МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

Веб-сайт "Delivery Food"

Виконав студент групи ІПЗк-20-1
ПЛАХОТНЮК Дмитро Володимирович

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«2» серпня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПЛАХОТНЮК Дмитро Володимирович**
Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**
Тема роботи: **«Веб-сайт "Delivery Food"»,**
затверджена Наказом закладу вищої освіти від **«2» серпня 2023 р. №326/с**
Вихідні дані для роботи: Об’єктом дослідження є розробка веб-сайту для перегляду різної продукції та її покупки .
Предметом дослідження є використання сучасних технологій розробки веб-сайту для виконання поставленої мети.
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	02.08.2023р.	02.08.2023р.
2	Чижмотря О.В.	02.08.2023р.	02.08.2023р.
3	Чижмотря О.В.	02.08.2023р.	02.08.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	3 серпня 2023 – 4 серпня 2023	виконано
2	Пошук та аналіз аналогічних веб-додатків	5 серпня 2023 – 7 серпня 2023	виконано
3	Обирання технологій для розробки веб-додатку	8 серпня 2023 – 11 серпня 2023	виконано
4	Планування структури програми	12 серпня 2023 – 18 серпня 2023	виконано
5	Проектування та реалізація алгоритмів веб-системи	19 серпня 2023 – 20 серпня 2023	виконано
6	Розробка та реалізація практичної частини веб-додатку	21 серпня 2023 – 24 серпня 2023	виконано
7	Тестування роботи веб-додатку	25 серпня 2023 – 25 серпня 2023	виконано
8	Оформлення пояснювальної записки	26 серпня 2023 – 29 серпня 2023	виконано
9	Попередній захист	25 серпня 2023	

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплекса веб-сайту "Delivery Food" та пояснювальної записки. Пояснювальна записка до випускної роботи містить 80 сторінки, 32 ілюстрацій, 6 таблиць, 1 додаток та 15 використаних джерел.

Метою роботи є розробка веб-сайту "Delivery Food" засобами HTML, CSS, JavaScript та за допомогою текстового форматом обміну даними JSON.

В роботі було визначено основні завдання на розробку веб-сайту, проаналізовано аналоги розробленої системи, визначено основний функціонал та вимоги до системи. Розглянуто методику роботи служб страхових компаній, також було обрано технології розробки веб-сайту, спроектовано та реалізовано алгоритми роботи системи. Програмний продукт був протестований.

КЛЮЧОВІ СЛОВА: САЙТ З ЗАМОВЛЕННЯМ ЇЖІ, ВЕБ-САЙТ, JS, JSON, DELIVERY FOOD.

ABSTRACT

The bachelor thesis consists of the software complex of the website "Delivery Food" and an explanatory note. The explanatory note to the final thesis contains 80 pages, 32 illustrations, 6 table, 1 appendix and 15 used sources.

The purpose of the work is to develop the "Food Delivery" site using HTML, CSS, JavaScript and using the JSON text data exchange format.

In the work, the main tasks for the development of the website were defined, analogues of the developed system were analyzed, the main functionality and requirements for the system were determined. The working methods of insurance company services were considered, website development technologies were also chosen, and the algorithms of the system were designed and implemented. The software product has been tested.

KEY WORDS: FOOD ORDERING SITE, WEBSITE, JS, JSON, DELIVERY FOOD.

Зміст

ВСТУП **8**

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-САЙТУ "DELIVERY FOOD" **10**

1.1. Постановка задачі 10

1.2. Аналіз аналогів програмного продукту 11

1.3. Вибір архітектури веб-сайту "Delivery Food" 15

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення 19

Висновки до розділу 1 22

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ "DELIVERY FOOD" **23**

2.1. Визначення варіантів використання системи 23

2.2. Алгоритм роботи системи 26

2.3 Структура компонентів системи 30

Висновки до розділу 2 31

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ **32**

3.1. Архітектура додатку та його розгортання 32

3.2. Інтерфейс та порядок роботи з системою 33

3.3. Тестування роботи програмного комплексу 41

Висновки до розділу 3 43

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **45**

ДОДАТКИ **47**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC – Model-View-Controller (Модуль-Представлення-Компонент)

JS – JavaScript

JSON – JavaScript Object Notation

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

ВСТУП

Актуальність теми. Веб-сайт для замовлення їжі дуже актуальний у сучасну епоху цифрових технологій, коли люди все більше покладаються на Інтернет для задоволення своїх щоденних потреб, зокрема їжі. Зі зростанням популярності онлайн-платформ доставки їжі попит на веб-сайти для замовлення їжі значно зріс за останні роки.

Веб-сайти для замовлення їжі пропонують клієнтам зручність, дозволяючи їм переглядати меню, розміщувати замовлення та здійснювати платежі онлайн, не виходячи з дому. Це не тільки економить час, але й забезпечує безперебійну роботу користувача.

Більше компаній та стартапів входять на цей ринок, впроваджуючи нові функції та технології, такі як штучний інтелект для рекомендацій, оптимізовані маршрути доставки тощо. Дослідження актуальних інновацій може додати цінності роботи. [1]

Тема надає безліч можливостей для дослідження, починаючи від дизайну інтерфейсу користувача і оптимізації процесу замовлення до аналізу даних про переваги клієнтів.

Загалом, веб-сайт із замовленням їжі є дуже актуальною та важливою темою в сучасному швидкому цифровому світі, де високо цінуються зручність та ефективність.

Метою кваліфікаційної роботи є дослідження різних аспектів веб-сайтів із замовленням їжі, таких як їх дизайн, досвід користувача, функціональність і вплив на харчову промисловість.

Встановлена мета обумовлює наступні завдання веб-сайту:

– створення та керування меню;

– керування користувачами;

– розробка інформаційного та математичного забезпечення системи;

– проектування структурних складових та алгоритмів роботи системи;

Об’єктом дослідження є вивчення функцій веб-сайту, дизайну та взаємодії з користувачем, а також його впливу на ресторанну індустрію, клієнтів і суспільство в цілому.

Предметом дослідження є використання Web-орієнтованих технологій для розробки сайту з замовленням їжі.

В роботі було використано методи об’єктна орієнтованого проектування, web-програмування та дизайну.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-САЙТУ "DELIVERY FOOD"

1.1. Постановка задачі

Постановка завдання сайту замовлення їжі передбачає визначення конкретних цілей і завдань, на досягнення яких спрямований сайт. Нижче наведено деякі звичайні завдання, які може виконувати сайт із замовлення їжі:

1. Надання зручного інтерфейсу.

1.1. Сайт має бути простим у навігації та використанні, з чітким та інтуїтивно зрозумілим макетом, який дозволяє користувачам переглядати меню, розміщувати замовлення та здійснювати платежі.

2. Пропонувати різноманітні варіанти харчування.

2.1. Пропонувати широкий вибір варіантів їжі, щоб задовольнити різні смаки та вподобання, включаючи варіанти дієтичних обмежень і алергії.

3. Забезпечення своєчасної доставки.

3.1. Якщо сайт пропонує послуги доставки, він повинен гарантувати, що замовлення доставляються вчасно та в хорошому стані, забезпечуючи чіткий зв’язок і можливості відстеження для клієнтів.

4. Забезпечення безпечних варіантів оплати.

4.1. Захищають особисту та фінансову інформацію користувачів, наприклад зашифровані платіжні шлюзи та двофакторну автентифікацію.

5. Створення лояльної клієнтської бази

6. Оптимізація операцій

6.1. Мінімізувати помилки та затримки, наприклад, шляхом автоматизації обробки замовлень і оптимізації маршрутів доставки.

7. Тестування системи управління контентом.

Встановлюючи чіткі цілі та завдання для сайту замовлення їжі, він може гарантувати, що він задовольняє потреби своїх клієнтів, а також досягає своїх бізнес-цілей.

1.2. Аналіз аналогів програмного продукту

Останнім часом сайти для замовлення їжі набувають все більшої популярності завдяки своїй зручності та простоті використання. Ці сайти дозволяють клієнтам замовляти їжу зі своїх улюблених ресторанів онлайн, не виходячи з дому чи офісу. З появою смартфонів і мобільних додатків сайти для замовлення їжі також стали доступними в дорозі, дозволяючи клієнтам робити замовлення будь-де в будь-який час.

Пандемія COVID-19 також зіграла свою роль у популярності сайтів для замовлення їжі. Оскільки багатьом ресторанам довелося закрити свої двері або обмежити кількість місць, все більше клієнтів звернулися до онлайн-замовлень, щоб отримати улюблені страви прямо до порогу. Пандемія також прискорила тенденцію до безготівкових розрахунків, що часто є особливістю сайтів для замовлення їжі. Крім того, сайти для замовлення їжі стали популярними, оскільки пропонують більший вибір страв, ніж традиційні ресторанні послуги доставки. Клієнти можуть робити замовлення з кількох ресторанів через єдиний сайт або додаток, що дозволяє їм вибирати з цілого ряду кухень і пунктів меню.

Загалом, зручність, простота використання та різноманітність сайтів для замовлення їжі зробили їх популярним вибором для багатьох клієнтів.

Розглянемо приклади сайтів з замовленням їжі:

Першим розгляну мережу магазинів в Україні «KILOGRAMM SUSHI PROJECT». KILOGRAMM Sushi Project – це суші-ресторан або служба доставки в Україні, яка пропонує онлайн-замовлення та доставку суші кілограмами. Клієнти можуть переглядати меню та вибирати суші, які їм потрібні, а потім вказувати кількість у кілограмах, яку вони бажають замовити.

Після оформлення замовлення «KILOGRAMM Sushi Project» приготує суші та доставить їх у вказане замовником місце по всій Україні.

Сайт має простий вигляд і інтерфейс. В ньому в основному можна замовити суші, що можна зрозуміти з назви. Ціна залежить від міста де

знаходиться, саме місто можна змінити. Реєструватися на сайті не має можливості. Вигляд меню на сайті зображено на (рис. 1.1) [3]

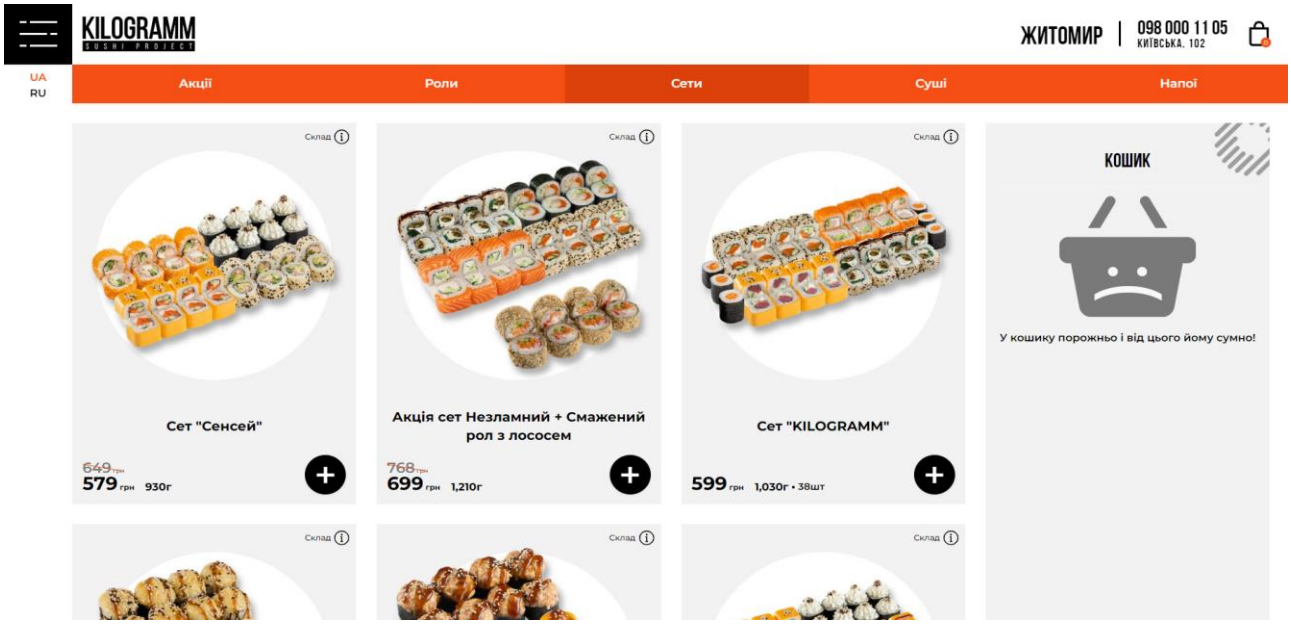

Рис. 1.1 Вигляд сторінки з меню

Замовлення можна виконанти додавши щось до кошика з меню. Після додавання чого небуть до кошика він зразу з'явиться з правої сторони і буде показувати все, що ви додали до кошика. Потім натиснувши на «Замовити», відкриється сторінка вона зображена на (рис. 1.2). Там одразу можна помітити карту на якій зображено зону міста на яку можливо виконати доставку. При заповненні необхідних даних можна виклнати замовлення.

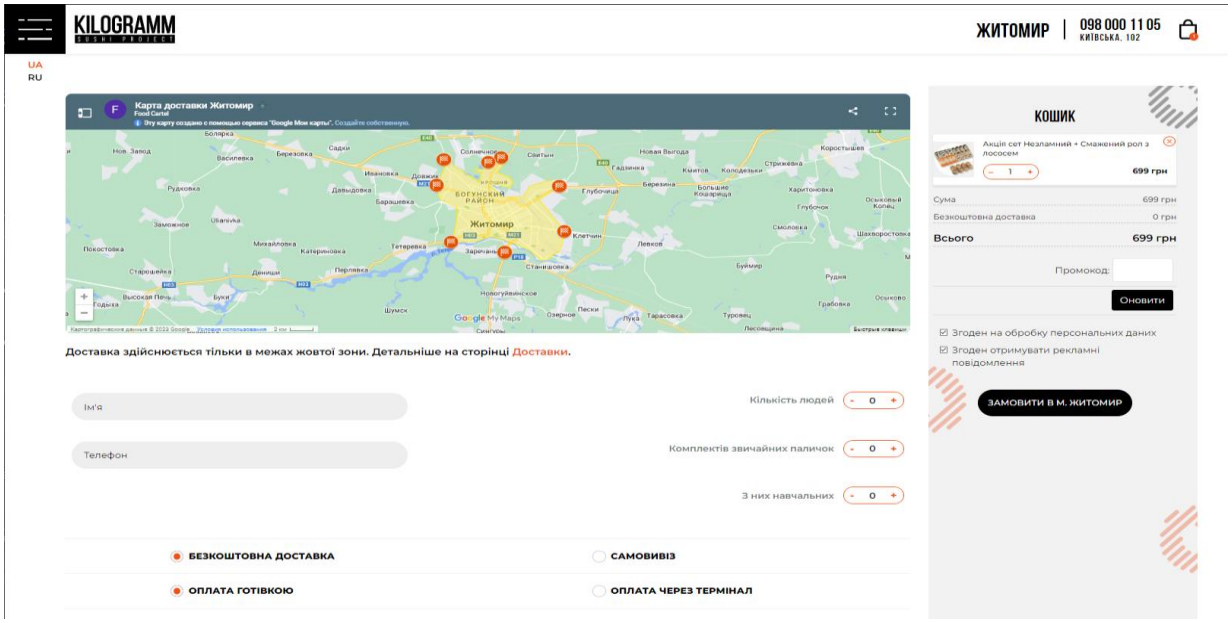

Рис. 1.2 Вигляд сторінки з замовленням

Далі розгляну сайт зі замовлення під назвою «Хочу З'їсти» він працює тільки у місті Житомир. Має різний невеличкий асортимент меню для замовлення вигляд меню зображени на (рис 1.3)

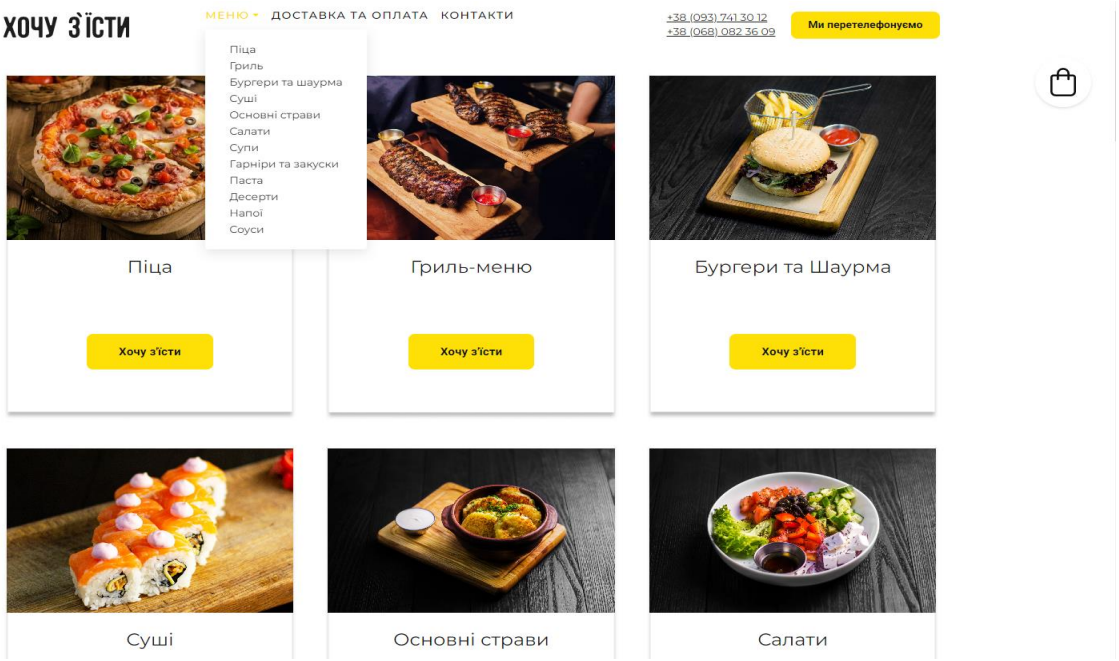

Рис. 1.3 Вигляд головної сторінки і меню

Головна сторінка сайту починається одразу з меню. Можливість реєструватися на сайті немає. Оформлення замовлення виконується дуже просто додає все, що хочеш замовити в кошик, після цього заходиш в самий кошик і всю необхідну інформацію для замовлення. Вигляд кошика зображена на (рис 1.4) [2]

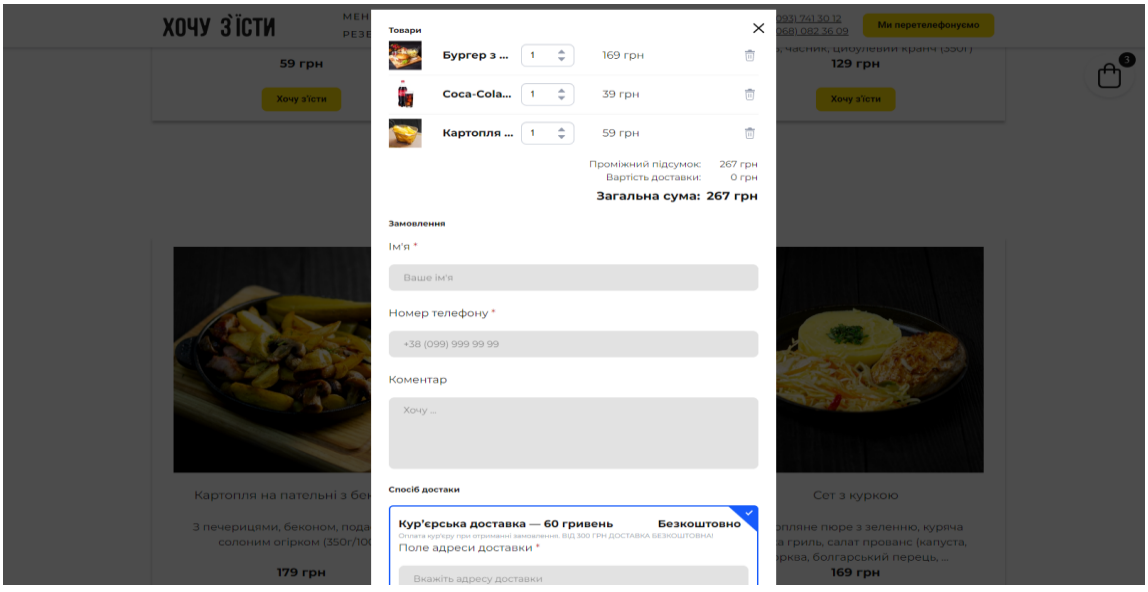

Рис. 1.4 Вигляд кошика на сайту «Хочу З'їсти»

«Salateira» – це мережа фаст фудів здорового харчування в Київі. Має невеликий асортимент і конструктор в якому можна створити власну страву. Також ресторан має власний додаток де також можна замовляти їжу. Нас сайті можна зареєструваися по номеру телефона. Головна сторінка сайту зображена на (рис. 1.5) [4]

