НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Курсова робота

з дисципліни «Програмування»

на тему: «Система замовлення їжі»

Виконав:
студент 1 курсу, групи ІА-31
Козир Ярослав Олександрович _____
(підпис)

Керівник:
асистент кафедри ІСТ
Степанов Андрій Сергійович _____
(підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент, Козир Я. О. _____

(підпис)

Київ – 2024 року

ЗМІСТ

ВСТУП

У сучасному світі сучасних технологій, де доступ до інформації та послуг швидко зростає, інтернет-сервіси замовлення їжі - важливий інструмент для задоволення потреб споживачів. Мій проект спрямований на забезпечення зручного та ефективного способу замовлення їжі, що відповідає сучасному ритму життя та потребам наших клієнтів.

У нашому сьогоднішньому світі, коли багато людей мають напружені графіки та не завжди можуть знайти час для приготування їжі або відвідання ресторану, інтернет-сервіс замовлення їжі стає незамінним помічником. З його допомогою, користувачі можуть швидко та зручно замовити смачну їжу в будь-який час, не виходячи з дому або робочого місця.

Основними завданнями мого проекту є:

- Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для швидкого та простого замовлення їжі.
- Розширення асортименту страв та меню, щоб задовольнити різноманітні смаки та вподобання наших клієнтів.
- Можливість грамотного розділення замовлень між робітниками, оскільки буде візуально зрозуміло які замовлення оброблюються, а які вже виконані.
- Передача власних даних задля розуміння того, хто є замовником.

1 ВИМОГИ ДО СИСТЕМИ

1.1 Функціональні вимоги до системи

Дефолтний користувач повинен мати можливість:

- Переглядати доступну інформацію про страви на меню.
- Створювати замовлення, обираючи які страви та яку їх кількість беремо та вводимо власні дані, щоб було зрозуміло куди та кому доставляти.

Адміністратор повинен мати можливості:

- Переглядати доступну інформацію про страви на меню.
- Можливість додавати нові страви до меню.
- Можливість видаляти страви з меню.
- Можливість переглядати замовлення, що створюють дефолтні користувачі.
- Можливість приймати, відхиляти та оформлювати замовлення як виконані.

1.2 Нефункціональні вимоги до системи

1. Відкрита архітектура:
   Система повинна мати модульну та розширювану архітектуру, що дозволить легко інтегрувати нові функції та розширювати можливості системи у майбутньому.
2. Веб-інтерфейс:
   Система має мати веб-інтерфейс, який забезпечить зручний доступ до функціоналу системи через будь-який сучасний веб-браузер.
3. Зручний та інтуїтивно-зрозумілий інтерфейс користувача:
   Інтерфейс користувача повинен бути зрозумілим і простим у використанні навіть для користувачів без досвіду використання подібних систем. Він має бути інтуїтивно зрозумілим та забезпечувати легкий доступ до всіх основних функцій системи.
4. Кросплатформенність:
   Система повинна бути доступною на різних платформах, таких як комп'ютери, планшети та смартфони, і забезпечувати увесь функціонал та зручний інтерфейс користувача на всіх цих пристроях.

2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами є користувачі системи: дефолтний користувач та адміністратор.

Адміністратору доступна майже уся функціональність, що і юзеру, окрім створення замовлення. В свою ж чергу, адмін мож редагувати меню, а не просто передивлятися, також може передивитися замовлення та обробити його.

Рисунок 2.1 – Діаграма прецедентів

2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.10

Таблиця 2.1 – Сценарій використання «Вхід в систему»

Назва	Вхід в систему
ID	1
Опис	Користувач виконує вхід у систему за допомогою свого облікового запису.
Актори	Дефолтний користувач, адміністратор
Вигоди компанії	Забезпечення доступу до персоналізованих можливостей та інформації для користувачів та адміністраторів
Частота користування	Постійно
Тригери	Користувач або адміністратор переходить на сторінку входу у систему
Передумови	Сторінка входу доступна для усіх користувачів
Постумови	Користувач або адміністратор успішно увійшов у систему і отримує доступ до персоналізованого інтерфейсу
Основний розвиток	Користувач або адміністратор вводить свої облікові дані (електронну пошту та пароль), натискає кнопку "Увійти"
Альтернативні розвитки	-

Виняткові ситуації	У випадку неправильних облікових даних користувача або адміністратора відображається повідомлення про помилку: нас перенаправляє на відповідну сторінку, де є змога повернутися на сторінку входу та ще раз ввести дані

Таблиця 2.2 – Сценарій використання «Головна сторінка(User)»

Назва	Головна сторінка(User)
ID	2
Опис	Користувач переглядає головну сторінку системи, де відображені посилання на меню з доступними стравами та сторінку створення замовлення.
Актори	Дефолтний користувач
Вигоди компанії	Забезпечення зручного та ефективного взаємодії з користувачами та сприяння зростанню продажів.
Частота користування	При відкритті сторінки
Тригери	Користувач переходить на головну сторінку системи
Передумови	Користувач увійшов у систему
Постумови	Користувач переглядає інформацію про страви та може створювати замовлення
Основний розвиток	Користувач відкриває головну сторінку системи
Альтернативні розвитки	Користувач виходить з облікового запису та повертається на сторінку входу, щоб змінити статус на адміністратора

