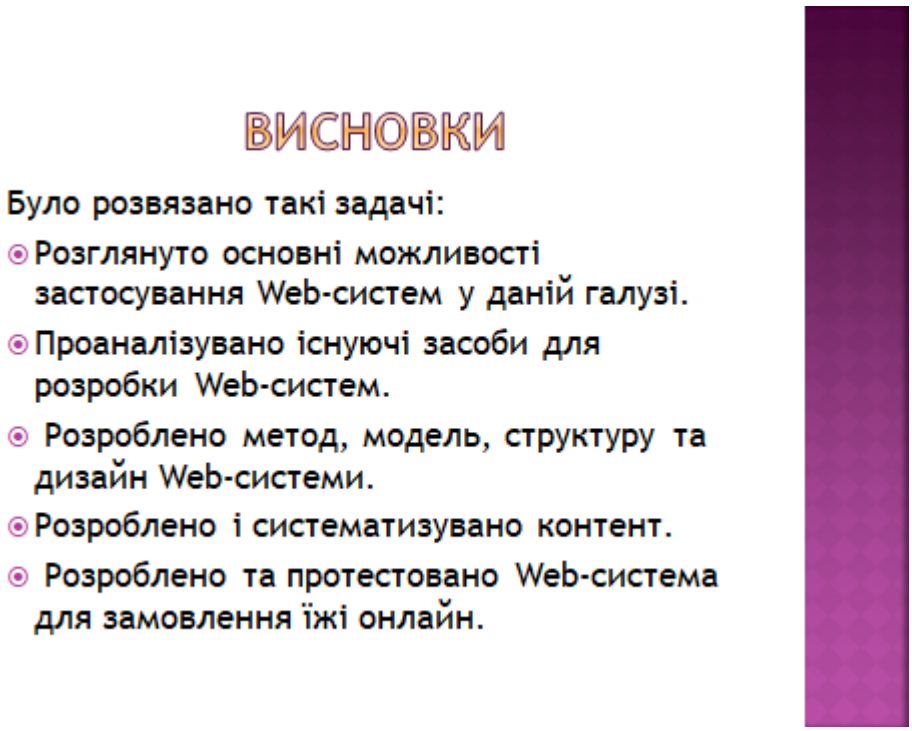

Рисунок В.11- Висновки

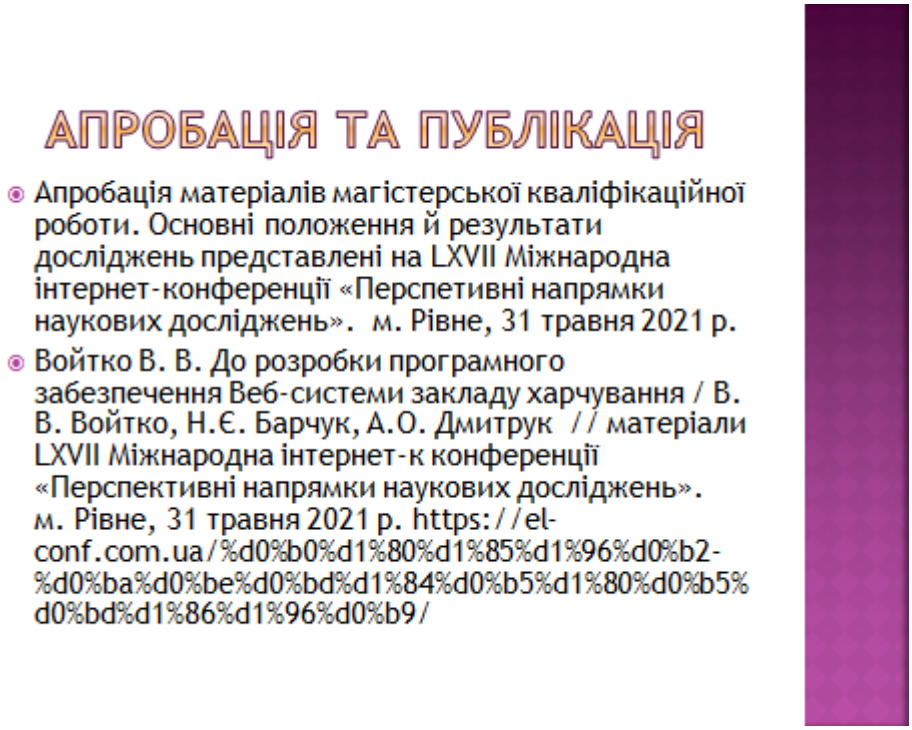

Рисунок В.12- Апробація та публікація