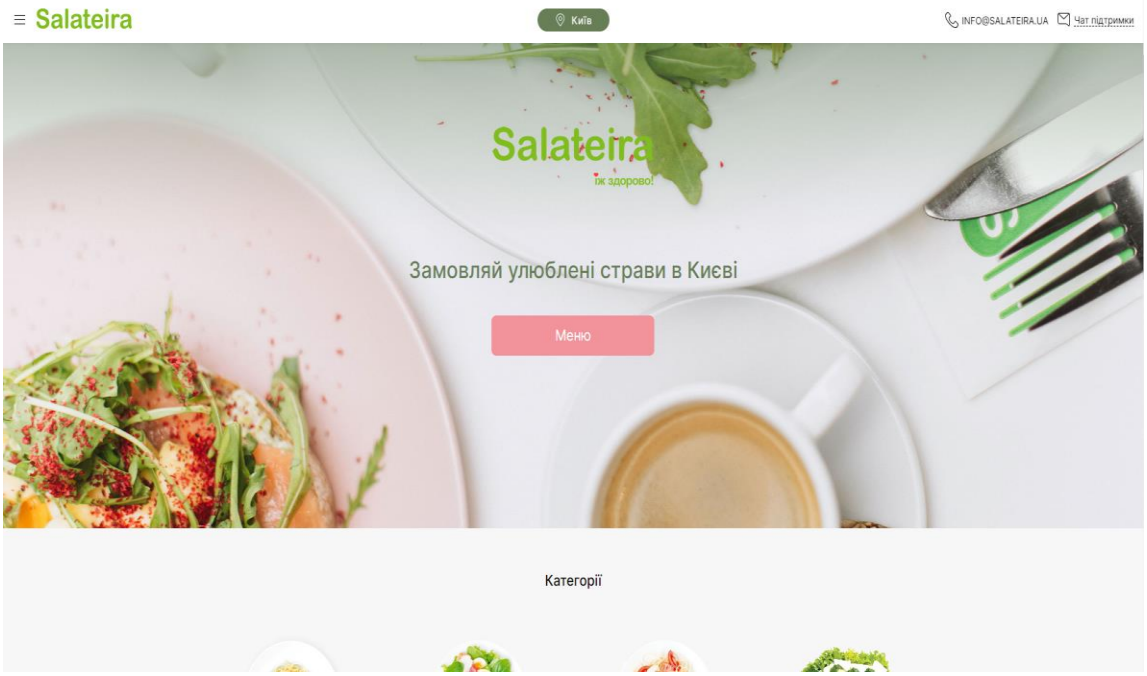

Рис. 1.5 Головна сторінка сайту «Salateira»

Зробити замовлення можна без реєстраії, просто потрібно додати товар до кошику та оформити його заповнивши дані, що зображені на (рис. 1.6)

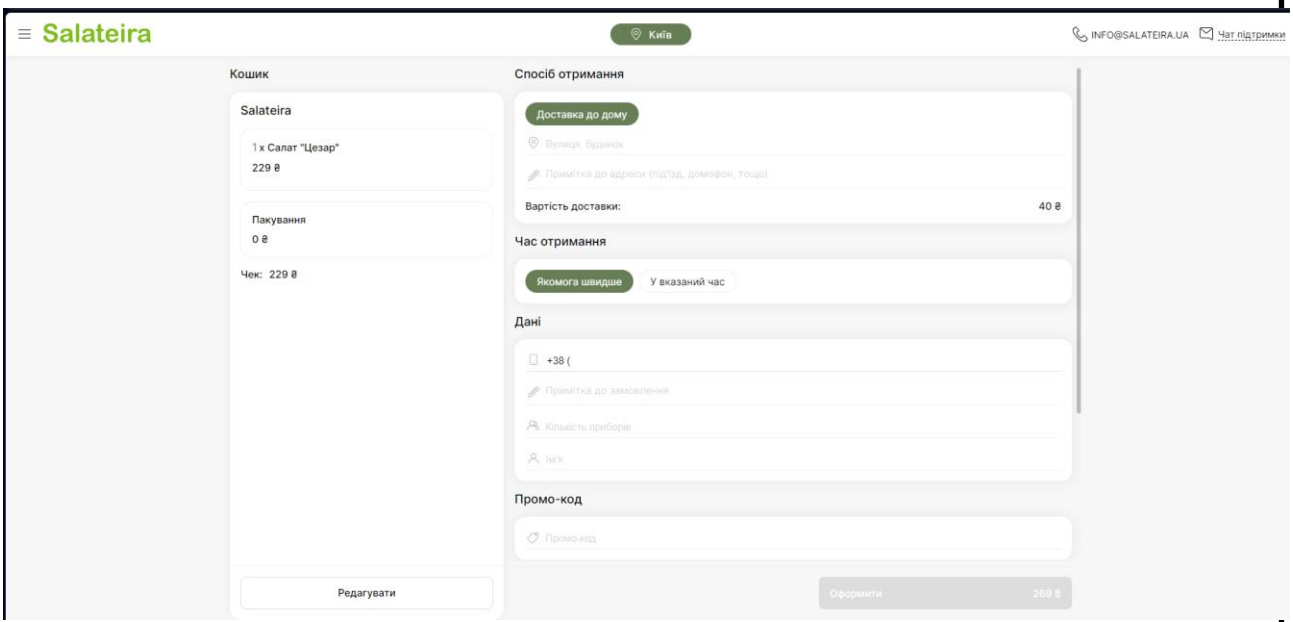

Рис. 1.6 Вигляд кошика при оформленні замовлення

Переглянувши приклади сайтів з замовленням їжі, що зображені вище була створення таблиця (табл. 1.1) основних функцій та контенту який може бути в кваліфікаційної роботи.

Таблиця 1.1

Порівняння сайтів з замовлення їжі

Порівняльні характеристики	KILOGRAMM SUSHI PROJECT	Хочу З'їсти	Salateira
Зручність у використанні	+	+	+
Різноманітність харчування	-	+	-
Обслуговування клієнтів	-	+	+
Наявність додатку	-	-	+
Наявність реєстрації	-	-	+
Відгуки та оцінка	-	+	-
Знижки та акції	+	-	-

1.3. Вибір архітектури веб-сайту "Delivery Food"

Архітектура веб-програми — це схема того, як різні компоненти веб-програми, такі як бази даних, програми та системи проміжного програмного забезпечення, взаємодіють один з одним і обмінюються даними.

Він визначає протокол для обміну даними через Інтернет, гарантуючи, що клієнтська та серверна сторони можуть зрозуміти й обробити їх.

Існує кілька типів архітектур веб-додатків, які слід розглянути, включаючи монолітну, мікросервісну, безсерверну та трирівневу. Кожен має свій набір переваг і недоліків.

Добре спроектована архітектура веб-програми має важливе значення для продуктивності, масштабованості та зручності обслуговування веб-програми. Це допомагає створити програму, яка може легко адаптуватися до мінливих

потреб бізнесу, обробляти високий рівень трафіку та забезпечувати безперебійну роботу користувача.

Архітектура веб-додатку зазвичай складається з трьох основних частин:

Веб-браузер: це клієнтський компонент, який дозволяє користувачам переглядати веб-вміст, взаємодіяти з ним і отримувати доступ до нього. Він надсилає HTTP-запит на веб-сервер і відображає результати для користувача.

Веб-сервер: це внутрішній компонент, який отримує запити від клієнта та обробляє їх за допомогою бізнес-логіки. Він отримує дані з бази даних і передає їх у веб-браузер для відображення.

Сервер бази даних: цей компонент зберігає та отримує дані структурованим і ефективним способом, відповідаючи на запити користувачів і надаючи необхідні дані для відображення в інтерфейсі користувача. [5]

Існує кілька поширених архітектур для створення та організації компонентів веб-сайту, зокрема:

Монолітна архітектура. Ця архітектура має підхід до розробки програмного забезпечення, при якому весь додаток розглядається як єдиний компонент, який розгортається та працює разом. У монолітній архітектурі весь код, логіка та функціональність розміщуються в одній програмі або одному додатку.

Її перевагами є простота в розробці та тестуванні, оскільки всі компоненти додатку взаємодіють безпосередньо один з одним, що дозволяє швидко та легко вирішувати проблеми.

А недоліком є важке масштабування окремих частин додатку незалежно, що може призводити до витрат на ресурси там, де це необхідно, а у разі потреби використання різних технологій для різних компонентів додатку можуть виникнути обмеження, ще велика монолітна програма може бути важкою для розгортання та керування, особливо якщо вона має багато залежносте.

Архітектура мікросервісів. Є підходом до розробки програмного забезпечення, при якому додаток розбивається на невеликі, незалежні та

самодостатні компоненти, відомі як мікросервіси. Кожен мікросервіс відповідає за виконання конкретної функції або послуги і взаємодіє з іншими мікросервісами через легкі механізми комунікації, такі як API.

За допомогою цієї архітектури усуваються будь-які довгострокові зобов'язання щодо технології. При розробці нового сервісу ви можете вибрати новий стек технологій. Будь-який сервіс у системі можна замінити. Його можна переписати з нуля в межах прийнятного часу та бюджету без необхідності перебудовувати всю систему. Її недоліком є складність в управлінні ,селадність в управлінні даними та консинстентності і має комунікаційне та мережну складність. [6]

Безсерверна архітектура описує архітектури, в яких компанії або зацікавлені сторони ефективно передають обробку даних серверами сторонній стороні. Експерти зазначають, що безсерверна архітектура не означає, що немає серверів, які беруть участь в обробці даних - це просто означає, що компанія звільняє себе від відповідальності за управління та догляд за серверами.

З розвитком хмарних обчислень та програмного забезпечення в якості сервісу (SaaS) виробники розробили сервіси, які дозволять компаніям використовувати архітектури без серверів. Деякі з них можуть називатися доповненням як послуга або як якась форма надання хмарних послуг. Одним з відомих і популярних прикладів є веб-сервіс Amazon (AWS). AWS є популярним корпоративним вибором для програмного забезпечення як сервісу і пропонує послуги, які вважають себе архітектурними рішеннями без сервера. По суті, компанії можуть приймати дані з серверів AWS, щоб їм не потрібно було підтримувати свої власні. Це приносить очевидні переваги з точки зору витрат, ефективності та нижчого тягаря відповідальності за технічне обслуговування обладнання. [7]

Трьохрівнева архітектура - це підхід до розробки програмного забезпечення, який розбиває додаток на три логічно розділені рівні або шари. Кожен рівень відповідає за певний аспект функціональності додатку та має свою визначену роль. Ця архітектурна модель спрощує розробку, тестування та

підтримку додатку, оскільки різні рівні можуть працювати незалежно один від одного.

Основні рівні трьохрівневої архітектури:

1. Представлення (Presentation Layer): Цей рівень відповідає за взаємодію з користувачем. Він включає інтерфейс користувача, такий як веб-сторінки, мобільні додатки або графічний інтерфейс. Завданням цього рівня є відображення даних та обробка користувацьких дій.

2. Бізнес-логіка (Business Logic Layer): Цей рівень містить бізнес-логіку додатку. Він відповідає за обробку та аналіз даних, здійснення різних операцій та логічних перевірок, пов'язаних з основною функціональністю додатку. Тут приймаються рішення на основі обробки даних.

3. Доступ до даних (Data Access Layer): Цей рівень відповідає за взаємодію з базою даних або іншими джерелами даних. Він забезпечує зберігання та отримання даних, необхідних для бізнес-логіки та відображення. В цьому рівні зазвичай реалізовані запити до бази даних та операції з даними.

Трьохрівнева архітектура дозволяє знизити залежність між різними аспектами додатку, полегшити розробку, підтримку та масштабування. Кожен рівень може бути розроблений та підтриманий незалежно, що полегшує роботу над проектом командам розробників. [8]

Архітектура MVC (Model-View-Controller) - це підхід до організації програмного забезпечення, який розділяє додаток на три основних компоненти: Model, View та Controller. Кожен з цих компонентів має визначену роль та відповідальність, що сприяє розділенню логіки та спрощує розробку та підтримку додатку.

Основні компоненти архітектури MVC:

Модель (Model): Цей компонент відповідає за управління даними, бізнес-логікою та станом додатку. Модель представляє собою абстракцію даних, з якими працює додаток. Вона також відповідає за виконання операцій з даними та взаємодію з базою даних.

Вид (View): Відповідальність виду полягає в відображенні даних користувачу та збірці вхідних даних від користувача. Він представляє інтерфейс користувача та відображає дані, які надаються моделлю. Вид також може відправляти повідомлення контролеру про дії користувача.

Контролер (Controller): Контролер відповідає за обробку взаємодії між користувачем та додатком. Він приймає вхідні дії користувача через вид, взаємодіє з моделлю для отримання чи оновлення даних та визначає, який вид має бути показаний.

Принципом MVC є розділення відповідальностей між цими трьома компонентами. Це дозволяє знизити залежність між різними частинами додатку, полегшує розробку та тестування, а також дає можливість робити зміни в одному компоненті, не впливаючи на інші. MVC є популярним підходом в багатьох фреймворках та технологіях розробки програмного забезпечення.Архітектура клієнт-сервер. В цій архітектурі веб-додаток складається з двох основних компонентів: клієнтської частини, яка забезпечує інтерфейс для користувача, і серверної частини, яка обробляє логіку додатку та забезпечує доступ до даних.[9]

Переглянувши різни види архітектур можливо сказати, що через нехват часу для виконання кваліваційної роботи буду використовувати архітектуру простого статичного веб-сайту.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Існує багато різних технологій для створення веб-сайту. В нашому випадку була обрана мова програмування JavaScript. JavaScript - це високорівнева мова програмування, яка використовується для розробки динамічних та інтерактивних веб-сайтів. Вона дозволяє вставляти скрипти безпосередньо в HTML-код веб-сторінок, що дозволяє змінювати вміст

сторінки, відгуки на події користувача та взаємодіяти з сервером без потреби перезавантаження сторінки. [10]

Обґрунтування вибору JavaScript:

1. Клієнт-серверний характер вебу: JavaScript є мовою програмування, яка виконується на боку клієнта, тобто на браузері користувача. Це дозволяє створювати динамічні та інтерактивні веб-сайти без необхідності відправляти запити на сервер для кожної дії.

2. Широкий розповсюджений інструментарій: JavaScript має величезну кількість бібліотек і фреймворків, таких як React, Angular, Vue.js для створення інтерфейсів користувача, а також Node.js для серверної розробки. Це дозволяє розробникам ефективно створювати як фронтенд, так і бекенд додатки, використовуючи одну мову.

3.Зручність розробки і тестування: Розробка на JavaScript має широку підтримку розробничих інструментів, таких як редактори коду, інтегровані середовища розробки (IDE), системи контролю версій та інші. Також наявність багатьох інструментів для тестування сприяє покращенню якості продукту.

Основними інструменти, які буть використані для розробки веб-сайту "Delivery Food" на мові JS:

- CSS (Cascading Style Sheets) - це мова стилізації, яка використовується для оформлення та вигляду веб-сторінок. CSS визначає зовнішній вигляд елементів HTML, таких як кольори, розміри шрифтів, відступи, рамки, позиціонування та інші аспекти, які допомагають створювати естетичний та зручний інтерфейс для користувача. [11]
- HTML (HyperText Markup Language) - це стандартна мова розмітки для створення та відображення веб-сторінок у браузерах. Вона використовується для структурування та організації вмісту веб-сторінки, включаючи текст, зображення, відео, посилання, таблиці, форми та інші елементи.

- JSON (JavaScript Object Notation) - це легкий формат обміну даними, який використовується для структурування і передачі інформації між програмами. JSON використовується для зберігання і передачі структурованих даних, таких як об'єкти, масиви, рядки, числа, булеві значення та значення null. Синтаксис JSON дуже схожий до синтаксису об'єктів та масивів в багатьох мовах програмування. Він широко використовується для передачі даних між веб-серверами та клієнтськими додатками, для зберігання конфігураційних файлів та для зв'язку між різними програмними компонентами. [12]

Середовище розробки буде використовуватися WebStorm 2022.2.

WebStorm - це інтегроване середовище розробки (IDE) від компанії JetBrains, призначене для розробки веб-додатків на різних мовах програмування, таких як JavaScript, HTML, CSS, а також їхніх фреймворках та бібліотеках.

WebStorm надає набір інструментів для полегшення розробки веб-додатків, включаючи підсвічування синтаксису, автодоповнення коду, інспекцію коду, вбудовані інструменти для роботи з Git та іншими системами контролю версій, налаштування оточення для різних фреймворків та бібліотек, тестування та налагодження коду.

Отже, для створення веб-додатку було обрано такі інструментальні засоби та технології:

– мова гіпертекстової розмітки документів HTML;

– каскадна таблиця стилів CSS;

– мову програмування JavaScript;

– для зберігання даних JSON;

Використано такі програмні продукти: StarUML; середовище розробки WebStorm 2022.2.

Висновки до розділу 1

Отже, в першому розділі переглянувши список аналогів сайтів з замовленням їжі, можна зробити висновки, що головними функціями майбутнього проекту можуть бути: спосіб переглядати товари, оформлювати замовлення, пошук та загальна інформація про види страхування і можливо про компанію. На основі аналізу було зроблено висновок, щодо функціоналу веб-додатку.

При огляді різної архітектури зважуючи на мінуси в часі на розробку буду використовувати архітектуру простого статичного веб-сайту.

Для реалізації було обрано для створення програмного комплексу такі інструментальні засоби та технології:

- мова гіпертекстової розмітки документів HTML;
- каскадна таблиця стилів CSS;
- мову програмування JS;
- для зберігання даних JSON;

Використано такі програмні продукти: StarUML; середовище розробки WebStorm 2022.2.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ "DELIVERY FOOD"

2.1. Визначення варіантів використання системи

Діаграма варіантів використання (Use Case Diagram) - це однин з ключових інструментів у сфері аналізу вимог і моделювання програмного забезпечення. Вона використовується для візуалізації функціональності системи та взаємодії між користувачами та самою системою.

Діаграма варіантів використання складається з користувачів, варіантів використання та зв'язків між ними. Користувачі представляють ролі, які взаємодіють з системою, наприклад, користувачі, зовнішні системи або інші компоненти. Варіанти використання описують конкретні дії або сценарії, які можуть бути виконані акторами у системі. [13]

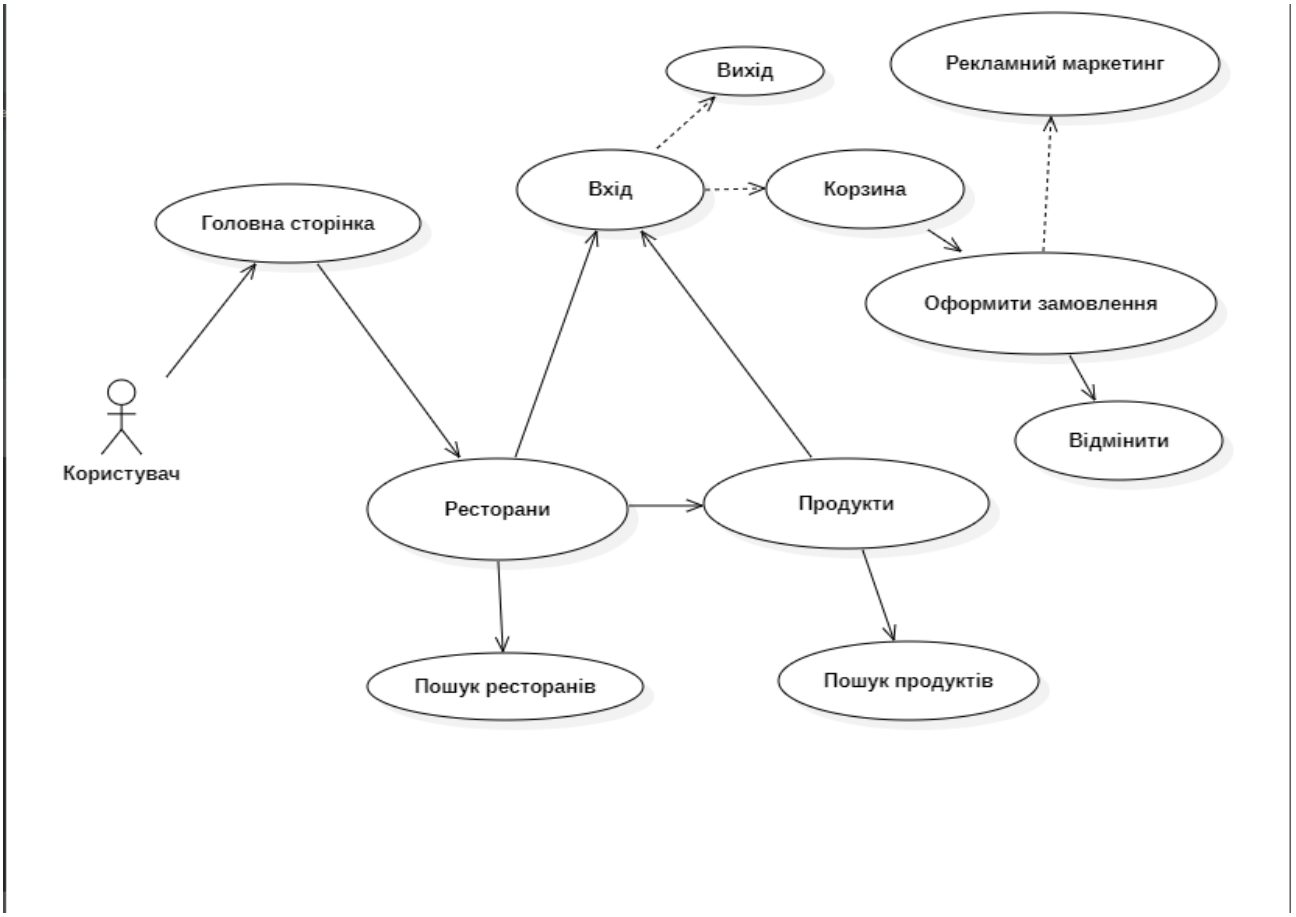

Рис. 2.1 – Діаграма варіантів використання

Таблиця 2.1

Опис варіанту використання для переходу з ресторанів на продукцію

Ім’я варіанту використання: Перегляд		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:		Перехід з ресторанна на продукцію.
Передумови:		Натиснути на ресторан .
Пост-умови:		
Відносини:		
	Асоціація:	Перейти на сторінку продукції.
	Включення:	
	Розширення:	
	Узагальнення:	
Основний потік:		Перегляд користувачем продуктів.
Альтернативні/виключні потоки:		