Виняткові ситуації	-

Таблиця 2.3 – Головна сторінка (адміністратор)»

Назва	Головна сторінка (адміністратор)
ID	3
Опис	Користувач переглядає головну сторінку системи, де відображені посилання на меню з доступними стравами та сторінку створення замовлення.
Актори	Адміністратор
Вигоди компанії	Забезпечення зручного та ефективного взаємодії з адміністратором та сприяння зростанню продажів.
Частота користування	При відкритті сторінки
Тригери	Адміністратор переходить на головну сторінку системи
Передумови	Адміністратор увійшов у систему
Постумови	Адміністратор переглядає інформацію про страви та замовлення, може управляти стравами та замовленнями
Основний розвиток	Адміністратор відкриває головну сторінку системи
Альтернативні розвитки	Адміністратор виходить з облікового запису та повертається на сторінку входу, щоб змінити статус на дефолтного користувача.
Виняткові ситуації	-

Таблиця 2.4 – Сценарій використання «Перегляд меню»

Назва	Перегляд меню
ID	4
Опис	Користувач переглядає інформацію про всі доступні страви на меню
Актори	Дефолтний користувач, адміністратор
Вигоди компанії	Забезпечення зручного перегляду страв для користувачів, що збільшує ймовірність замовлення
Частота користування	Постійно
Тригери	Користувач переходить на сторінку перегляду меню
Передумови	Список страв доступний для перегляду на сторінці меню
Постумови	Користувач бачить перелік страв разом з інформацією про них, включно назву, ціну, естетичний вигляд страви та інгредієнти, що входять до даної страви.
Основний розвиток	Користувач відвідує сторінку меню і переглядає всі доступні страви
Альтернативні розвитки	Від імені Адміністратора можна окрім перегляду ще й видаляти наявні елементи меню
Виняткові ситуації	Можна повернутися на головну сторінку натиснувши на емблему сервісу

Таблиця 2.5 – Сценарій використання «Додавання нового елемента в меню»

Назва	Додавання нового елемента в меню
ID	5
Опис	Адміністратор додає нову страву до меню системи.
Актори	Адміністратор
Вигоди компанії	Поповнення асортименту страв та приваблення нових клієнтів.
Частота користування	При додаванні нових страв
Тригери	Адміністратор обирає опцію "Додати нову страву"
Передумови	Адміністратор увійшов у систему
Постумови	Нова страва додається до меню
Основний розвиток	Адміністратор обирає опцію "Додати нову страву", заповнює необхідні дані (назва, ціна, категорія, інгредієнти, фото продукту) та підтверджує додавання.
Альтернативні розвитки	Адміністратор скасовує додавання нової страви та повертається назад.
Виняткові ситуації	Через невірні підібрані за форматом фото після натиснення на кнопку додавання вас переадресує на сторінку з меню, а там не буде доданого елемента.

Таблиця 2.6 – Сценарій використання «Створення замовлень»

Назва	Створення замовлень
ID	6

Опис	Користувач складає замовлення на обрані страви
Актори	Дефолтний користувач
Вигоди компанії	Збільшення обсягу продажів, задоволення потреб клієнтів
Частота користування	Постійно
Тригери	Користувач обирає страви та здійснює дію "Створити замовлення"
Передумови	Користувач увійшов у систему та обрав страви з меню
Постумови	Замовлення створено, вам відобразило «чек» та зареєєструвало у списку активних замовлень
Основний розвиток	Користувач обирає необхідні страви з меню, вказує кількість та інші додаткові опції (якщо є), після чого підтверджує створення замовлення.
Альтернативні розвитки	Користувач скасовує створення замовлення У випадку того що база даних меню пуста у нас не буде з чого обирати замовлення
Виняткові ситуації	Якщо не виберете жодної страви та натиснете «Оформити заиовлення» або ж не введете ваші дані задля отримання інформації про замовника то отримаєте відповідне повідомлення: «Замовлення не може бути створене. Будь ласка, виберіть страви та введіть ваші дані перед оформленням замовлення.»

Таблиця 2.7 – Сценарій використання «Перегляд переліку замовлень»

Назва	Перегляд переліку замовлень
ID	7
Опис	Адміністратор переглядає список замовлень у системі
Актори	Адміністратор
Вигоди компанії	Адміністратор може відстежувати та керувати всіма замовленнями
Частота користування	Постійно
Тригери	Адміністратор обирає функцію "Переглянути замовлення"
Передумови	Адміністратор увійшов у систему
Постумови	Відображається список всіх замовлень у системі
Основний розвиток	Адміністратор переходить до панелі керування або обирає функцію "Замовлення", де може переглянути та керувати всіма замовленнями
Альтернативні розвитки	-
Виняткові ситуації	У випадку відсутності замовлень буде повідомлення: «Наразі повідомлень немає»

Таблиця 2.8 – Сценарій використання «Прийняття замовлення»

Назва	Прийняття замовлення
ID	8
Опис	Адміністратор підтверджує отримання та приймає замовлення
Актори	Адміністратор
Вигоди компанії	Адміністратор підтверджує та обробляє замовлення
Частота користування	Постійно
Тригери	Адміністратор обирає функцію "Прийняти замовлення"
Передумови	Адміністратор увійшов у систему та переглянув список замовлень
Постумови	Замовлення позначається як прийняте, з'являється підтвердження, повідомлення стає зеленим
Основний розвиток	Адміністратор обирає замовлення та підтверджує його прийняття
Альтернативні розвитки	-
Виняткові ситуації	-

Таблиця 2.9 – Сценарій використання «Відхилення замовлення»

Назва	Відхилення замовлення
ID	9
Опис	Адміністратор відхиляє неприйняте замовлення
Актори	Адміністратор
Вигоди компанії	Адміністратор відхиляє неприйняте замовлення
Частота користування	Постійно
Тригери	Адміністратор обирає функцію "Відхилити замовлення"
Передумови	Адміністратор увійшов у систему та переглянув список замовлень
Постумови	Замовлення позначається як відхилене, зникає зі списку
Основний розвиток	Адміністратор обирає замовлення та підтверджує його відхилення
Альтернативні розвитки	-
Виняткові ситуації	-

Таблиця 2.10 – Сценарій використання «Позначення замовлення як виконане»

Назва	Позначення замовлення як виконане
ID	10
Опис	Адміністратор позначає замовлення як виконане
Актори	Адміністратор
Вигоди компанії	Адміністратор позначає замовлення як виконане
Частота користування	Постійно
Тригери	Адміністратор обирає функцію "Виконано"
Передумови	Адміністратор увійшов у систему та переглянув список замовлень
Постумови	Замовлення позначається як виконане, загорається жовтим, зникає зі списку через невеликий час
Основний розвиток	Адміністратор обирає замовлення та позначає його як виконане
Альтернативні розвитки	-
Виняткові ситуації	-