Таблиця 2.2

Опис варіанту використання для входу

Ім’я варіанту використання: Перегляд		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:		Вхід у систему.
Передумови:		Заповнення даних.
Пост-умови:		
Відносини:		
	Асоціація:	Виконати вхід .
	Включення:	Перевірити дані.
	Розширення:	
	Узагальнення:	
Основний потік:		Якщо дані правильні, то здійснюється вхід і користувач матиме змогу добавлять товари в корзину. Якщо припущена помилка то система повідомить про це.
Альтернативні/виключні потоки:		

Таблиця 2.3

Опис варіанту використання для пошуку продукції

Ім’я варіанту використання: Перегляд		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:		Пошук продуктів.
Передумови:		Вести початкові ініціали назви продукції.
Пост-умови:		
Відносини:		
	Асоціація:	Виведе ті продукції, які співпали з початковими ініціалами
	Включення:	
	Розширення:	
	Узагальнення:	
Основний потік:		Показує товари які співпали по назві
Альтернативні/виключні потоки:		

Таблиця 2.4

Опис варіанту використання для оформлення замовлення

Ім’я варіанту використання: Перегляд		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:		Зайти до корзини
Передумови:		Натиснути на кнопку оформлення замовлення
Пост-умови:		
Відносини:		
	Асоціація:	Виведеться повідомлення про рекламний маркетинг
	Включення:	
	Розширення:	
	Узагальнення:	
Основний потік:		При прийняті в рекламі перейде на посилання до міні гри.
Альтернативні/виключні потоки:		

Таблиця 2.5

Опис варіанту використання для рекламного маркетингу

Ім’я варіанту використання: Перегляд		Рівень важливості: Високий
Основний актор: Користувач		
Короткий опис:		При прийняті в рекламному маркетингу
Передумови:		Перейдете до гри
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	
	Розширення:	
	Узагальнення:	
Основний потік:		В грі потрібно управляти машиною. Вона рухається по дорозі без зупинки і нею можна управляти .
Альтернативні/виключні потоки:		

2.2. Алгоритм роботи системи

Алгоритм - це точно визначений набір інструкцій або правил, за якими виконується певне завдання або обчислення. Він складається з послідовності кроків, кожен з яких виконує певну операцію або перетворення даних. Алгоритми можуть бути використані для вирішення різноманітних задач, таких як сортування даних, пошук інформації, оптимізація процесів, шифрування та багато іншого.

У випадку роботи системи, алгоритм описує послідовність дій, які система виконує для обробки вхідних даних та надання відповіді або результату.

Діаграма активності - це графічний інструмент, який використовується для моделювання послідовності активностей або дій в процесі. Вона дозволяє

зобразити взаємодію різних об'єктів (людей, систем, компонентів програмного забезпечення тощо) і показати, як вони взаємодіють між собою.

Діаграми активності широко використовуються в галузі системного аналізу, проектування програмного забезпечення та управління процесами. Вони допомагають уявити послідовність дій, умови та рішення, що пов'язані з конкретним процесом.

У діаграмі активності активність представляється у вигляді прямокутників, а стрілки вказують послідовність виконання цих активностей. Також можуть використовуватися різні вузли для репрезентації умов, паралельних процесів, зворотних зв'язків та інших елементів, що розширюють можливості моделювання. [14]

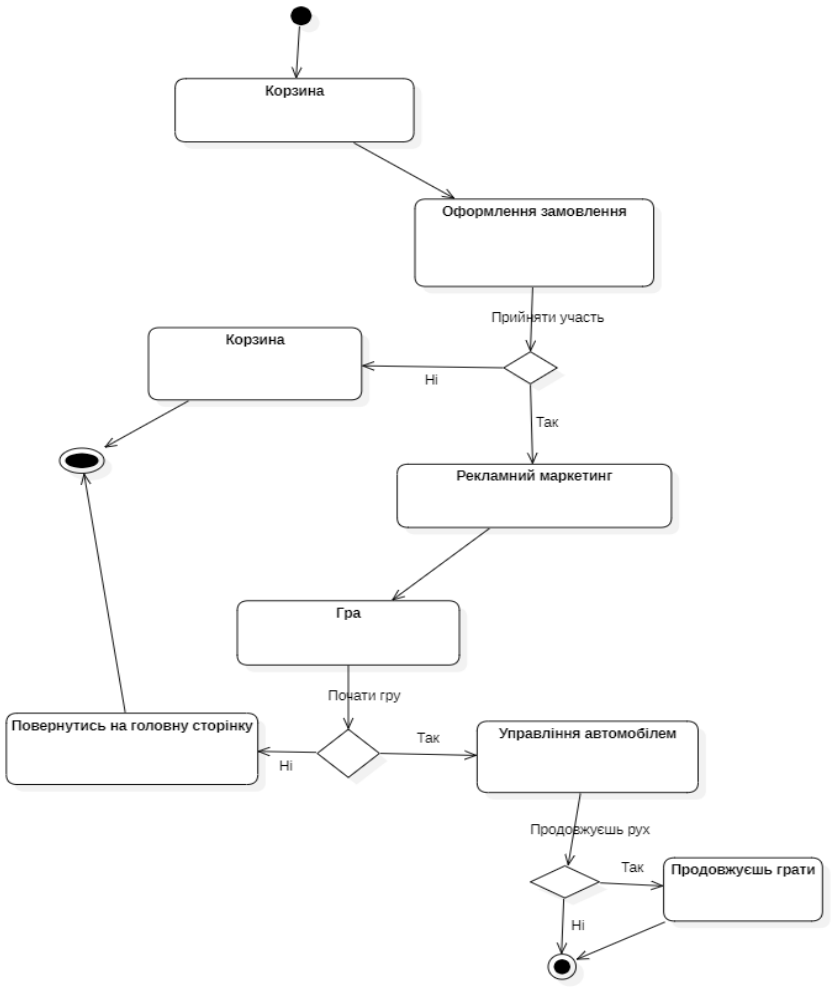

Рис. 2.2 Діаграма активності рекламного маркетингу

Отже, по діаграмі (рис.2.2) коли в корзині натискаєш на кнопку оформити замовлення виводиться на екран повідомлення в якому пропонують прийнять

участь в грі. Якщо відказатися то користувач повернеться до корзини, а якщо прийме участь то перейде до гри. В грі є дві кнопки початку гри та повернутися назад. При натисненні на почати гру на екрані з’явиться дорога на, якій є червоний автомобіль та декілька синіх. Червоним автомобілем управляє користувач і йому потрібно ухилиться від синіх автомобілів. Гра закінчиться тоді коли користувач вріжеться в синій автомобіль. Сама гра у вигляді 2D.

Заходим на головну сторінку (рис.2.3) переходимо на сторінку ресторанів. На сторінці ресторанів тиснем на кнопку «Увійти». Водимо необхідні данні і якщо все правильно ввели то заходим під користувача, а якщо ні то виведе, що не правильно ввели. Далі потрібно перейти на сторінку товарів і добавити товар в корзину. Потім переходимо до корзини і натискаємо оформити замовлення і після цього виведе повідомлення в прийнятті у грі.

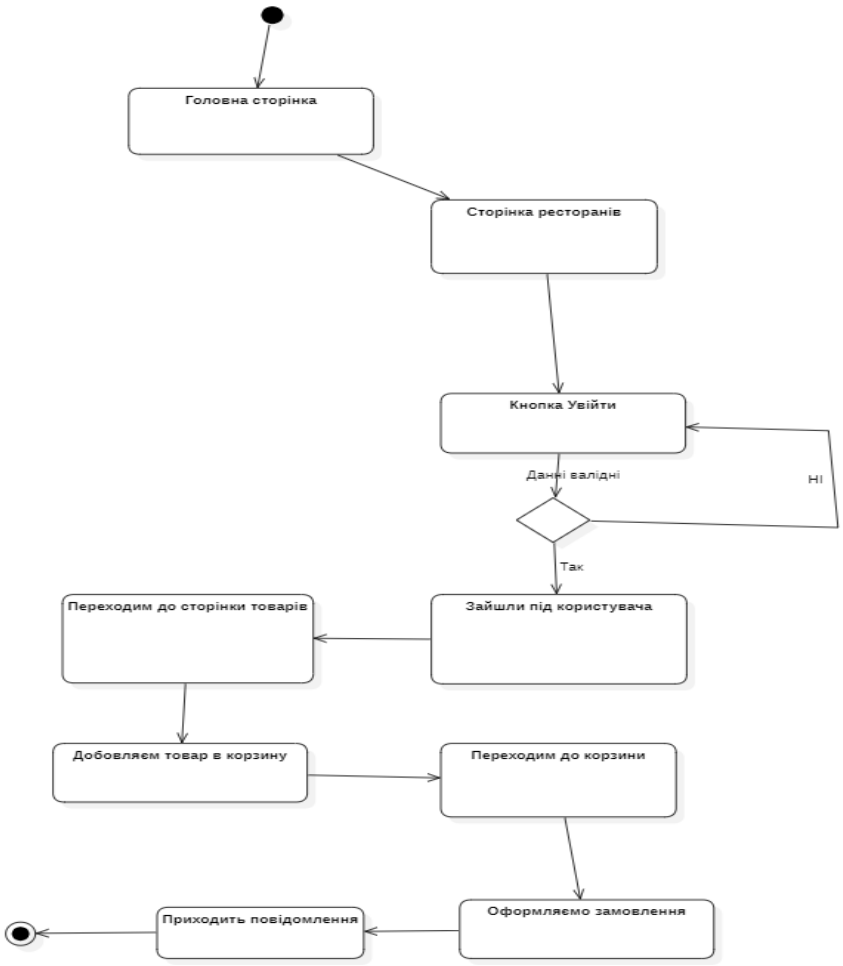

Рис. 2.3 – Діаграма активності оформлення замовлення користувачем

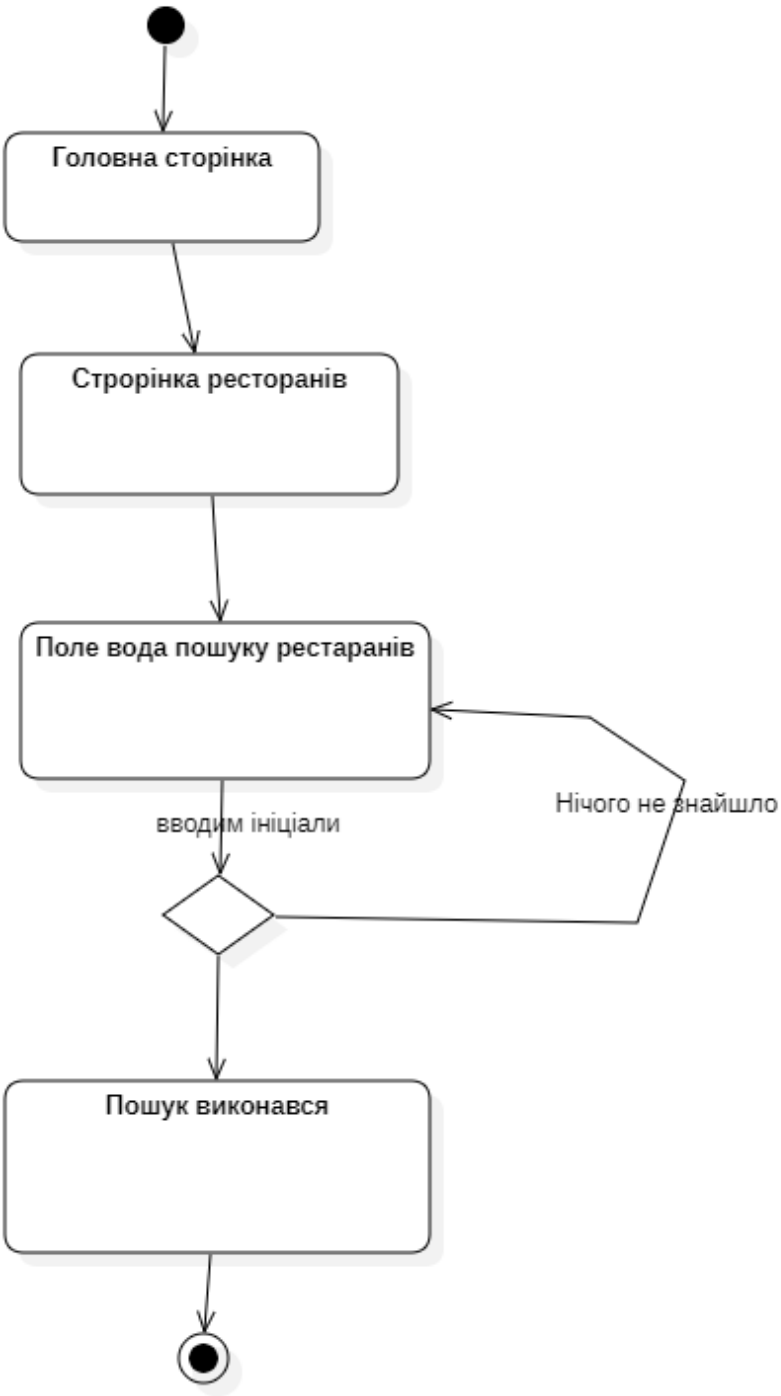

Рис. 2.4 – Діаграма активності пошуку

Заходим на головну сторінку , далі на сторінку ресторанів. Водимо ініціали і вже по першим виводи ті що сходяться з тими ініціалами, що вводить користувач. Якщо в полі вводах ініціали не сходяться з назвами ресторанна то нічого не находить

2.3 Структура компонентів системи

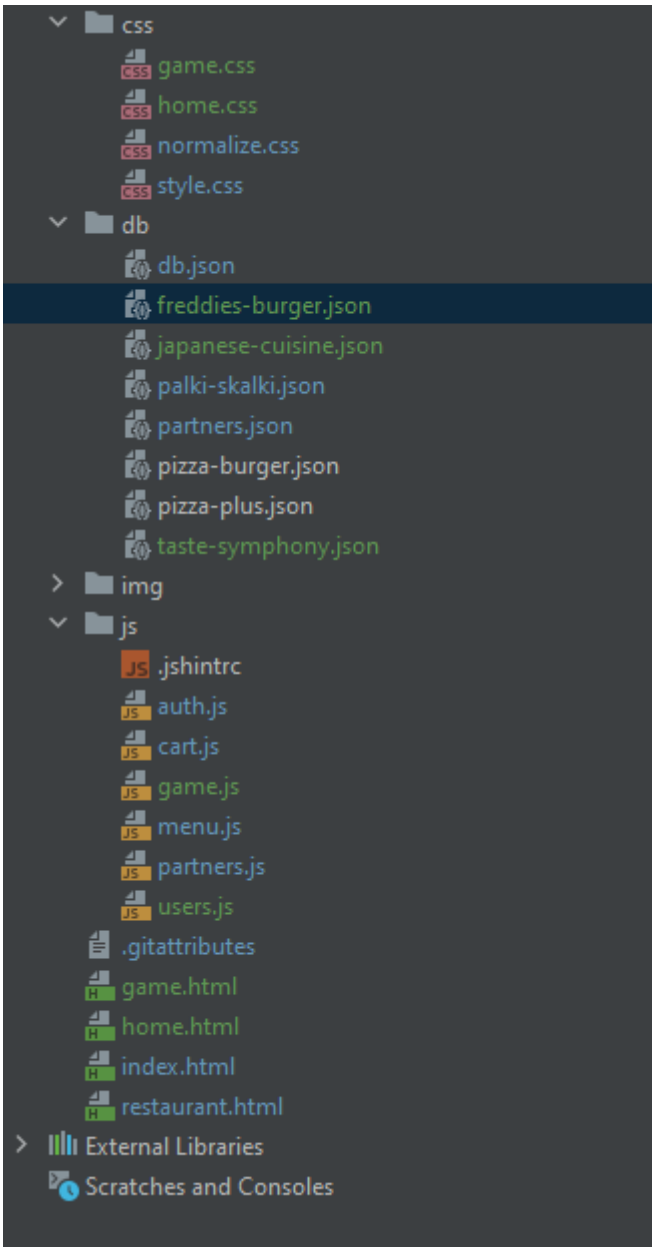

Рис. 2.5 - Загальна структура проекту

Один з важливих частин коду є компонент users.js , він дає змогу зберігати всіх користувачів, що входили данні в браузер до розділа «LocalStorage». Цей спосіб є не надійний. Краще використовувати серверверну технологію.

```
const sha256 = (input) => {
    return window.sha256(input);
};

const saveData = (email, phone, password) => {
    const storedData = JSON.parse(localStorage.getItem('storedData')) || [];
    storedData.push({
```

```
        id: new Date().getTime(),
        email,
        phone,
        hashedPassword: sha256(password)
    });
    localStorage.setItem('storedData', JSON.stringify(storedData));
};

const buttonLogin = document.querySelector(".button-login");

buttonLogin.addEventListener('click', () => {
    const inputEmail = document.getElementById("email").value;
    const inputPhone = document.getElementById("phone").value;
    const inputPassword = document.getElementById("password").value;

    saveData(inputEmail, inputPhone, inputPassword);
    console.log('Дані збережено');
});

const storedData = JSON.parse(localStorage.getItem('storedData'));
console.log(storedData);
```

Висновки до розділу 2

У другому розділі було визначено варіанти використання за діаграми варіантів використання. Основні дії з діаграми варіантів використання були описані в таблицях, а саме: перегляд даних, вхід , подача заявки та зміна заявки. Було розглянуто алгоритм системи і створено діаграми активності користувача.

В останньому пункті було показано основні компоненти системи і деяку реалізацію.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1. Архітектура додатку та його розгортання

Для розробки сайту використовується архітектуру простого статичного веб-сайту. Тобто сайт складається зі статичних HTML, CSS і JavaScript файлів, які відображають інформацію користувачам, але він не має серверної частини або здатності взаємодіяти з базою даних. Цей тип архітектури найчастіше використовується для простих веб-сайтів, де динамічність і інтерактивність не є основною потребою.

Оскільки ця архітектура не використовує серверної частини, відсутність бази даних та інших серверних технологій обмежує можливості сайту. Основна увага зосереджується на розробці статичних сторінок з використанням HTML, стилізації їх за допомогою CSS і можливої додаткової функціональності з використанням JavaScript.

Перевагою такого підходу є простота та швидкість розробки, відсутність необхідності в серверних ресурсах та інфраструктурі.

Діаграма розгортання - це вид діаграми, що використовується в моделюванні програмного забезпечення для візуалізації фізичної архітектури системи та розміщення її компонентів на фізичних або віртуальних пристроях.

Діаграма розгортання відображає фізичну організацію компонентів системи та зв'язки між ними. Вона дозволяє показати, які компоненти програмного забезпечення, обладнання або сервіси розгорнуті на конкретних пристроях (фізичних або віртуальних) та як вони взаємодіють між собою через мережу. На (рис. 3.1) зображено діаграма компонентів, яку хотілось використовувати в цьому проекті.

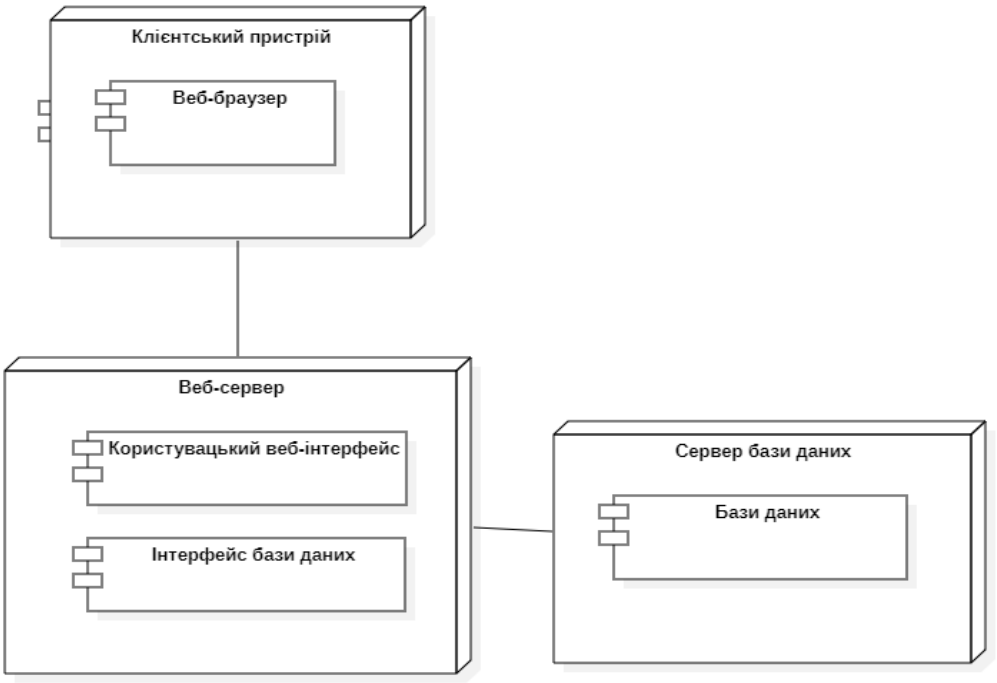

Рис. 3.1 - Діаграма компонентів

3.2. Інтерфейс та порядок роботи з системою

При завантаженні стартової сторінки користувач має можливість переглянути спеціальні пропозиції, що пропонує сайт з замовленню їжі та інформацію про нас.