3 АРХІТЕКТУРА СИСТЕМИ

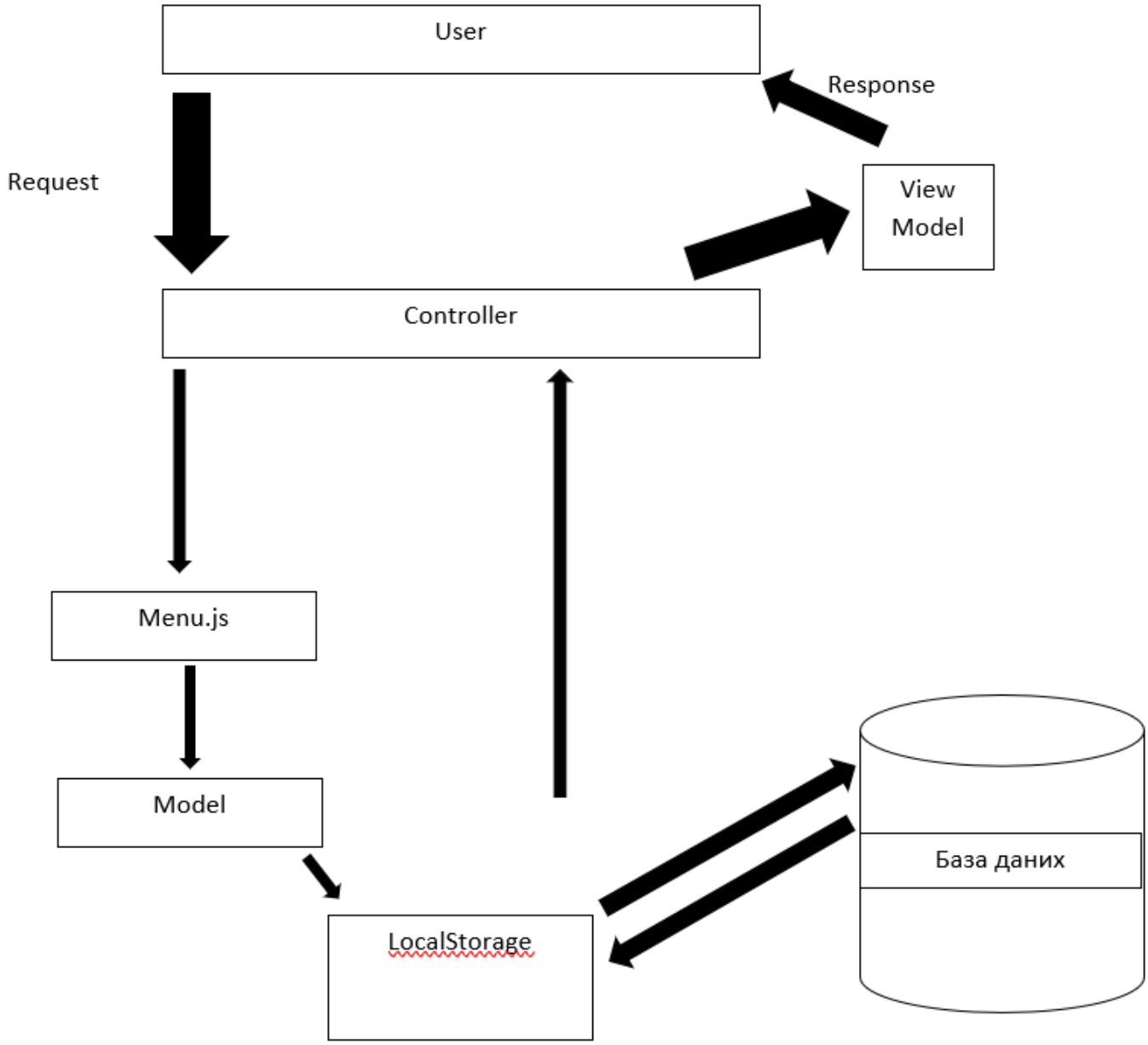


Рисунок 3.1 – Загальна архітектура системи

Система нашого проекту складається з наступних компонентів:

Графічний інтерфейс: цей компонент відповідає за візуальне відображення системи для користувача та взаємодію з ним. Забезпечує можливість користувачам переглядати доступні страви у меню, робити замовлення та отримувати повідомлення про статус замовлення.

Серверна частина: ця частина системи виконує обробку запитів від користувачів, реалізацію бізнес-логіки та взаємодію з базою даних. Отримує запити від графічного інтерфейсу, обробляє їх та відправляє відповіді.

База даних: цей компонент зберігає всю необхідну інформацію для роботи системи, включаючи дані про страви у меню, інформацію про замовлення користувачів та інші важливі дані.

До серверної частини належать такі елементи:

- Контролер: приймає дані з графічного інтерфейсу та маршрутизує їх до відповідного сервісу для подальшої обробки.

- Сервіс: виконує основну бізнес-логіку системи, таку як обробка замовлень, керування меню тощо.

- Модель: представляє структуру даних, що використовується в системі, наприклад, модель страви чи модель замовлення.

- LocalStorage: відповідає за збереження та отримання даних на клієнтському боці, що дозволяє забезпечити постійність даних між сеансами роботи користувача.

Контролер отримує дані з графічного інтерфейсу та передає їх до відповідного сервісу. Сервіс обробляє ці дані відповідно до бізнес-логіки системи, також взаємодіє з базою даних через LocalStorage для збереження або отримання необхідних даних.