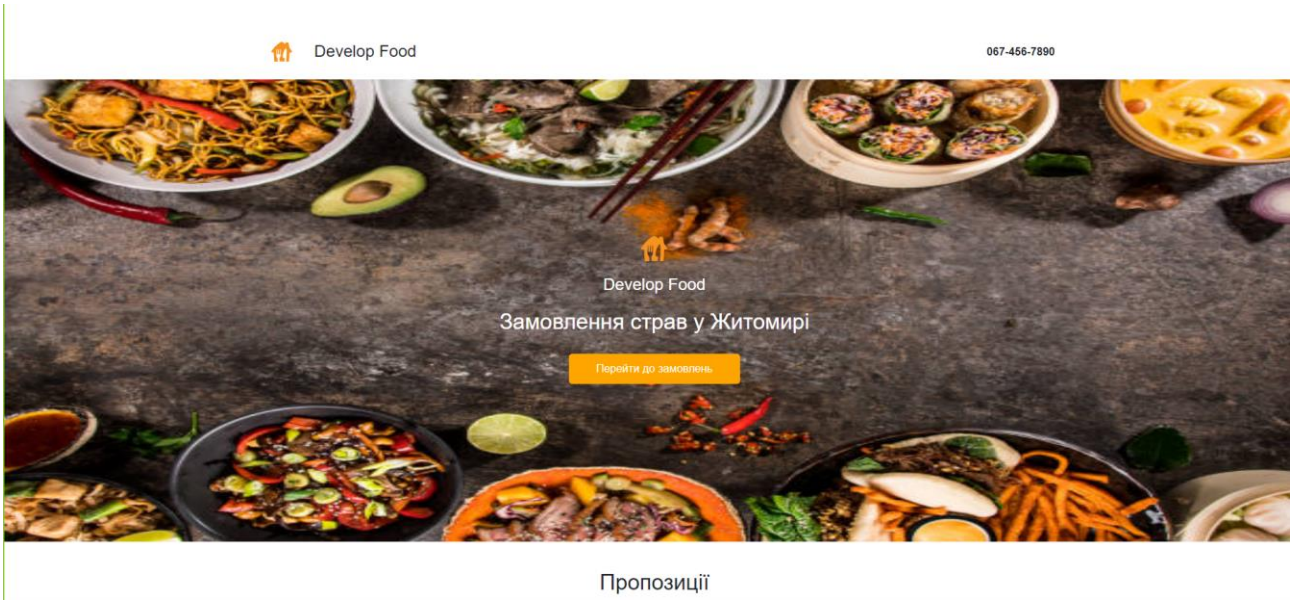

Рис.3.2 - Головна сторінка

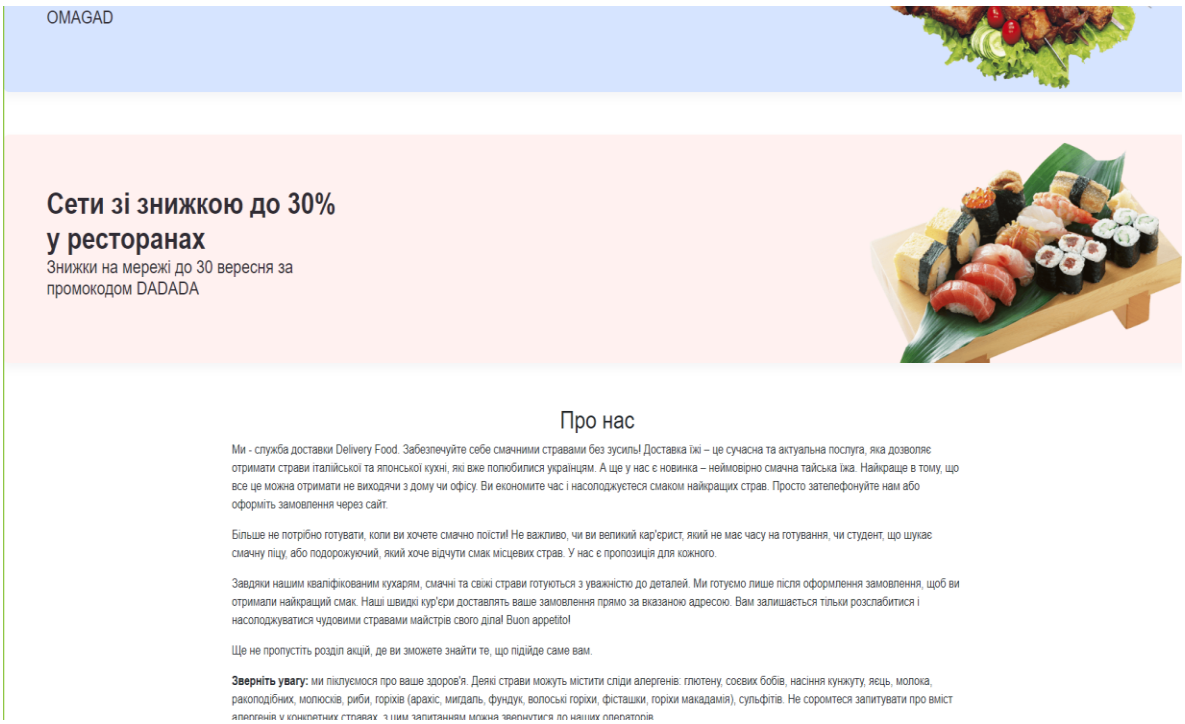

Рис. 3.3 – Спеціальні послуги та інформація по сайт

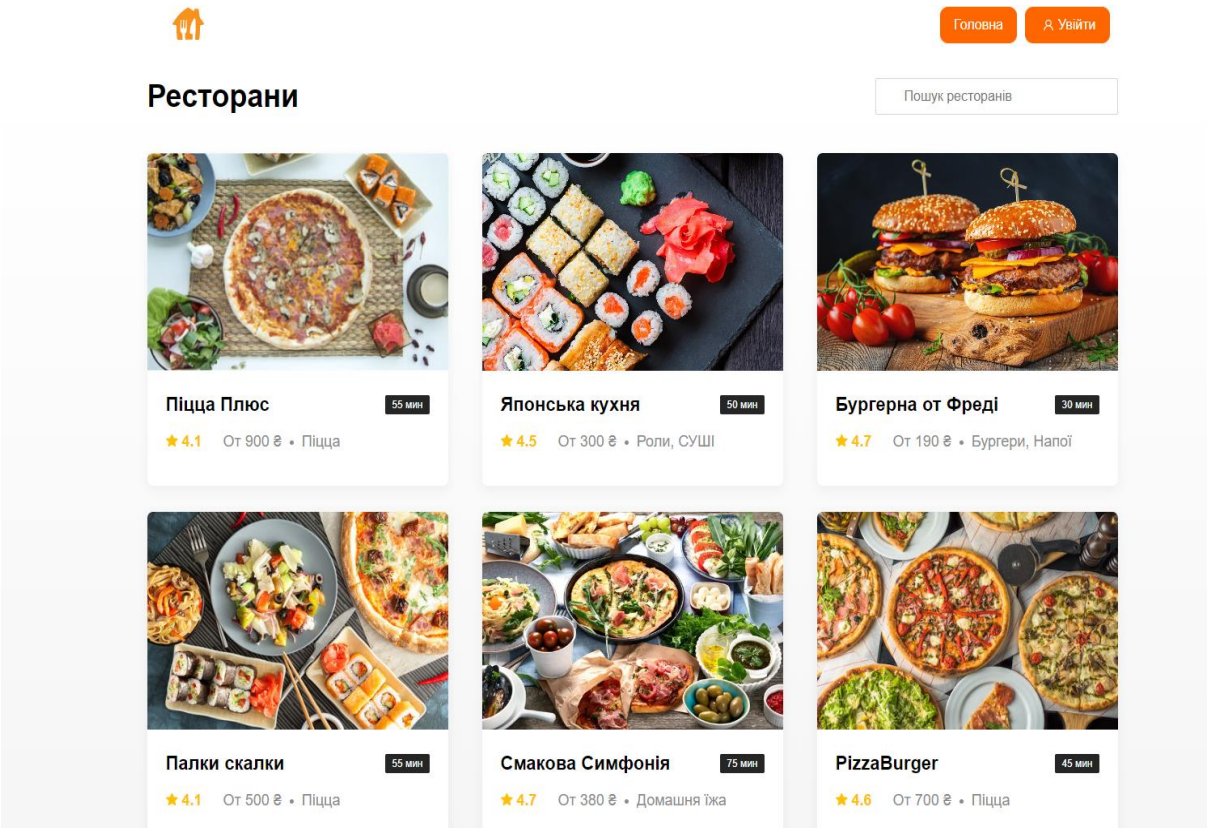

Рис. 3.4 – Сторінка з ресторанами

Далі натиснувши на кнопку Перейти до замовлення , перейдем на сторінку ресторанів.(рис. 3.4)

На сторінці ресторанна є дві кнопки, головна сторінка та кнопка Увійти.

При натисненні кнопка Увійти виведеться форма для заповнення, що зайти під коростувача.

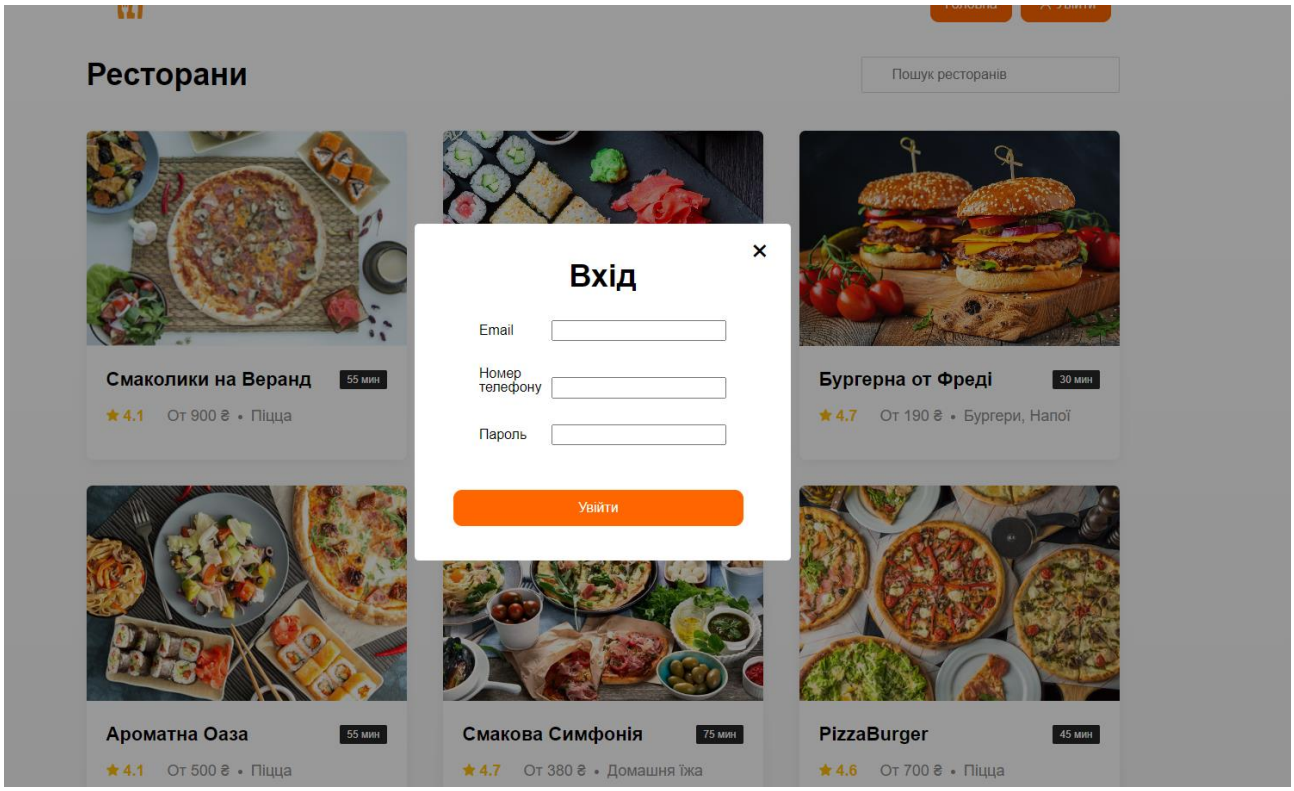

Рис. 3.5 – Виконання входу

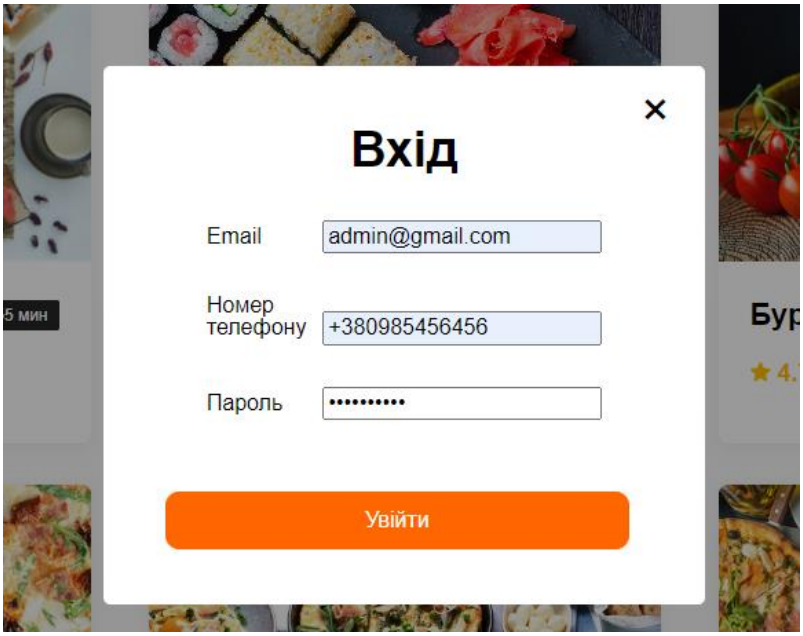

Рис. 3.6 - Виконання входу

Після того як ми зайшли з’явився кошик і кнопка вийти з під користувача.

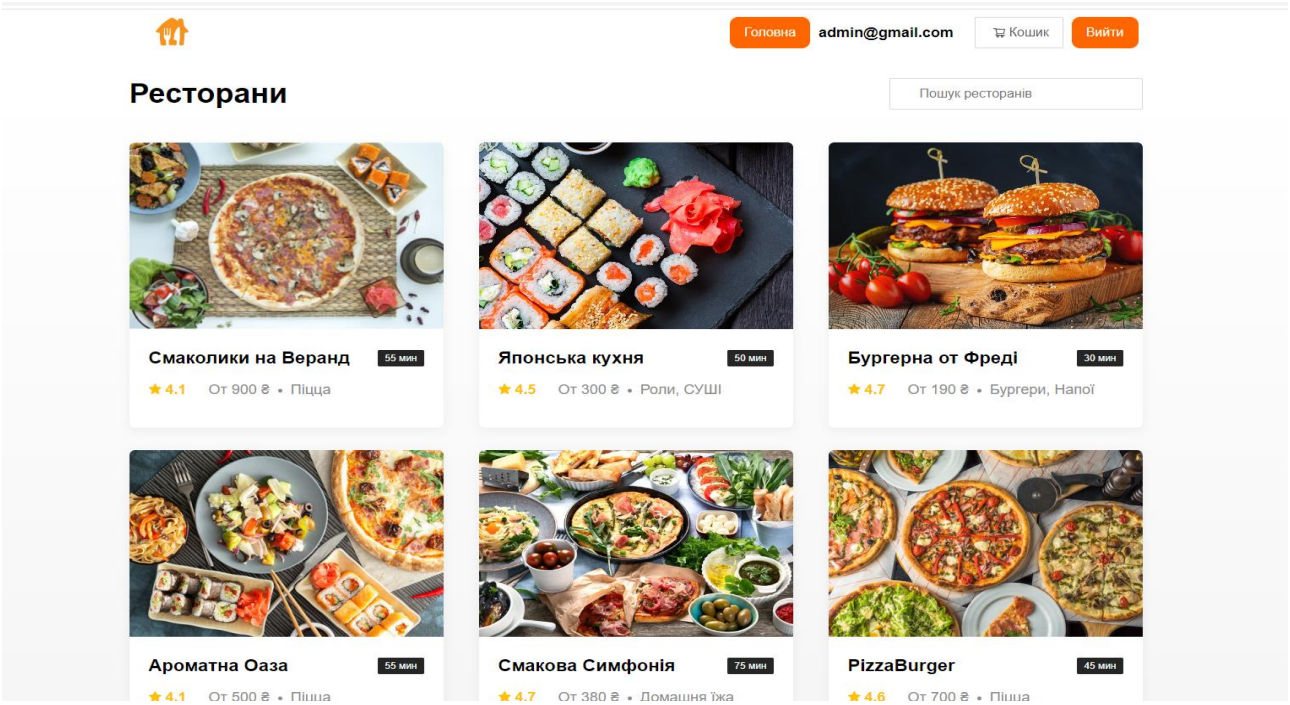

Рис. 3.7 - Вдалось зайти під користувача

Натиснувши на ресторан ми перейдемо до сторінки з товарами цього ресторану.

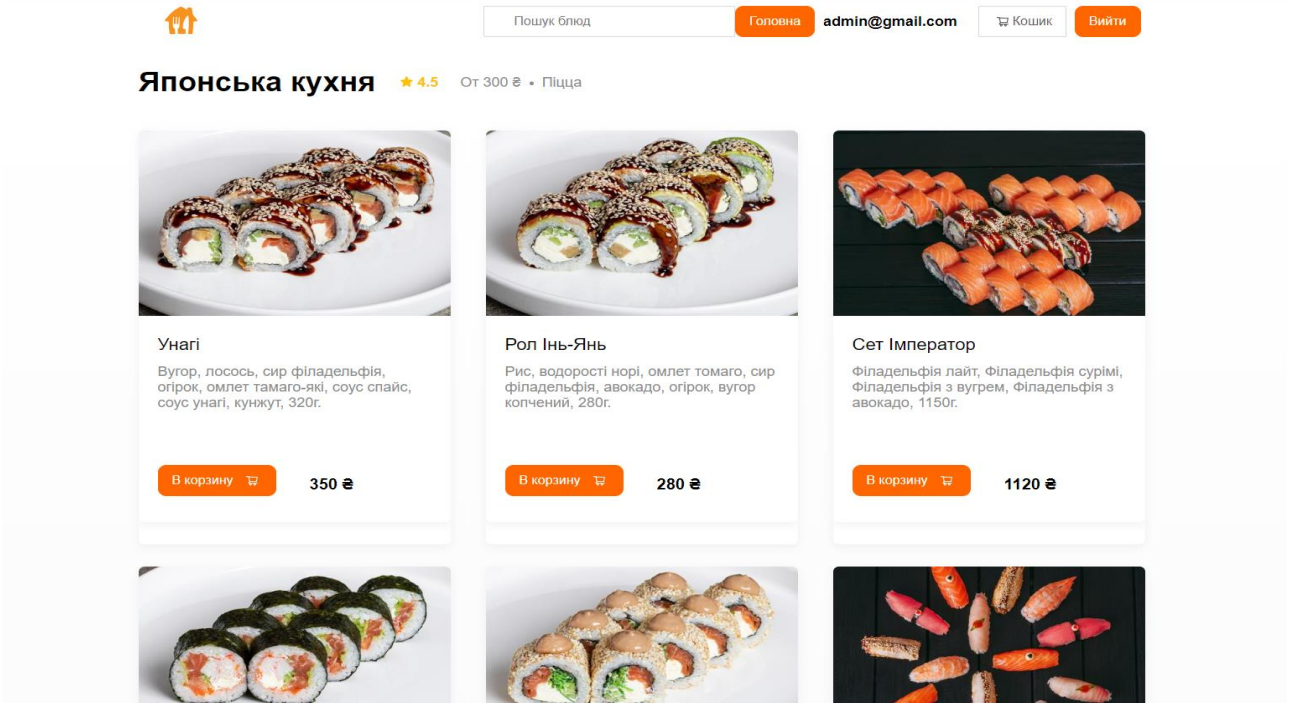

Рис. 3.8 - Сторінка з товарами Японської кухні

При на тиснені на кнопку в корзину цей товар добавиться в кошик .