4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

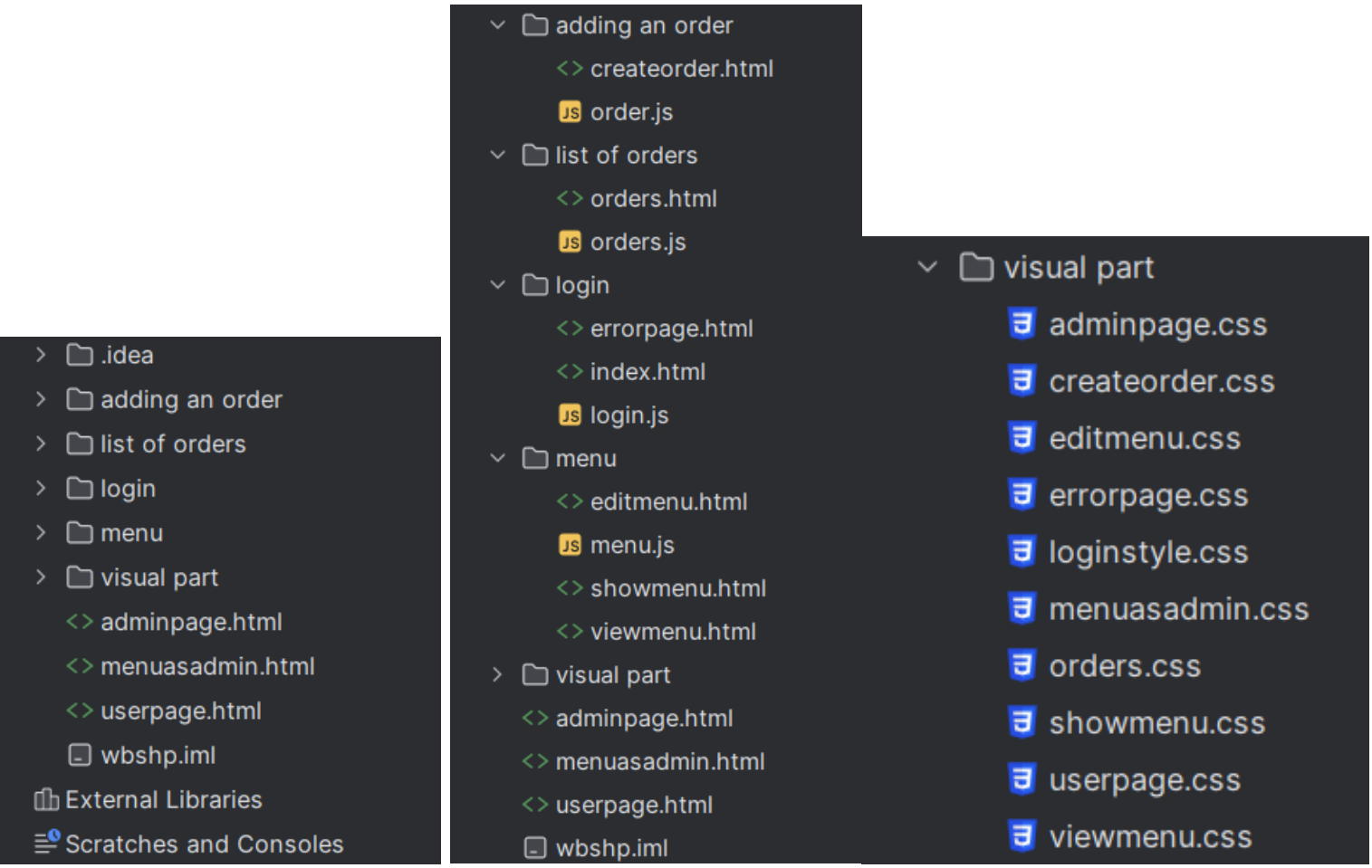


Рисунок 4.1 – Загальна структура проекту

Проект складається з вихідного коду, який в свою чергу можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти.

Я не імпортую бібліотеки, як потрібно при створенні логіки виключно на Java, натомість використовую JavaScript для прописання логіки зв’язку страв та їх особливостей між додаванням та переглядом меню з погляду різних акторів; створення замовлення та його передачі до списку замовлень.

4.2 Компоненти рівня доступу до даних

1. Локальне сховище (LocalStorage): Я використовую локальне сховище браузера для зберігання даних про замовлення. Це дозволяє зберігати дані про замовлення клієнтів без необхідності звертатися до сервера.

Отримання даних про замовлення з локального сховища

```
let orders = JSON.parse(localStorage.getItem('orders'))
```

Перевірка наявності даних про замовлення

```
if (orders && orders.length > 0) {
```

Логіка обробки даних про замовлення

```
} else {
    (Логіка обробки відсутності даних про замовлення) }
```

2. JavaScript-код взаємодіє з об'єктно-орієнтованою моделлю документу для отримання доступу до елементів сторінки і взаємодії з ними. Наприклад, зчитується дані з полів вводу, кнопок та інших елементів, а також відображається та оновлюється інформація на сторінці. Наприклад:

Задля отримання посилання на DOM-елементи

```
let submitButton = document.getElementById('submitOrder');
let customerNameInput = document.getElementById('customerName');
```

Я додаю обробники подій до елементів, таких як click, change, submit, і реагуєте на взаємодію користувача з елементами сторінки.

```
submitButton.addEventListener('click', submitOrder);
```

Встановлюю нові дані сторінки

```
customerInfo.textContent = `Замовив: ${orderData.name},  Дата: ${new Date(orderData.timestamp).toLocaleString()}`
```

Тут я створюю нові DOM-елементи за допомогою document.createElement(),та додаю їх до DOM-структури з допомогою appendChild() чи видаляю елементи зі сторінки.

```
let orderContainer = document.createElement('div');
orderContainer.classList.add('order-container');
```

3. Модель даних: передбачається наявність моделі даних, що описує структуру замовлень, включаючи дані про клієнта, вміст замовлення та інші деталі. Ця модель використовується для зберігання та обробки даних про замовлення.

4. Обробник подій: JavaScript-код включає обробники подій, які реагують на взаємодію користувача зі сторінкою (наприклад, клік по кнопці "Оформити замовлення" або "Прийняти"). Ці обробники виконують дії, такі як збереження даних з форми замовлення або зміна статусу замовлення.

```
submitButton.addEventListener('click', submitOrder);
acceptButton.addEventListener('click', () => acceptOrder(index, orderContainer));
rejectButton.addEventListener('click', () => rejectOrder(index, orderContainer));
completeButton.addEventListener('click', () => completeOrder(orderContainer));
```

1) Цей рядок встановлює обробник події для кнопки "Оформити замовлення". Коли користувач клікає на цю кнопку, виконується функція submitOrder, яка ініціює процес оформлення замовлення.

2) Цей рядок встановлює обробник події для кнопки "Прийняти" в блоку замовлення. Коли користувач клікає на цю кнопку, виконується функція acceptOrder, яка приймає індекс замовлення та контейнер замовлення та виконує дії з прийняттям цього замовлення.

3) Цей рядок встановлює обробник події для кнопки "Відхилити" в блоку замовлення. Коли користувач клікає на цю кнопку, виконується функція rejectOrder, яка приймає індекс замовлення та контейнер замовлення та виконує дії з відхиленням цього замовлення.