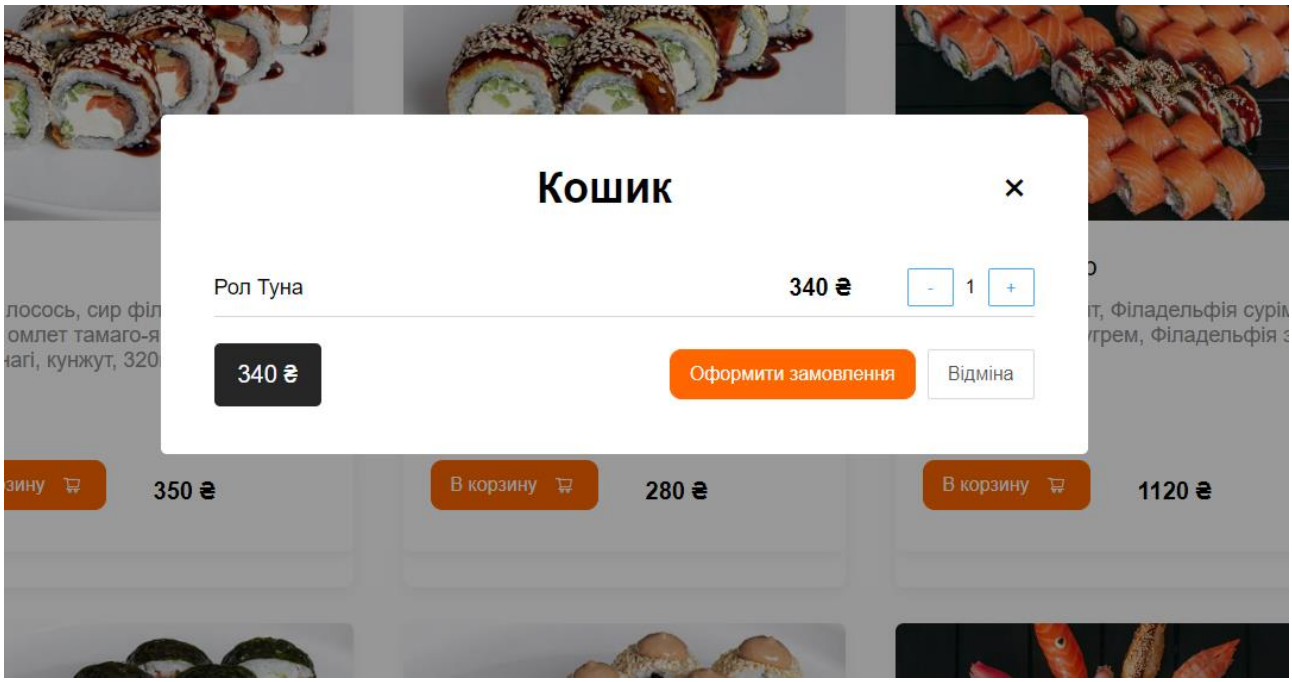

Рис. 3.9 – До кошика додався товар

При виконанні пошуку система шукає по першим або другім словам. На сторінці з ресторанами пошук робить так само.

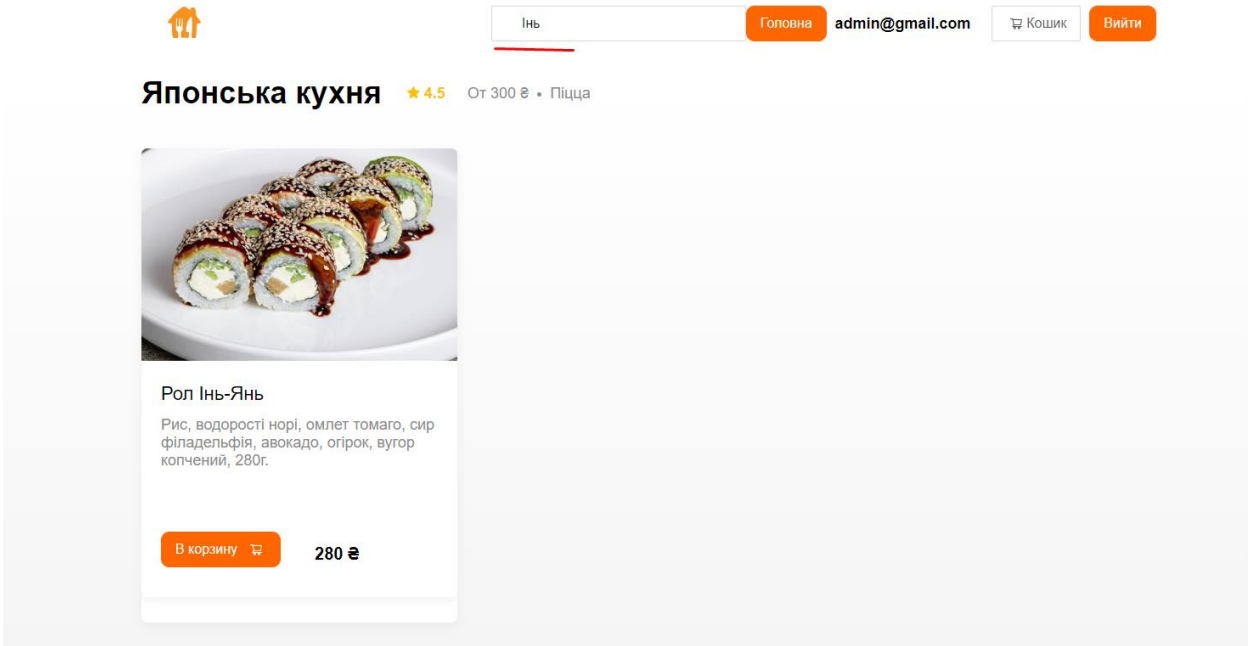

Рис. 3.10 - Виконання пошуку.

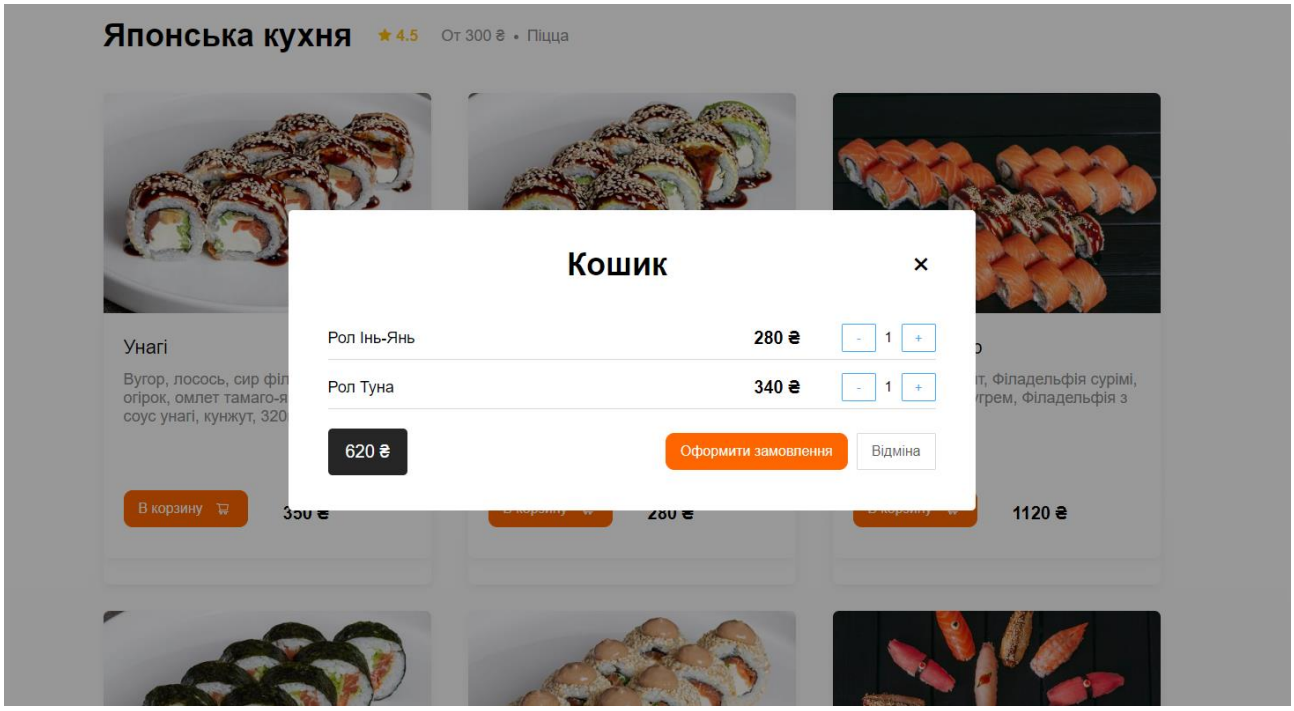

Рис. 3.11 – Робота з кошиком.

Додам до кошика, ще один товар. При використанні + та – в кошиму можна міняти кількість товару.

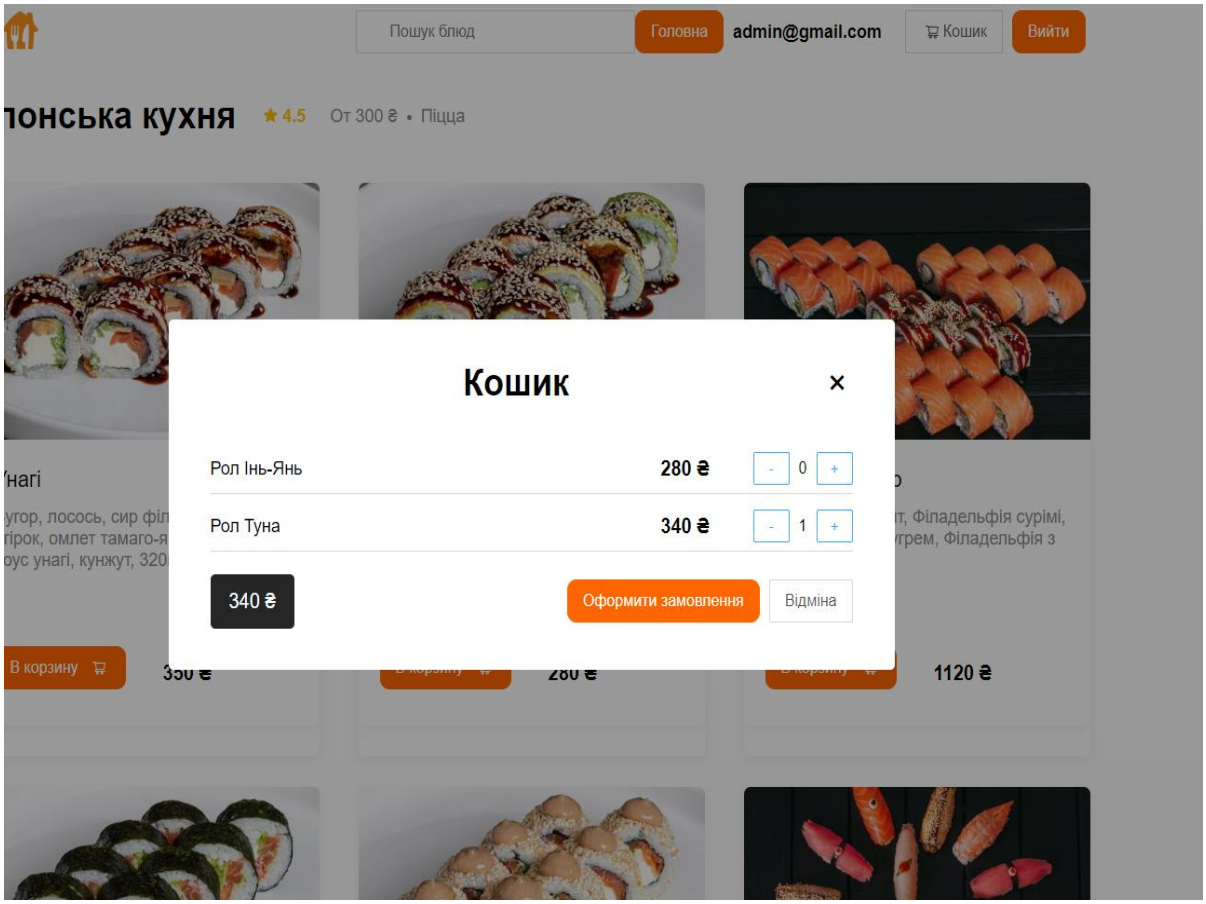

Рис. 3.12 – Робота з кошиком

При натисненні на кнопку оформити замовлення виведе повідомлення для у часті в грі.

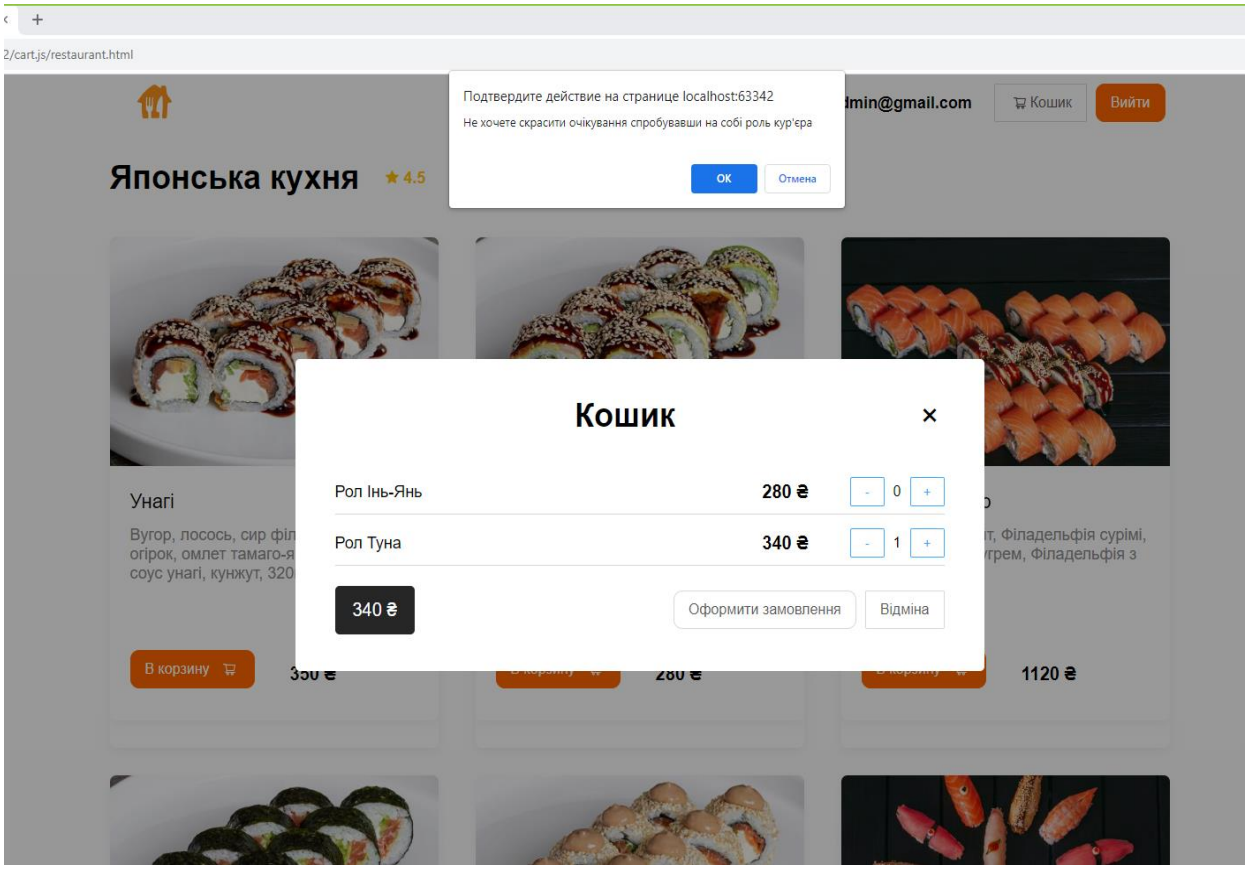

Рис. 3.12 – Робота з кошиком

При прийнятті в участі в грі користувач перейде на сторінку з грою.

Рис. 3.12 – Вигляд гри

В цій грі користувач управляє автомобілем стрілочками на клавіатурі і йому потрібно потрібно ухилятися від інших автомобілів.

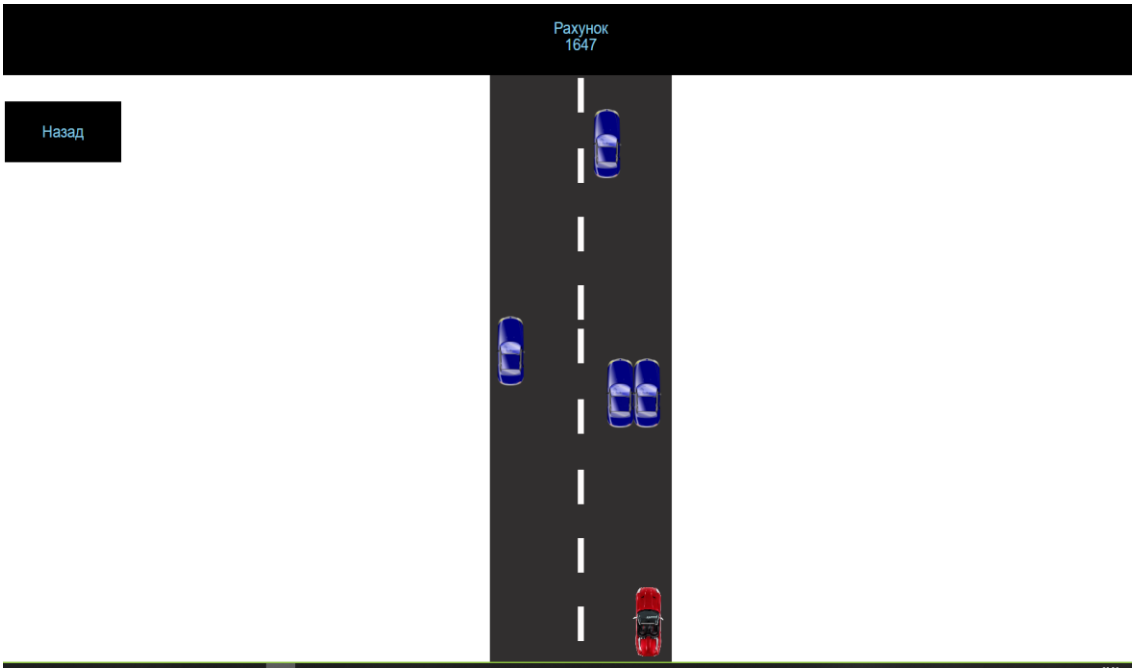

Рис. 3.13 Вигляд гри

При зіткнені гра зупиняється. Але можна почати знову.

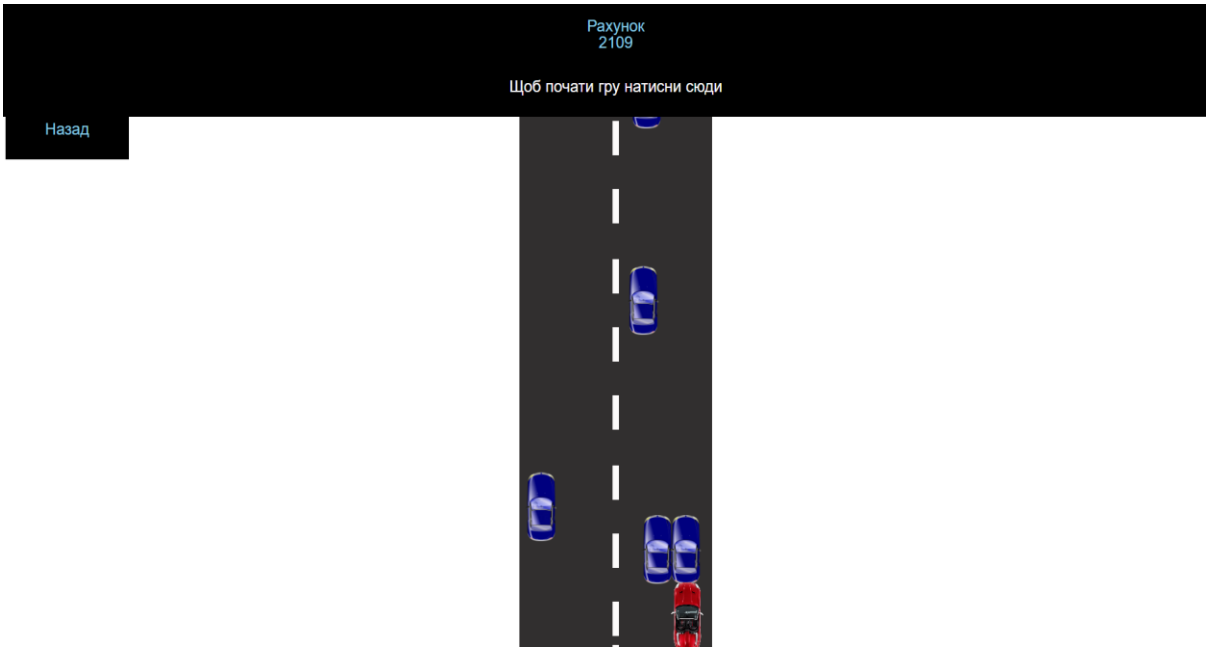

Рис. 3.14 Вигляд гри

1.3. Тестування роботи програмного комплексу

При виконанні входу, перевіряє правильність вводу даних і якшо не правильно введені дані повідомляє про це.

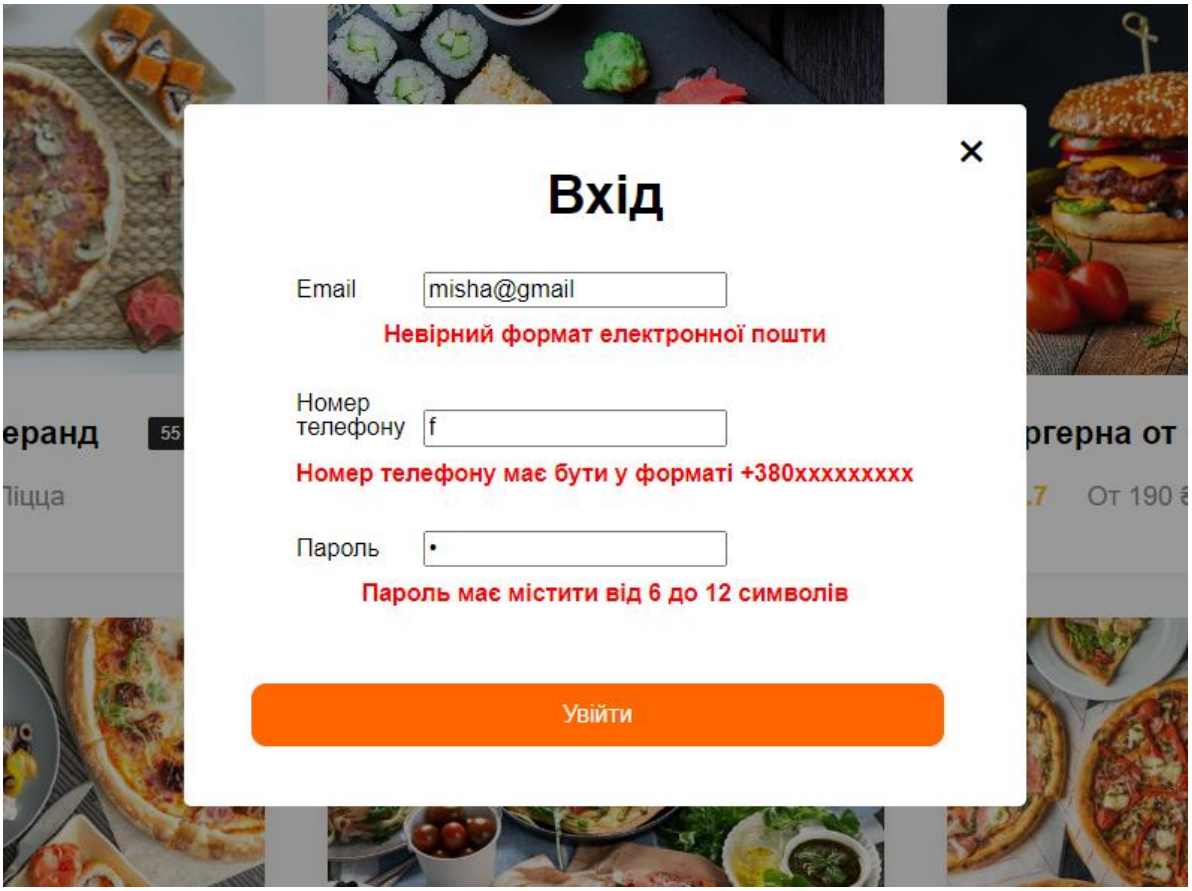

Рис. 3.15 – Форма вводу даних

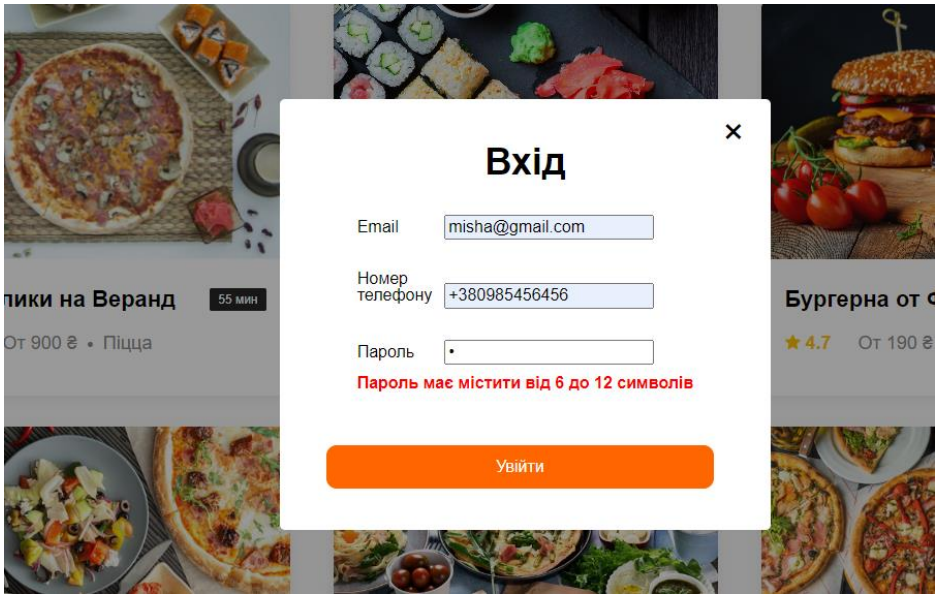

Рис. 3.16 – Форма вводу даних

При на тисканню на кнопку оформленні замовлення на сторінці з ресторанами та на сторінці з продуктами повідомлення про рекламний маркетинг працює.

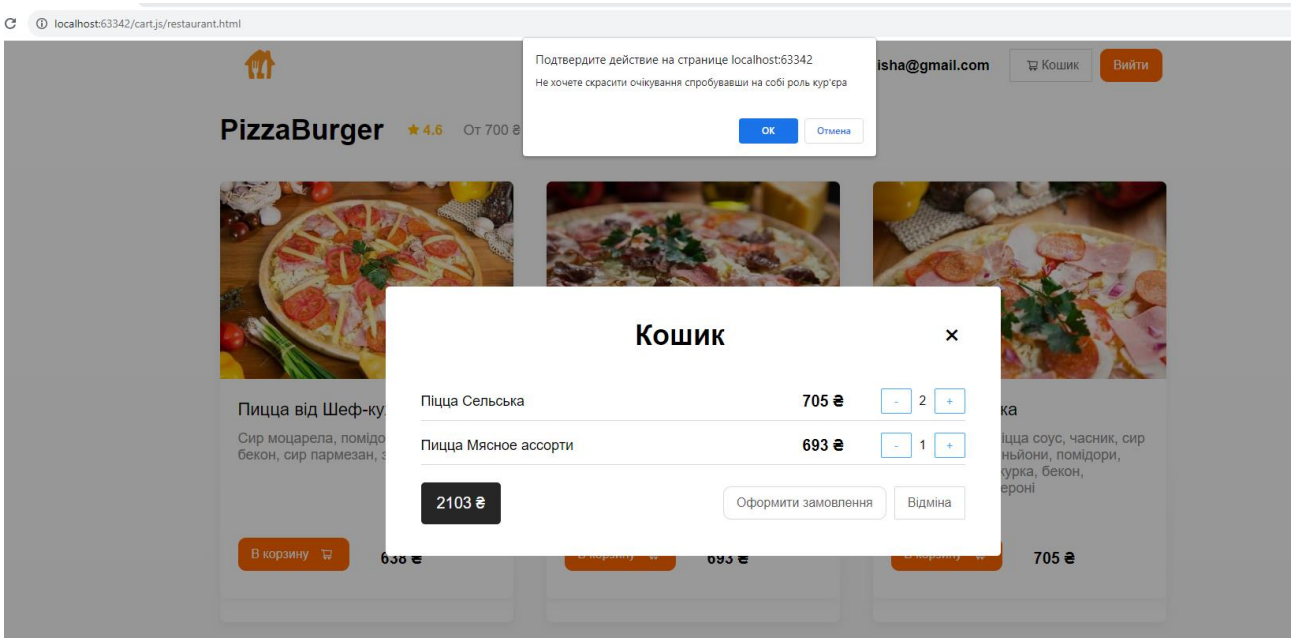

Рис. 3.17 – Перевірка рекламний маркетинга

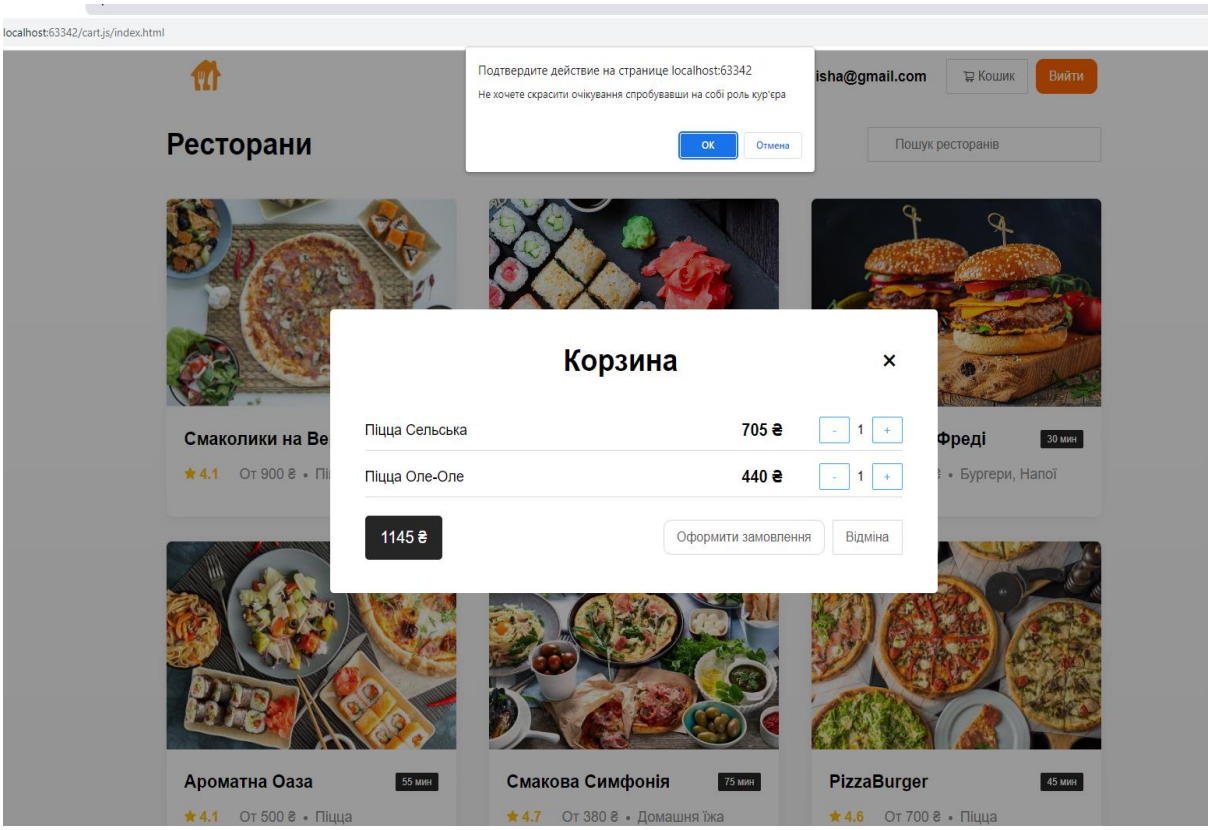

Рис. 3.18 – Перевірка рекламний маркетинга

Перевірив гру на помилки і при перевірці їх не було виявлено.

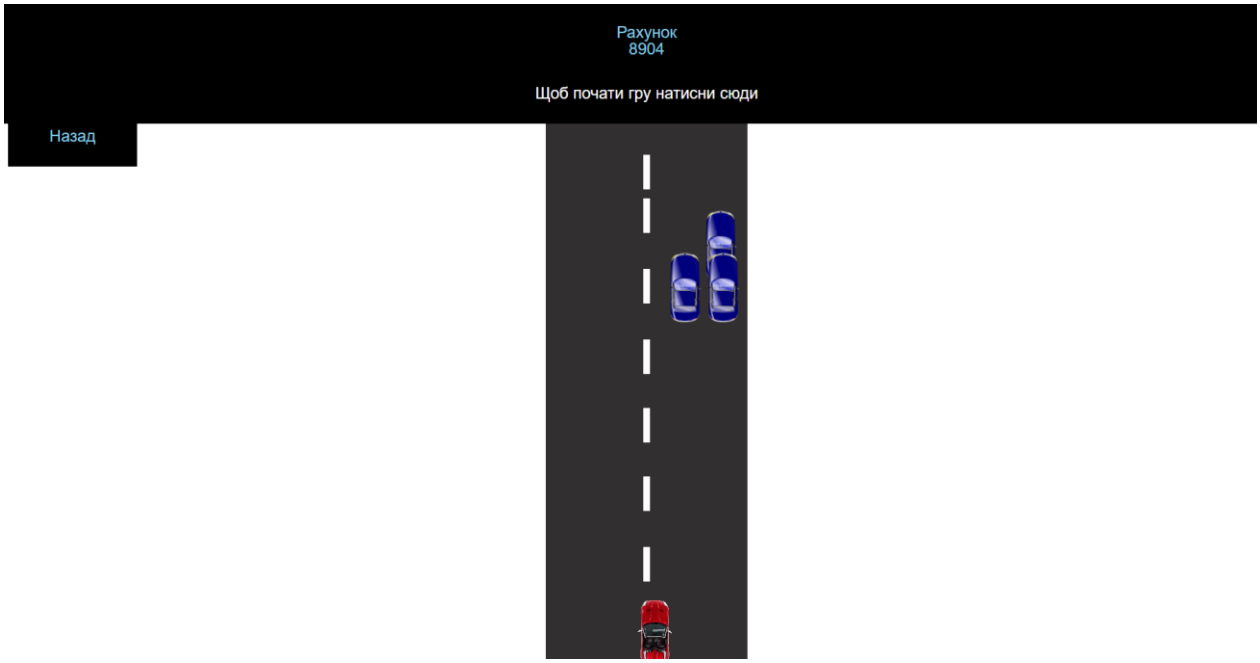

Рис. 3.19 – Перевірка гри на помилкі

Висновки до розділу 3

В цьому розділі була розроблена архітектура додатку та діаграми компонентів.

Було розглянуто та описано роботу веб-додатку. Переглянуто весь функціонал та його вміст.

Було проведено тестування веб-додатку. Виправленні незначні помилки. В цілому додаток придатний до використання і без проблем виконує мінімальний набір функцій для подібних додатків.

ВИСНОВКИ

В результаті виконання випускної роботи бакалавра було спроектовано та реалізовано «Веб-сайт “Delivery Food”. Під час розробки цього веб-проекту було вирішено такі завдання:

1. Поставлено завдання на роботу.

2. Розглянуто методологію сайтів з замовленням їжі та були проаналізовані їх аналоги і специфіка. Визначено набір функцій який необхідний для веб-сайту.

3. Було розглянуто типи різних архітектури, в якому була вибрана архітектуру простого статичного веб-сайту.

4. Обрано набір технології для розробки веб-додатку. Основною мову для написання сайту була js для динамічного коду з використанням стандартних мов HTML, CSS з використанням файлу json та бібліотеки js-sha256.

5. За допомогою діаграми використання було розглянуто сценарії використання веб-сайту та за допомогою діаграми активності був розглянути його алгоритм.

6. Показана структура компонентів системи та її основна реалізація роботи з поясненням наведеного часткового коду.

7. На завершеному етапі розробки кваліфікаційної роботи було описано принцип роботи з сайтом, а також проведено тестування роботи функцій веб-сайту.

Отже, у результаті розробки веб-додатку було виявлено мінус в якому не було розроблено особистий кабінет користувача, який би поліпшив роботу перевірки заявок та дав користувачеві візуальний план його оформлених та відправлених заявок. А так в основному інтерфейс простий для розуміння і виконує все що необхідно для веб-додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Створення сайту з замовленням їжі [Електронний ресурс] – Режим доступу до ресурсу: https://smartcafe.com.ua/uk/infocentr/site-dostavki
2. Сайт «Хочу З'їсти» [Електронний ресурс] – Режим доступу до ресурсу: https://hz-food.com.ua
3. Сайт «KILOGRAMM Sushi Project» [Електронний ресурс] – Режим доступу до ресурсу: https://kilogramm.com.ua
4. Сайт «Salateira» [Електронний ресурс] – Режим доступу до ресурсу: https://salateiramenu.com.ua
5. Архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://www.peerbits.com/blog/web-application-architecture.html
6. Архітектура мікросервіс [Електронний ресурс] – Режим доступу до ресурсу:https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/
7. Безсерверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://uk.theastrologypage.com/serverless-architecture
8. Трирівнева архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://jak.koshachek.com/articles/tririvneva-arhitektura-ce.html
9. Архітектура MVS [Електронний ресурс] – Режим доступу до ресурсу: https://docs.phalcon.io/4.0/en/mvc
10. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/what-is/javascript/
11. CSS з JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/CSS_and_JavaScript
12. JSON [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

13. Use Case Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://dut.edu.ua/ua/news-1-626-7758-zastosuvannya-uml-v-diplomnih-robotah_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy
14. Activity Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://jak.waykun.com/articles/diagrama-aktivnosti-dijalnosti-activity-diagram.html
15. Deployment Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/uml-deployment-diagram

ДОДАТКИ

Додаток А

Файл home.html

```
<!DOCTYPE html>
<html lang="ua">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Delivery Food</title>

  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="css/home.css"/>
  <link rel="stylesheet" href="css/style.css"/>
</head>
<body>

<div class="container">
  <div class="row align-items-center py-3">
    <div class="col-md-6">
      <div class="d-flex align-items-center">
        <img src="img/icon/logo.svg" alt="Logo" class="logo1 mr-2">
        <h1 class="m-0">Develop Food</h1>
      </div>
    </div>
    <div class="col-md-6 text-md-right">
      <p class="m-0"> <span id="phone-number">067-456-7890</span></p>
    </div>
  </div>
</div>

<div class="header">
  <img src="img/promo/home.jpg" alt="Header Image" class="header-img">
  <div class="center-content">
    <img src="img/icon/logo.svg" alt="Logo" class="logo">
    <h1 class="header-text">Develop Food</h1>
    <h2 class="header-text">Замовлення страв у Житомирі</h2>
    <a href="index.html" class="orange-button">Перейти до замовлень</a>
  </div>
 </div>

<h2 class="text-center">Пропозиції</h2>
<section class="container-promo">

  <section class="promo kebab">
    <h1 class="promo-title">Шашлики на травневі <br> зі знижкою 35%</h1>
    <p class="promo-text">
      Замовте шашлики в будь-якому ресторані до 10 травня та отримайте знижку за
промокодом OMAGAD
    </p>
  </section>
  <!-- <section class="promo vegetables">
     <h1 class="promo-title">Знижка 20% на всю їжу <br> за промокодом
LOVE.JS</h1>
     <p class="promo-text">
       Страву з ресторану привезуть разом із двома подарунковими книгами по
фронтенду
     </p>
   </section>
   -->
  <section class="promo sushi">
    <h1 class="promo-title">Сети зі знижкою до 30% <br> у ресторанах</h1>
```

```
    <p class="promo-text">
      Знижки на мережі до 30 вересня за промокодом DADADA
    </p>
  </section>
</section>

<h2 class="text-center">Про нас</h2>
<div class="container">
  <div class="row">
    <div class="col">
      <p class="intro-text">
        Ми - служба доставки <span class="brand-name">Delivery Food</span>.
Забезпечуйте себе смачними стравами без зусиль!
        Доставка їжі - це сучасна та актуальна послуга, яка дозволяє отримати
страви італійської та японської кухні,
        які вже полюбилися українцям. А ще у нас є новинка - неймовірно смачна
тайська їжа. Найкраще в тому, що все це
        можна отримати не виходячи з дому чи офісу. Ви економите час і
насолоджуєтеся смаком найкращих страв.
        Просто зателефонуйте нам або оформіть замовлення через сайт.
      </p>
      <p class="intro-text">
        Більше не потрібно готувати, коли ви хочете смачно поїсти! Не важливо,
чи ви великий кар'єрист, який не має
        часу на готування, чи студент, що шукає смачну піцу, або подорожуючий,
який хоче відчути смак місцевих страв.
        У нас є пропозиція для кожного.
      </p>
      <p class="intro-text">
        Завдяки нашим кваліфікованим кухарям, смачні та свіжі страви готуються з
уважністю до деталей. Ми готуємо
        лише після оформлення замовлення, щоб ви отримали найкращий смак. Наші
швидкі кур'єри доставлять ваше
        замовлення прямо за вказаною адресою. Вам залишається тільки
розслабитися і насолоджуватися чудовими стравами
        майстрів свого діла! Buon appetito!
      </p>
      <p class="intro-text">
        Ще не пропустіть розділ акцій, де ви зможете знайти те, що підійде саме
вам.
      </p>
      <p class="intro-text">
        <strong>Зверніть увагу:</strong> ми піклуємося про ваше здоров'я. Деякі
страви можуть містити сліди алергенів:
        глютену, соєвих бобів, насіння кунжуту, яєць, молока, ракоподібних,
молюсків, риби, горіхів (арахіс,
        мигдаль, фундук, волоські горіхи, фісташки, горіхи макадамія),
сульфітів. Не соромтеся запитувати
        про вміст алергенів у конкретних стравах, з цим запитанням можна
звернутися до наших операторів.
      </p>
    </div>
  </div>
</div>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></scri
pt>

</body>
</html>
```