4) Цей рядок встановлює обробник події для кнопки "Виконано" в блоку замовлення. Коли користувач клікає на цю кнопку, виконується функція completeOrder, яка виконує дії з позначенням замовлення як виконане та приховання контейнера замовлення.

4.3 Компоненти рівня бізнес-логіки

1. OrderService

Конструктор приймає об'єкт типу LocalStorageService.

Реалізує такі дії:

- Зберігання замовлення у локальному сховищі браузера.
- Оновлення статусу замовлення (прийняте, відхилене, виконане).
- Отримання даних про замовлення з локального сховища.

2. MenuService

Конструктор приймає об'єкт типу LocalStorageService.

Реалізує такі дії:

- Зберігання меню (інформації про страви та їх характеристики) у локальному сховищі браузера.
- Отримання списку страв та їх характеристик з локального сховища.

3. LocalStorageService

Реалізує роботу з локальним сховищем браузера для зберігання даних.

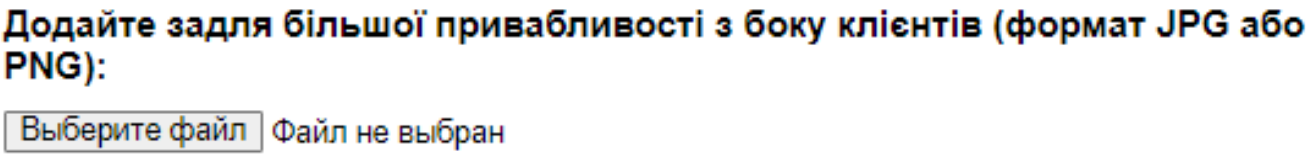
Реалізує такі дії:

- Збереження даних у локальному сховищі.
- Отримання даних з локального сховища.
- Оновлення даних у локальному сховищі.
- Видалення даних з локального сховища.

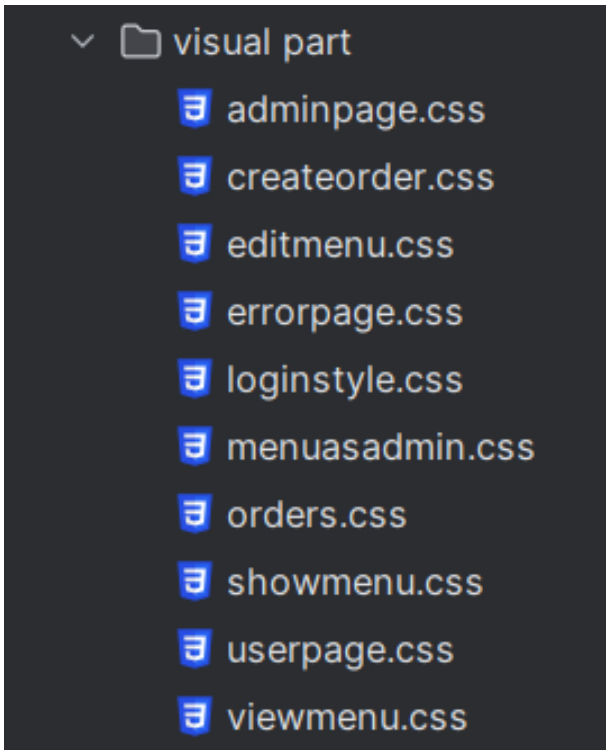
4.4 Компоненти рівня інтерфейсу користувача

Загалом, Інтерфейс складається з html-сторінок, на яких і міститься вся інформація, усі форми та кнопки, усі можливості дати запити та побачити відповідь. Але, задля зручності та кращого сприйняття, я також використав такі засоби, що суттєво апгрейднули візуальну складову:

- Зображення: при додаванні нового компонента меню, серед усіх особливостей страви, у нас є можливість додати і фотографію, аби замовник страви краще розумів, з чим він має справу. Таким чином ми підіймаємо інтерес до нашого продукту.



- CSS-компоненти: саме вони відповідають за стилізацію HTML – файлів: мова, яка безпосередньо відповідає за розташування на екрані, розмір, колір, вигляд форми, шрифт та додаткові, приємні для ока процеси. Саме завдяки css користувачу буде простіше та приємніше робити замовлення саме на цьому сайті.



- JavaScript: використовується для взаємодії з елементами сторінки та здійснення дій на основі дій користувача. Наприклад, коли користувач натискає на кнопку,

JavaScript може викликати функцію, яка змінює вміст сторінки або виконує інші дії, такі як відправка запиту на сервер.
Що стосується більш точного зв’язку, я звернуся до прикладів з мого коду:

1. Формування меню страв: у функції window.onload ви формуєте меню страв за допомогою JavaScript, яке потім відображається на сторінці. Це створює інтерфейс, який користувач може переглядати та вибирати страви.

2. Взаємодія з полями вводу: використовується JavaScript, щоб отримати дані з полів вводу, таких як ім'я, телефон і адреса. Ці дані потім використовуються для оформлення замовлення.