Файл index.html

```
<!DOCTYPE html>
<html lang="ua">

<head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Delivery Food</title>

   <link rel="stylesheet" href="css/normalize.css" />
   <link rel="stylesheet" href="css/style.css" />
   <link rel="stylesheet" href="css/game.css"/>


</head>

<body>
   <div class="container">
      <header class="header">
         <a class="logo">
            <img src="img/icon/logo.svg" alt="Logo" />
         </a>

         <div class="buttons">
            <a href="home.html" class="button button-primary" style="text-
decoration: none;">
               <span class="button-text">Головна</span>
            </a>
            <span class="user-name"></span>
            <button class="button button-primary button-auth">
               <span class="button-auth-svg"></span>
               <span class="button-text">Увійти</span>
            </button>
            <button class="button button-cart" id="cart-button">
               <span class="button-cart-svg"></span>
               <span class="button-text">Кошик</span>
            </button>
            <button class="button button-primary button-out">
               <span class="button-text">Вийти</span>
               <span class="button-out-svg"></span>
            </button>
         </div>
      </header>
   </div>



   <main class="main">
      <div class="container">
         <section class="restaurants">
            <div class="section-heading">
               <h2 class="section-title">Ресторани</h2>
               <label class="search">
                  <input type="text" class="input input-search"
onkeyup="searchFood()" placeholder="Пошук ресторанів" id="searchbar" />
               </label>
            </div>
            <div class="cards cards-restaurants">

            </div>
```

```
         </section>
      </div>

   </main>

   <footer class="footer">
      <div class="container">
         <div class="footer-block">
            <img src="img/icon/logo.svg" alt="logo" class="logo footer-logo" />
         </div>

      </div>
   </footer>

   <div class="modal modal-cart">
      <div class="modal-dialog">
         <div class="modal-header">
            <h3 class="modal-title">Корзина</h3>
            <button class="close">&times;</button>
         </div>

         <div class="modal-body">
         </div>

         <div class="modal-footer">
            <span class="modal-pricetag">0 ₴</span>
            <div class="footer-buttons">
               <button class="button button-primary"
onclick="tryGame()">Оформити замовлення</button>
               <button class="button clear-cart">Відміна</button>
            </div>
         </div>

      </div>

   </div>

   <div class="modal-auth">
      <div class="modal-dialog modal-dialog-auth">
         <button class="close-auth">&times;</button>
         <form id="logInForm">
            <fieldset class="modal-body">
               <legend class="modal-title">Вхід</legend>
               <label class="label-auth">
                  <span>Email</span>
                  <input required id="email" type="email">
                  <div class="error-message" id="email-error"></div>
               </label>
               <label class="label-auth">
                  <span>Номер телефону</span>
                  <input required id="phone" type="tel" pattern="^\+38\d{10}$"
title="Введіть номер у форматі +380xxxxxxxxx">
                  <div class="error-message" id="phone-error"></div>
               </label>
               <label class="label-auth">
                  <span>Пароль</span>
                  <input id="password" type="password" minlength="6"
maxlength="12">
                  <div class="error-message" id="password-error"></div>
               </label>
            </fieldset>

            <div class="">
               <div class="footer-buttons">
```

```
                    <button class="button button-primary button-login"
type="submit" disabled>Увійти</button>
                </div>
            </div>
        </form>
    </div>
</div>
<script src="./js/auth.js"></script>
<script src="./js/partners.js"></script>
<script src="./js/cart.js"></script>
<script src="./js/menu.js"></script>
<script src="./js/game.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/js-
sha256/0.9.0/sha256.min.js"></script> <!--бібліотека js-sha256 для шифрування
пароля -->
<script src="./js/users.js"></script>


</body>

</html>
```

Файл menu.js

```
const menu = () => {
const cardsMenu = document.querySelector('.cards-menu');

const changeTitle = (restaurant) => {
    const restaurantTitle = document.querySelector('.restaurant-title');
    const price = document.querySelector('.price');
    const rating = document.querySelector('.rating');
    restaurantTitle.textContent = restaurant.name;
    price.textContent = `От ${restaurant.price} ₴`;
    rating.textContent = restaurant.stars;
};

const cartArray = localStorage.getItem('cart') ?
 JSON.parse(localStorage.getItem('cart')): [];

const addToCart = (cartItem) => {
    if (cartArray.some((item) => item.id === cartItem.id)) {
        cartArray.map(item => {
            if (item.id === cartItem.id){
                item.count++;
            }
            return item;
        });
    } else {
        cartArray.push(cartItem);
    }
    localStorage.setItem('cart', JSON.stringify(cartArray));
};

const renderIrems = (data) => {
    data.forEach(({description, id, image, name, price}) => {
        const card = document.createElement('div');
        card.classList.add('card');
        card.innerHTML = `
            <div class="card">
                <img src=${image} alt=${name} class="card-image" />
                <div class="card-text">
```

```
                    <div class="card-heading">
                        <h3 class="card-title card-title-reg">${name}</h3>
                    </div>
                    <div class="card-info">
                        <div class="ingredients">${description}
                        </div>
                    </div>
                    <div class="card-buttons">
                        <button class="button button-primary button-add-cart">
                            <span class="button-card-text">В корзину</span>
                            <span class="button-cart-svg"></span>
                        </button>
                        <strong class="card-price-bold">${price} ₴</strong>
                    </div>
                </div>
            </div>`;
        card.querySelector('.button-add-cart').addEventListener('click', () => {
            addToCart({ name, price, id, count: 1 });
        });
        cardsMenu.appendChild(card);
    });
};

if (localStorage.getItem('restaurant')) {
    const restaurant = JSON.parse(localStorage.getItem('restaurant'));
    changeTitle(restaurant);

    fetch(`../db/${restaurant.products}`)
    .then((response) => response.json())
    .then((data) => renderIrems(data))
    .catch((error) => {
        console.log("Щось, пішло не так");
    });
} else {
    window.location.href = "index.html";
}
};
menu();

const slider = document.querySelector('.slider');
const prevArrow = document.querySelector('.prev-arrow');
const nextArrow = document.querySelector('.next-arrow');
const slideWidth = document.querySelector('.slide').clientWidth;

let currentIndex = 0;

prevArrow.addEventListener('click', () => {
    currentIndex = (currentIndex - 1 + slider.children.length) %
slider.children.length;
    updateSliderPosition();
});

nextArrow.addEventListener('click', () => {
    currentIndex = (currentIndex + 1) % slider.children.length;
    updateSliderPosition();
});

function updateSliderPosition() {
    slider.style.transform = `translateX(-${currentIndex * slideWidth}px)`;
}

updateSliderPosition();
```

```
function search_food() {
    let input = document.getElementById('searchbar').value;
    input=input.toLowerCase();
    let x = document.getElementsByClassName('card');

    for (i = 0; i < x.length; i++) {
        if (!x[i].innerHTML.toLowerCase().includes(input)) {
            x[i].style.display="none";
        }
        else {
            x[i].style.display="list-item";
        }
    }
}


function tryGame() {
    if (confirm(`Не хочете скрасити очікування спробувавши на собі роль
кур'єра`)) {
        // Save it!
        window.location = "game.html";
    } else {
        // Do nothing!
    }
}
```

Файл partners.js

```
const partners = () => {
const cardsRestaurants = document.querySelector('.cards-restaurants');

const renderIrems = (data) => {
    data.forEach((item) => {
        const {image, name, time_of_delivery, stars, price, kitchen, products} =
item;
        const a = document.createElement('a');
        a.setAttribute('href', 'restaurant.html');
        a.classList.add('card');
        a.classList.add('card-restaurant');
        a.dataset.products = products;
        a.innerHTML = `
            <img src=${image} alt="${name}" class="card-image" />
            <div class="card-text">
                <div class="card-heading">
                    <h3 class="card-title">${name}</h3>
                    <span class="card-tag tag">${time_of_delivery} мин</span>
                </div>
                <div class="card-info">
                    <div class="rating">
                        ${stars}
                    </div>
                    <div class="price">От ${price} ₴</div>
                    <div class="category">${kitchen}</div>
                </div>
            </div>` ;

        a.addEventListener('click', (e) => {
            e.preventDefault();
            localStorage.setItem('restaurant', JSON.stringify(item));

            window.location.href = 'restaurant.html';
        });

        cardsRestaurants.appendChild(a);
```

```
    });
};


fetch(`../db/partners.json`)
    .then((response) => response.json())
    .then((data) => renderIrems(data))
    .catch((error) => {
        console.log("Щось пішло не так, помилка");
    });
};
partners();
```

Файл game.js

```
const score = document.querySelector('.score'),
   start = document.querySelector('.start'),
   gameArea = document.querySelector('.gamearea'),
   car = document.createElement('div');
car.classList.add('car');

const keys = {
   ArrowUp: false,
   ArrowDown: false,
   ArrowRight: false,
   ArrowLeft: false
};

const setting = {
   start: false,
   score: 0,
   speed: 3,
   traffic: 3
};

function getQuantityElements(heightElement) {
   return document.documentElement.clientHeight / heightElement + 1;
}

function startGame() {
   start.classList.add('hide');
   gameArea.innerHTML = '';



   for (let i = 0; i < getQuantityElements(100); i++){
      const line = document.createElement('div');
      line.classList.add('line');
      line.style.top = (i * 100) + 'px';
      line.y = i * 100;
      gameArea.appendChild(line);
   }

   for (let i = 0; i < getQuantityElements(100 * setting.traffic); i++){
      const enemy = document.createElement('div');
      enemy.classList.add('enemy');
      enemy.y = 100 * setting.traffic * (i + 1);
      enemy.style.left = Math.floor((Math.random() * (gameArea.offsetWidth -
50))) + 'px';
      enemy.style.top = enemy.y + 'px';
      enemy.style.background = 'transparent url(./img/game/enemy2.png) center /
cover no-repeat';
      gameArea.appendChild(enemy);
```

```
    }
    setting.score = 0;
    setting.start = true;
    gameArea.appendChild(car);
    car.style.left = gameArea.offsetWidth/2 - car.offsetWidth/2;
    car.style.top = 'auto';
    car.style.bottom = '10px';
    setting.x = car.offsetLeft;
    setting.y = car.offsetTop;
    requestAnimationFrame(playGame);
}

function playGame() {
    moveRoad();
    moveEnemy();
    if (setting.start) {
        setting.score += setting.speed;
        score.innerHTML = 'Рахунок<br>' + setting.score;
        if (keys.ArrowLeft && setting.x > 0) {
            setting.x -= setting.speed;
        }
        if (keys.ArrowRight && setting.x < (gameArea.offsetWidth -
car.offsetWidth)) {
            setting.x += setting.speed;
        }
        if (keys.ArrowDown && setting.y < (gameArea.offsetHeight -
car.offsetHeight)) {
            setting.y += setting.speed;
        }
        if (keys.ArrowUp && setting.y > 0) {
            setting.y -= setting.speed;
        }
        car.style.left = setting.x + 'px';
        car.style.top = setting.y + 'px';
        requestAnimationFrame(playGame);
    }
}

function startRun(event) {
    event.preventDefault();
    keys[event.key] = true;
}

function stopRun(event) {
    event.preventDefault();
    keys[event.key] = false;
}
function moveRoad() {
    let lines = document.querySelectorAll('.line');
    lines.forEach(function (line) {
        line.y += setting.speed;
        line.style.top = line.y + 'px';

        if (line.y > document.documentElement.clientHeight) {
            line.y = -100;
        }
    });
}

function moveEnemy() {
    let enemy = document.querySelectorAll('.enemy');
    enemy.forEach(function (item) {
        let carRect = car.getBoundingClientRect();
```

```
      let enemyRect = item.getBoundingClientRect();

      if (carRect.top <= enemyRect.bottom &&
         carRect.right >= enemyRect.left &&
         carRect.left <= enemyRect.right &&
         carRect.bottom >= enemyRect.top) {
            setting.start = false;
            start.classList.remove('hide');
            start.style.top = start.offsetHeight;
      }
      item.y += setting.speed/2;
      item.style.top = item.y + 'px';
      if (item.y >= document.documentElement.clientHeight) {
         item.y = -100 * setting.traffic;
         item.style.left = Math.floor((Math.random() * (gameArea.offsetWidth -
50))) + 'px';
      }
   });
}

start.addEventListener('click', startGame);
document.addEventListener('keydown', startRun);
document.addEventListener('keyup', stopRun);
```

Файл auth.js

```
const auth = () => {
    const buttonAuth = document.querySelector(".button-auth");
    const buttonOut = document.querySelector(".button-out");
    const modalAuth = document.querySelector(".modal-auth");
    const userName = document.querySelector('.user-name');
    const closeAuth = document.querySelector(".close-auth");
    const logInForm = document.getElementById("logInForm");
    const inputEmail = document.getElementById("email");
    const inputPhone = document.getElementById("phone");
    const inputPassword = document.getElementById("password");
    const buttonLogin = document.querySelector(".button-login");
    const buttonCart = document.querySelector('.button-cart');


    const validateForm = () => {
        const isValidEmail = inputEmail.checkValidity();
        const isValidPhone = inputPhone.checkValidity();
        const isValidPassword = inputPassword.checkValidity();

        buttonLogin.disabled = !(isValidEmail && isValidPhone &&
isValidPassword);
    };

    inputEmail.addEventListener('input', validateForm);
    inputPhone.addEventListener('input', validateForm);
    inputPassword.addEventListener('input', validateForm);

    const email = (user) => {
        buttonAuth.style.display = 'none';
        buttonOut.style.display = 'flex';
        userName.style.display = 'flex';
        buttonCart.style.display = 'flex';
        userName.textContent = user.email;
        modalAuth.style.display = "none";
    };
```

```
    const logout = () => {
        buttonAuth.style.display = 'flex';
        buttonOut.style.display = 'none';
        userName.style.display = 'none';
        userName.textContent = "";
        buttonCart.style.display = 'none';

        localStorage.removeItem('user');
    };

    buttonAuth.addEventListener('click', () => {
        modalAuth.style.display = "flex";
    });

    buttonOut.addEventListener('click', () => {
        logout();
    });

    closeAuth.addEventListener('click', () => {
        modalAuth.style.display = 'none';
    });

    logInForm.addEventListener('submit', (e) => {
        e.preventDefault();
        const user = {
            email: inputEmail.value,
            phone: inputPhone.value,
            password: inputPassword.value
        };

        localStorage.setItem('user', JSON.stringify(user));
        email(user);
    });

    if (localStorage.getItem('user')) {
        email(JSON.parse(localStorage.getItem('user')));
    }
};

auth();

const inputEmail = document.getElementById("email");
const inputPhone = document.getElementById("phone");
const inputPassword = document.getElementById("password");

const showError = (input, message) => {
    const errorElement = document.createElement('div');
    errorElement.className = 'error-message';
    errorElement.textContent = message;
    input.parentElement.appendChild(errorElement);
};

const clearError = (input) => {
    const errorElement = input.parentElement.querySelector('.error-message');
    if (errorElement) {
        errorElement.remove();
    }
};

const validateInput = (input, condition, errorMessage) => {
    clearError(input);
    if (!condition(input.value)) {
        showError(input, errorMessage);
        return false;
```

```
    }
    return true;
};

const validateEmail = (email) => /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

const validatePhone = (phone) => /^\+38\d{10}$/.test(phone);

const validatePassword = (password) => password.length >= 6 && password.length
<= 12;

const validateFormOnSubmit = (e) => {
    e.preventDefault();

    const isValidEmail = validateInput(inputEmail, validateEmail, 'Невірний
формат електронної пошти');
    const isValidPhone = validateInput(inputPhone, validatePhone, 'Номер
телефону має бути у форматі +380xxxxxxxxx');
    const isValidPassword = validateInput(inputPassword, validatePassword,
'Пароль має містити від 6 до 12 символів');

    if (isValidEmail && isValidPhone && isValidPassword) {
        const user = {
            email: inputEmail.value,
            phone: inputPhone.value,
            password: inputPassword.value
        };

        localStorage.setItem('user', JSON.stringify(user));
        auth(user);
    }
};

logInForm.addEventListener('submit', validateFormOnSubmit);

const validateInputOnFocusOut = (input, validationFunction, errorMessage) => {
    input.addEventListener('focusout', () => {
        clearError(input);
        if (input.value.trim() !== '' && !validationFunction(input.value)) {
            showError(input, errorMessage);
        }
    });
};

validateInputOnFocusOut(inputEmail, validateEmail, 'Невірний формат електронної
пошти');
validateInputOnFocusOut(inputPhone, validatePhone, 'Номер телефону має бути у
форматі +380xxxxxxxxx');
validateInputOnFocusOut(inputPassword, validatePassword, 'Пароль має містити від
6 до 12 символів');
```

Файл game.css

```
article,
aside,
details,
figcaption,
figure,
footer,
header,
hgroup,
menu,
```

```
nav,
section {
    display: block;
}

body {
    line-height: 1;
}

ol,
ul {
    list-style: none;
}

blockquote,
q {
    quotes: none;
}

blockquote:before,
blockquote:after,
q:before,
q:after {
    content: '';
    content: none;
}

table {
    border-collapse: collapse;
    border-spacing: 0;
}
/* End of Reset.css*/

body {
    font-family: sans-serif;
}
.hide {
    display: none;
}

.car, .enemy {
    width: 50px;
    height: 100px;
    background: transparent url("../img/game/player.png") center / cover no-
repeat;
    position: absolute;

}

.gameArea {
    width: 300px;
    height: 100%;
    position: relative;
    margin: auto;
    background: rgb(49, 47, 47);
    overflow: hidden;
}

.line {
    position: absolute;
    width: 10px;
    height: 50px;
    background: rgb(253, 251, 251);
    left: 145px;
```

```
}

.start {
   position: absolute;
   left: 0;
   top: 0;
   width: 100%;
   margin: 0;
   padding: 2rem 0;
   text-align: center;
   z-index: 200;
   color: white;
   font-size: 1.5rem;
   background: #000;
   cursor: pointer;

}

.score {
   position: absolute;
   top: 0;
   left: 0;
   width: 100%;
   background: #000;
   padding: 2rem 0;
   z-index: 200;
   font-size: 1.5rem;
   color: skyblue;
   text-align: center;
}

.back {
   position: absolute;
   top: 150px;
   left: 10px;
   width: 10%;
   background: #000;
   padding: 2rem 0;
   z-index: 200;
   font-size: 1.5rem;
   color: skyblue;
   text-align: center;
   text-decoration: none;
}
```

Файл restaurant.html

```
<!DOCTYPE html>
<html lang="uk">

<head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Доставка їжі додому</title>
   <link
href="https://fonts.googleapis.com/css?family=Roboto:400,700&display=swap&subset
=cyrillic" rel="stylesheet" />
   <link rel="stylesheet" href="css/normalize.css" />
   <link rel="stylesheet" href="css/style.css" />
   <link rel="stylesheet" href="css/game.css" />
```

```
</head>

<body>
   <div class="container">
      <header class="header">
         <a href="index.html" class="logo">
            <img src="img/icon/logo.svg" alt="Logo" />
         </a>
         <label class="search">
            <input type="text" class="input input-search"
onkeyup="search_food()" placeholder="Пошук блюд" id="searchbar" />
         </label>
         <div class="buttons">
            <a href="home.html" class="button button-primary " style="text-
decoration: none;">
               <span class="button-text">Головна</span>
            </a>
            <span class="user-name"></span>
            <button class="button button-primary button-auth">
               <span class="button-auth-svg"></span>
               <span class="button-text">Увійти</span>
            </button>
            <button class="button button-cart" id="cart-button">
               <span class="button-cart-svg"></span>
               <span class="button-text">Кошик</span>
            </button>
            <button class="button button-primary button-out">
               <span class="button-text">Вийти</span>
               <span class="button-out-svg"></span>
            </button>
         </div>
      </header>
   </div>



   <main class="main">
      <div class="container">
         <section class="menu">
            <div class="section-heading">
               <h2 class="section-title restaurant-title">Піцца Плюс</h2>
               <div class="card-info">
                  <div class="rating">
                     4.5
                  </div>
                  <div class="price">До 900 ₴</div>
                  <div class="category">Піцца</div>
               </div>

            </div>
            <div class="cards cards-menu">
            </div>
         </section>
      </div>

   </main>

   <footer class="footer">
      <div class="container">
         <div class="footer-block">
            <img src="img/icon/logo.svg" alt="logo" class="logo footer-logo" />
         </div>
```

```
         <!-- /.footer-block -->
      </div>
   </footer>

   <div class="modal modal-cart">
      <div class="modal-dialog">
         <div class="modal-header">
            <h3 class="modal-title">Кошик</h3>
            <button class="close">&times;</button>
         </div>
         <!-- /.modal-header -->
         <div class="modal-body">

         </div>
         <!-- /.modal-body -->
         <div class="modal-footer">
            <span class="modal-pricetag">0 ₴</span>
            <div class="footer-buttons">
               <button class="button button-primary"
onclick="tryGame()">Оформити замовлення</button>
               <button class="button clear-cart">Відміна</button>
            </div>
         </div>

      </div>

   </div>

   <div class="modal-auth">
      <div class="modal-dialog modal-dialog-auth">
         <button class="close-auth">&times;</button>
         <form id="logInForm">
            <fieldset class="modal-body">
               <legend class="modal-title">Вхід</legend>
               <label class="label-auth">
                  <span>Email</span>
                  <input required id="email" type="email">
                  <div class="error-message" id="email-error"></div>
               </label>
               <label class="label-auth">
                  <span>Номер телефону</span>
                  <input required id="phone" type="tel" pattern="^\+38\d{10}$"
title="Введіть номер у форматі +380xxxxxxxxx">
                  <div class="error-message" id="phone-error"></div>
               </label>
               <label class="label-auth">
                  <span>Пароль</span>
                  <input id="password" type="password" minlength="6"
maxlength="12">
                  <div class="error-message" id="password-error"></div>
               </label>
            </fieldset>

            <div class="">
               <div class="footer-buttons">
                  <button class="button button-primary button-login"
type="submit" disabled>Увійти</button>
               </div>
            </div>
         </form>
      </div>
   </div>
   <script src="./js/auth.js"></script>
   <script src="./js/menu.js"></script>
```

```
    <script src="./js/partners.js"></script>
    <script src="./js/cart.js"></script>
    <script src="./js/game.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/js-
sha256/0.9.0/sha256.min.js"></script>  <!--бібліотека js-sha256 для шифрування
пароля -->
    <script src="./js/users.js"></script>


</body>

</html>
```

Файл japanese-cuisine.json

```
[
  {
    "id": "jc01",
    "name": "Унагі",
    "description": "Вугор, лосось, сир філадельфія, огірок, омлет тамаго-які,
соус спайс, соус унагі, кунжут, 320г.",
    "price": 350,
    "image": "img/japanese-cuisine/unagi.jpg"
  },
  {
    "id": "jc02",
    "name": "Рол Інь-Янь",
    "description": "Рис, водорості норі, омлет томаго, сир філадельфія, авокадо,
огірок, вугор копчений, 280г.",
    "price": 280,
    "image": "img/japanese-cuisine/yin-yang.jpg"
  },
  {
    "id": "jc03",
    "name": "Сет Імператор",
    "description": "Філадельфія лайт, Філадельфія сурімі, Філадельфія з вугрем,
Філадельфія з авокадо, 1150г. ",
    "price": 1120,
    "image": "img/japanese-cuisine/seth-emperor.jpg"
  },
  {
    "id": "jc04",
    "name": "Са-шин",
    "description": "Лосось, сир філадельфія, огірок, помідори, тобіко, сурімі.
220г. ",
    "price": 240,
    "image": "img/japanese-cuisine/sa-shin.jpg"
  },
  {
    "id": "jc05",
    "name": "Хаяші",
    "description": "Рис, кунжут, сир філадельфія, огірок, лосось, чука-вакаме,
норі, горіховий соус, соус спайс, 300г. ",
    "price": 220,
    "image": "img/japanese-cuisine/hayashi.jpg"
  },
  {
    "id": "jc06",
    "name": "Сет Хіпіpі",
    "description": "Сет з 5-ти суш: з лососем, креветкою, вугрем, морським
```

```
окунем та тунцем, 500г.",
    "price": 650,
    "image": "img/japanese-cuisine/seth-nigiri.jpg"
  },
  {
    "id": "jc07",
    "name": "Сет Футомакі",
    "description": "Складається з таких ролів: Осака, Хонсю, Чікен-рол, Курума
Ебі, 900г.",
    "price": 600,
    "image": "img/japanese-cuisine/seth-futomaki.jpg"
  },
  {
    "id": "jc08",
    "name": "Рол Туна",
    "description": "Рис, норі, тунець філе, огірок, авокадо, вугор, салат мікс,
омлет японський, тобіка чорна, 340г.",
    "price": 340,
    "image": "img/japanese-cuisine/role-tuna.jpg"
  },
  {
    "id": "jc09",
    "name": "Васабі",
    "description": "20г.",
    "price": 8,
    "image": "img/japanese-cuisine/wasabi.jpg"
  },
  {
    "id": "jc10",
    "name": "Імбир",
    "description": "20г.",
    "price": 14,
    "image": "img/japanese-cuisine/ginger.jpg"
  }
]
```

Файл taste-symphony.json

```
[
  {
    "id": "gl01",
    "name": "Пельмені теля с поросем",
    "description": "Справжні самоліпні пельмені з пухнастого тіста на сільських
жовтках та сметані",
    "price": 390,
    "image": "img/taste-symphony/dumplings.jpg"
  },
  {
    "id": "gl02",
    "name": "Плов",
    "description": "Плов на вугіллі з правильного рису з бараниною, пряностями
та овощами на курдючному жирі",
    "price": 520,
    "image": "img/taste-symphony/plov.jpg"
  },
  {
    "id": "gl03",
    "name": "Курчя на вертелі",
    "description": "Подається на лаваші зі свіжим томатом та огірком, з
маринованою цибулею та з шашличним соусом.",
    "price": 490,
    "image": "img/taste-symphony/chick.jpg"
  },
  {
```

```
    "id": "gl04",
    "name": "Суп Курочка ряба",
    "description": "Курка тушка, морква, макарони, зелень, яйцу, сіль, спеції.",
    "price": 140,
    "image": "img/taste-symphony/chicken-soup.jpg"
  },
  {
    "id": "gl05",
    "name": "Відбивна з порося",
    "description": "Товстий шматок парної свинини з томатами та грибами під
сирною скоринкою",
    "price": 490,
    "image": "img/taste-symphony/pig-chop.jpg"
  },
  {
    "id": "gl06",
    "name": "Теля в сливочному соусі",
    "description": "Теля з шампіньйонами та цибулею обсмажується у вершково-
сметанному соусі з картопляним пюре та солоними огірками.",
    "price": 530,
    "image": "img/taste-symphony/calf-sauce.jpg"
  },
  {
    "id": "gl07",
    "name": "Судак запечений",
    "description": "Філе судака з картоплею, сільською сметаною, травами та
цибулею",
    "price": 490,
    "image": "img/taste-symphony/zander.jpg"
  },
  {
    "id": "gl08",
    "name": "Мурманські кальмари",
    "description": "Мурманський кальмар з часниковою олією",
    "price": 420,
    "image": "img/taste-symphony/squid.jpg"
  },
  {
    "id": "gl09",
    "name": "Щуча юшка на карасях",
    "description": "Наваристa похідна юшка на чотирьох рибах, лаврусі, перці та
цибулі з горілкою.",
    "price": 380,
    "image": "img/taste-symphony/ear.jpg"
  }
]
```

Файл style.css

```
* {
    box-sizing: border-box;
}

body {
    font-family: "Roboto", sans-serif;
    padding-top: 44px;
}

a,
button {
```

```
    cursor: pointer;
    color: inherit;
}

.hide {
    display: none;
}

.container {
    max-width: 1200px;
    margin: auto;
}

.header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-bottom: 40px;
    margin-top: 10px;
}

.input {
    background-color: #ffffff;
    border: 1px solid #d9d9d9;
    border-radius: 2px;
    font-size: 16px;
    line-height: 24px;
    padding: 8px 8px 8px 35px;
    background-repeat: no-repeat;
    background-position: left 11px center;
}

.address {
    flex: 0.8;
}

.input-address {
    width: 100%;
    background-image: url(../img/icon/home.svg);
}

.search {
    margin-left: auto;
}

.input-search {
    width: 300px;
    background-image: url(../img/icon/search.svg);

}

.buttons {
    display: flex;
    align-items: center;
}

.button {
    display: flex;
    align-items: center;
    padding: 8px 16px;
    background: #ffffff;
    border: 1px solid #d9d9d9;
    box-sizing: border-box;
    box-shadow: 0 0 2px rgba(0, 0, 0, 0.0015);
```

```
    border-radius: 2px;
    color: #595959;
    font-size: 16px;
    line-height: 24px;
}

.button:hover {
    background: #1890ff;
    border: 1px solid #1890ff;
    color: #fff;
}


.button-primary {
    background: #ff6600;
    border: 1px solid #ff6600;
    color: #fff;
    margin-right: 10px;
    border-radius: 10px 10px;
}

.button-primary:hover {
    background: #ffffff;
    border: 1px solid #d9d9d9;
    color: #595959;
}

.button-icon {
    margin-right: 6px;
}


.button-card-text {
    margin-right: 10px;
}

.button-auth {
    background-position: 20px 13px;
}

.button-primary .button-auth-svg {
    width: 24px;
    height: 24px;
    background-color: #fff;
    -webkit-mask: url("../img/icon/user.svg") no-repeat 50% 50%;
    mask: url("../img/icon/user.svg") no-repeat 50% 50%;
    background-repeat: no-repeat;
}

.button-primary:hover .button-auth-svg {
    background-color: #595959;
}

.button .button-cart-svg {
    width: 24px;
    height: 24px;
    background-color: #595959;
    -webkit-mask: url("../img/icon/shopping-cart.svg") no-repeat 50% 50%;
    mask: url("../img/icon/shopping-cart.svg") no-repeat 50% 50%;
    background-repeat: no-repeat;
}

.button-primary .button-cart-svg {
    background-color: #fff;
```

```
}

.button:hover .button-cart-svg {
    background-color: #fff;
}

.button-primary:hover .button-cart-svg {
    background-color: #595959;
}

.error-message {
    color: red;
    font-weight: bold;
    text-align: center;
    margin-top: 10px;
}

.button-cart {
    display: none;
    margin: 0 5px;
}

.button-out {
    display: none;
    margin: 0 5px;
}

.user-name {
    display: none;
    margin-right: 20px;
    font-weight: bold;
    font-size: 18px;
}

.promo-slider {

    width: 600px;
    height: 300px;

}

.promo {
    box-shadow: 0 7px 12px rgba(158, 158, 163, 0.1);
    border-radius: 10px;
    padding: 68px 70px;
    margin-bottom: 56px;
}
/* Стилі для заголовка та слайдера */
.carousel-caption {
    background: rgba(0, 0, 0, 0.5);
    padding: 10px;
    border-radius: 5px;
    color: white;
}

.carousel-caption h5 {
    font-size: 24px;
    margin: 0;
}

.carousel-caption p {
    font-size: 16px;
    margin: 0;
}
```

```
.pizza {
    background: #fff1b8 url(../img/promo/pizza.png) no-repeat top -100px right -
250px / 830px
}

.kebab {
    background: #D6E4FF url(../img/promo/kebab.png) no-repeat top 45px right
40px / 450px;
}

.vegetables {
    background: #FFF566 url(../img/promo/vegetables.png) no-repeat top 0 right 0
/ 825px
}

.sushi {
    background: #FFF1F0 url(../img/promo/sushi.png) no-repeat top 10px right
15px / 500px
}

.promo-title {
    font-style: normal;
    font-weight: bold;
    font-size: 39px;
    line-height: 46px;
    color: #302c34;
}

.promo-text {
    font-style: normal;
    font-weight: normal;
    font-size: 24px;
    line-height: 28px;
    color: #302c34;
    max-width: 538px;
}

.main {
    background: linear-gradient(180deg, rgba(245, 245, 245, 0) 1.04%, #f5f5f5
100%);
}

.section-heading {
    display: flex;
    align-items: center;
    margin-bottom: 44px;
}

.section-title {
    font-style: normal;
    font-weight: bold;
    font-size: 36px;
    line-height: 42px;
    margin: 0 30px 0 0;
    color: #000000;
}

.cards {
    display: flex;
    align-items: flex-start;
    justify-content: space-between;
    flex-wrap: wrap;
}
```

```
.card {
    background: #ffffff;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
    border-radius: 7px;
    overflow: hidden;
    margin-bottom: 30px;
    flex-basis: 31%;
    text-decoration: none;

}

.card-restaurant {
    cursor: pointer
}

.card-image {
    width: 100%;
    height: 250px;
    object-fit: cover;
}


.card-text {
    padding: 20px 23px 35px;
    min-height: 275px;
    display: flex;
    flex-direction: column;
}



.restaurants .card-text {
    min-height: auto;
}

.card-heading {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-bottom: 10px;
}

.card-title {
    margin: 0;
    font-style: normal;
    font-weight: bold;
    font-size: 22px;
    line-height: 32px;
}

.card-title-reg {
    font-weight: 400;
}

.card-tag {
    font-style: normal;
    font-weight: normal;
    font-size: 12px;
    line-height: 20px;
    color: #ffffff;
    background: #262626;
    border-radius: 2px;
    padding: 1px 8px;
```

```
}

.card-info {
    display: flex;
    align-items: center;
    flex-wrap: wrap;
}

.card-buttons {
    display: flex;
    margin-top: 24px;
    flex-grow: 1;
    align-items: flex-end;
}

.card-price-bold {
    font-weight: bold;
    font-size: 20px;
    line-height: 32px;
    margin-left: 30px;
}

.rating {
    background-image: url("../img/icon/rating.svg");
    background-repeat: no-repeat;
    background-position: 0 7px;
    padding-left: 20px;
    margin-right: 26px;
    color: #ffc107;
    font-weight: bold;
    font-size: 18px;
    line-height: 32px;
}

.price,
.category {
    color: #8c8c8c;
    font-size: 18px;
    line-height: 32px;
}

.price {
    margin-right: 10px;
}

.ingredients {
    color: #8c8c8c;
    font-size: 18px;
    line-height: 21px;
}

.category {
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: nowrap;
    max-width: 150px;
}

.price:after {
    content: "";
    width: 5px;
    height: 5px;
    background-color: #8c8c8c;
    display: inline-block;
```

```
    vertical-align: middle;
    border-radius: 50%;
    margin-left: 10px;
}

.footer {
    padding: 60px 0;
}

.footer-block {
    display: flex;
    align-items: center;
    justify-content: space-between;
}

.footer-nav {
    margin-left: 35px;
    margin-right: auto;
}

.footer-link {
    display: inline-block;
    color: #595959;
    font-style: normal;
    font-weight: normal;
    font-size: 18px;
    line-height: 21px;
    text-decoration: none;
}

.footer-link:not(:last-child) {
    margin-right: 15px;
}

.social-links {
    display: flex;
    align-items: center;
}

.social-link:not(:last-child) {
    margin-right: 21px;
}

.modal {
    position: fixed;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.4);
    display: none;
    z-index: 999;
}

.modal-auth {
    position: fixed;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.4);
    display: none;
    z-index: 999;
}
```

```
.is-open {
    display: flex;
}
.curved-bg {
    position: relative;
    background-color: #ffffff;
    border-radius: 50% 50% 0 0;
    overflow: hidden;
    padding: 40px 0;
}

.form-container {
    padding: 0 45px;
}
.button-login {
    background: #ff6600;
    color: #fff;
    width: 100%;
    border-radius: 10px 10px;
    display: flex;
    justify-content: center;
    align-items: center;
}

.modal-dialog {
    max-width: 780px;
    width: 95%;
    background: #ffffff;
    border-radius: 5px;
    margin: auto;
    padding: 40px 45px;
}

.modal-dialog-auth {
    width: auto;
    position: relative;
}

.label-auth {
    display: block;
    margin: 30px;
}

.label-auth span {
    width: 80px;
    display: inline-block;
}

.modal-header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-bottom: 45px;
}

.modal-title {
    margin: 0 auto;
    font-weight: bold;
    font-size: 36px;
    line-height: 42px;
    text-align: center;
```

```
}

.close {
    font-size: 36px;
    border: none;
    background-color: transparent;
}

.close-auth {
    font-size: 36px;
    border: none;
    background-color: transparent;
    position: absolute;
    top: 10px;
    right: 20px;
}

.modal-body {
    margin-bottom: 22px;
}

.food-row {
    display: flex;
    align-items: flex-start;
    justify-content: space-between;
    padding-bottom: 8px;
    border-bottom: 1px solid #d9d9d9;
    margin-bottom: 15px;
}

.food-name {
    font-weight: normal;
    font-size: 18px;
    line-height: 32px;
}

.food-price {
    margin-left: auto;
    margin-right: 47px;
    font-weight: bold;
    font-size: 20px;
    line-height: 32px;
}

.food-counter {
    display: flex;
    align-items: center;
}

.counter-button {
    display: flex;
    align-items: center;
    justify-content: center;
    width: 39px;
    height: 32px;
    background: #ffffff;
    border: 1px solid #40a9ff;
    box-sizing: border-box;
    border-radius: 2px;
    font-weight: normal;
    font-size: 14px;
    line-height: 22px;
    color: #40a9ff;
}
```

```
.counter-button:hover {
    background: #40a9ff;
    border: 1px solid #ffffff;
    color: #ffffff;
}

.counter {
    font-size: 16px;
    line-height: 24px;
    margin-left: 10px;
    margin-right: 10px;
}

.modal-footer {
    display: flex;
    align-items: center;
    justify-content: space-between;

}

.modal-auth .modal-footer {
    justify-content: flex-end;
}

.footer-buttons {
    display: flex;
    align-items: center;
}

.modal-pricetag {
    background: #262626;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 20px;
    font-size: 20px;
    line-height: 23px;
}

@media (max-width: 1366px) {
    .container {
        max-width: 960px;
    }

    .pizza {
        background-position: center right -300px;
        background-size: 750px;
    }


    .kebab {
        background-position: top -5px right -93px;
        background-size: 530px;
    }

    .vegetables {
        background-position: top 0 right -93px;
        background-size: 820px;
    }

    .sushi {
        background-position: center right -90px;
    }
```

```
    .rating {
        margin-right: 15px;
    }

    .category,
    .price {
        font-size: 14px;
    }
}

@media (max-width: 992px) {
    .container {
        max-width: 750px;
    }



    .pizza {
        padding: 50px;
        background-size: 500px;
        background-position: center right -200px;
    }


    .kebab {
        background-position: top 55px right -100px;
        background-size: 385px;
    }

    .vegetables {
        background-position: top 0 right -280px;
        background-size: 810px;
    }

    .sushi {
        background-position: top 30px right -175px;
    }



    .promo-text {
        font-size: 18px;
        max-width: 400px;
    }

    .card {
        flex-basis: 49%;
    }

    .footer-link {
        font-size: 16px;
    }
}

@media (max-width: 768px) {
    .container {
        max-width: 560px;
    }

    .card-info {
        flex-wrap: wrap;
    }

    .card .rating {
```

```
        flex-basis: 100%;
    }

    .card-title {
        font-size: 18px;
    }

    .card-text {
        min-height: 290px;
    }

    .card-price-bold {
        margin-left: 20px;
    }


    .pizza {
        background-size: 400px;
        background-position: bottom 50px right -200px;
    }


    .kebab {
        background-position: top -5px right -74px;
        background-size: 300px;
    }

    .vegetables {
        background-position: top 0 right -240px;
        background-size: 702px;
    }

    .sushi {
        background-position: top -24px right -75px;
        background-size: 302px;
    }

    .promo-title {
        font-size: 24px;
        line-height: 1.4;
    }

    .promo-text {
        margin-top: 0;
    }
}

@media (max-width: 578px) {
    .container {
        width: 90%;
    }

    .address {
        min-width: 100%;
        order: 1;
    }

    .header {
        flex-wrap: wrap;
    }

    .input-address {
        margin-top: 15px;
        order: 5;
```

```
        flex: 1;
    }

    .promo {
        background-image: none;
    }

    .promo-title {
        margin-bottom: 10px;
    }

    .section-title {
        font-size: 20px;
    }

    .card {
        flex-basis: 100%;
    }

    .card-text {
        min-height: auto;
    }

    .card-image {
        width: 100%;
    }

    .footer {
        padding: 30px 0;
    }

    .footer-block {
        align-items: flex-start;
    }

    .footer-nav {
        margin-left: 0;
        margin-right: 0;
        order: 0;
        display: flex;
        flex-direction: column;
    }

    .footer-logo {
        margin-right: 15px;
        order: 1;
    }

    .social-links {
        order: 2;
    }
}

@media (max-width: 480px) {
    .button-text {
        display: none;
    }

    .button-icon {
        margin: 0;
    }

    .user-name {
        margin: 10px;
```

```
    }

    .buttons {
        flex-wrap: wrap;
        margin-left: auto;
    }

    .button {
        min-height: 40px;
        padding: 5px 12px;

    }

    .button-out-svg {
        width: 24px;
        height: 24px;
        background-color: #fff;
        -webkit-mask: url(../img/icon/logout.svg) no-repeat 50% 50%;
        mask: url(../img/icon/logout.svg) no-repeat 50% 50%;
        background-repeat: no-repeat;
        -webkit-mask-size: 20px;
        mask-size: 20px;
    }

    .promo {
        padding: 20px;
    }

    .section-heading {
        flex-wrap: wrap;
    }

    .footer-block {
        flex-wrap: wrap;
        flex-direction: column;
        align-items: center;
    }

    .footer-logo {
        order: 0;
        margin-bottom: 20px;
    }

    .footer-nav {
        margin-bottom: 20px;
        text-align: center;
    }

    .footer-link:not(:last-child) {
        margin-right: 0;
    }
}
```