3. Додавання обробників подій: я прикріплюю обробники подій до різних елементів, таких як кнопки "Оформити замовлення", "Прийняти", "Відхилити" та "Виконано". Це дозволяє реагувати на дії користувача і виконувати відповідні дії, такі як збереження замовлення або зміна його статусу.

ВИСНОВКИ

Під час написання роботи, було проведено аналіз ринку CRM-систем в Україні, з особливою увагою до сервісів доставки їжі, таких як Glovo, Mister Am та інші. В ході цього аналізу були визначені переваги та недоліки існуючих систем. Основуючись на цьому дослідженні, я прийняв рішення розробити просту систему доставки їжі, щоб отримати досвід у роботі з цим типом програмного забезпечення та зрозуміти основні виклики, які можуть виникнути. Ця система має великий потенціал для подальшого розвитку та вдосконалення, і вона дозволить мені отримати цінний досвід у галузі розробки програмного забезпечення для сфери послуг.

Першим кроком було сформульовано функціональні та нефункціональні вимоги до системи, що визначило першочергові реакції системи. Далі, було обрано технології, на яких буде написана система. HTML, CSS та JavaScript були обрані як основні технології для розробки графічного інтерфейсу користувача, оскільки вони надають можливість швидко розробляти і впроваджувати прості веб-інтерфейси.
Для локального збереження даних про замовлення було використано localStorage, що надає можливість зберігати дані клієнтів без необхідності звертатися до сервера. JavaScript використовувався для взаємодії з об'єктно-орієнтованою моделлю документу (DOM), отримання доступу до елементів сторінки та взаємодії з ними. Мова HTML стала основою сторінок, а CSS освоїла візуальну структуру.
Таким чином, основними технологіями, використаними в проекті, були HTML, CSS, JavaScript та localStorage.

Наступним кроком були описані сценарії використання. Система підтримує функціонал перегляду меню сервісу та створення замовлення, ввівши скільки яких страв ви хочете замовити. Та в результаті ви отримаєте чек, який вам покаже що ви замовили та яка вартість вашого замовлення. Головний нюанс використання проекту- різний функціонал від імені акторів User та Administrator. Administrator може редагувати страви, видаляючи та додаючи нові в пару кліків, що є важливим, враховуючи постійні зміни та нові ідеї персоналу. До того ж, вкрай ефективно реалізована система обробки замовлень:

1. Вкрай просто та зрозуміло переглядається кожне замовлення, де видно замовника, його адресу та номер, час замовлення, що передається автоматично, що варто готувати та яка виручка чекає за це замовлення.,
2. Замовлення можна відхилити, якщо у вас немає потрібних інгредієнтів чи є певні проблеми з замовником. А воно не буде відволікати від інших замовлень.
3. Замовлення можна прийняти, тоді інші співробітники будуть розуміти, що цим замовленням вже хтось займається та можна взятися за інше.
4. Замовлення можна позначити виконаним, аби воно не заважало.

Далі, ми створили загальну архітектуру проекту, використовуючи парадигму шаблону проектування MVC (Model-View-Controller). Це дозволило нам логічно розділити систему на основні компоненти.

У моєму проекті, використання LocalStorage можна інтегрувати на рівні контролера. Контролер може мати методи, які зберігають дані форми замовлення або налаштувань користувача в LocalStorage під час їх введення. Це дозволить тимчасово зберігати ці дані на боці клієнта, доки вони не будуть готові до відправлення на сервер. Після відправлення даних на сервер або завершення сесії, можна видалити ці дані з LocalStorage для очищення пам'яті браузера.

Але варто врахувати що такий підхід буде ефективним лише для такого демо-проекту

Підсумовуючи вищезазначене, можна сказати, що наша система вирізняється простотою налаштування та оновлення функціоналу. Завдяки відкритій архітектурі, її розширення не становить проблему. Крім того, графічний інтерфейс системи легко налаштовується. Наша система є конкурентоспроможною та має потенціал для подальшого розвитку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. webguild - українська IT-спільнота - Pure CSS Animated Login Form — анімована форма входу на чистому CSS. [Електронний ресурс] Режим доступу: https://t.me/webguild/915
2. Людмила Пашкевич - Технологічний стек: HTML, CSS, JavaScript. [Електронний ресурс] Режим доступу: https://www.linkedin.com/pulse/%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D0%B8%D0%B9-%D1%81%D1%82%D0%B5%D0%BA-html-css-javascript-liudmyla-pashkevych-t9hef
3. localStorage [Електронний ресурс] Режим доступу: http://xn--80adth0aefm3i.xn--j1amh/localstorage#:~:text=localStorage%20%2D%20%D0%B7%D0%B1%D0%B5%D1%80%D1%96%D0%B3%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85%20%D0%BD%D0%B0%20%D0%BB%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%D1%83%20%D1%81%D1%85%D0%BE%D0%B2%D0%B8%D1%89%D1%96%20%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%B0.
4. localStorage збереже все [Електронний ресурс] Режим доступу: https://css.in.ua/article/localstorage-zberezhe-vse_376
5. Glovo [Електронний ресурс] Режим доступу: https://glovoapp.com/
6. Сохранение элемента html в localStorage и вывод [Електронний ресурс] Режим доступу: https://ru.stackoverflow.com/questions/915585/%D0%A1%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0-html-%D0%B2-localstorage-%D0%B8-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4
7. Довідник по CSS – властивостям [Електронний ресурс] Режим доступу: https://css.in.ua/css/properties
8. webguild - українська IT-спільнота - Сет градієнтних кнопок [Електронний ресурс] Режим доступу: https://t.me/webguild/904

9. Использование CSS-градиентов [Електронний ресурс] Режим доступу: https://developer.mozilla.org/ru/docs/Web/CSS/CSS_images/Using_CSS_gradients
10. Евгений Шкляр - Что такое localStorage и как им пользоваться [Електронний ресурс] Режим доступу: https://htmlacademy.ru/blog/js/localstorage#:~:text=%D0%A7%D1%82%D0%BE%D0%B1%D1%8B%20%D0%BE%D1%87%D0%B8%D1%81%D1%82%D0%B8%D1%82%D1%8C%20localStorage%20%D1%81%20%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E,localStorage.
11. <button> - элемент кнопки [Електронний ресурс] Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTML/Element/button
12. Ivan Gagarinov - Как сохранить объект в localstorage js [Електронний ресурс] Режим доступу: https://ru.hexlet.io/qna/javascript/questions/kak-sohranit-ob-ekt-v-localstorage-js#:~:text=%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D1%82%D1%8C%20%D0%B2%20localStorage%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B%20%D0%BD%D0%B5%D0%BB%D1%8C%D0%B7%D1%8F,stringify()%20.
13. HTML Web Storage API [Електронний ресурс] Режим доступу: https://www.w3schools.com/html/html5_webstorage.asp
14. Какой формат даты в JSON? [Електронний ресурс] Режим доступу: https://ru.stackoverflow.com/questions/1202596/%D0%9A%D0%B0%D0%BA%D0%BE%D0%B9-%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82-%D0%B4%D0%B0%D1%82%D1%8B-%D0%B2-json
15. CSS Button Effect [Електронний ресурс] Режим доступу: https://vm.tiktok.com/ZMMt6D3AG/
16. Storing and retrieving JavaScript objects in localStorage [Електронний ресурс] Режим доступу: https://blog.logrocket.com/storing-retrieving-javascript-objects-localstorage/
17. JavaScript Functions [Електронний ресурс] Режим доступу: https://www.w3schools.com/js/js_functions.asp
18. JavaScript and localStorage in a nutshell [Електронний ресурс] Режим доступу: https://www.tiny.cloud/blog/javascript-localstorage/

ДОДАТОК А

Лістінг програми

Посилання на проект - https://github.com/gfnf9977/WebShopSImpleByKoz

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Опис сторінки">
    <meta name="keywords" content="ключові слова">
    <link rel="stylesheet" href="loginstyle.css">
    <title>Document</title>
</head>
<body>
<section>
    <form id="loginForm">
        <h1>Login</h1>
        <div class="inputbox">
            <ion-icon name="person-outline"></ion-icon>
            <input type="text" id="username" required>
            <label for="username">Ім'я користувача</label>
        </div>
        <div class="inputbox">
            <ion-icon name="lock-closed-outline"></ion-icon>
            <input type="password" id="password" required>
            <label for="password">Пароль</label>
        </div>
        <button type="submit">Увійти</button>
```

```
    </form>
</section>
<script type="module"
src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
<script nomodule
src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"></script>
<script src="login.js"></script>
</body>
</html>
```

errorpage.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Error</title>
    <link rel="stylesheet" href="../visual%20part/errorpage.css">
</head>
<body>
<div class="container">
    <h1>Error</h1>
    <p>Sorry, an error occurred. Please try again later.</p>
    <a href="index.html" class="btn">Go Back to Home Page</a>
</div>
</body>
</html>
```

login.js

```
document.getElementById('loginForm').addEventListener('submit', function(event) {
    event.preventDefault();
```

```
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    if (username === 'user' && password === 'user123') {
        window.location.href = 'userpage.html';
    } else if (username === 'admin' && password === 'admin123') {
        window.location.href = 'adminpage.html';
    } else {
        window.location.href = 'errorpage.html';    }
});
```

adminpage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Page</title>
    <link rel="stylesheet" href="visual%20part/adminpage.css">
</head>
<body>
<div class="container">
    <div class="menu">
        <a href="menuasadmin.html">Меню</a>
        <a href="list%20of%20orders/orders.html">Перелік замовлень</a>
        <a href="login/index.html">Вихід</a>
    </div>
</div>
</body>
</html>
```

menuasadmin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Menu Management</title>
    <link rel="stylesheet" href="visual%20part/menuasadmin.css">
</head>
<body>
<div class="container">
    <a href="adminpage.html"><img
src="https://c.files.bbci.co.uk/9017/production/_105278863_gettyimages-855098134.jpg"
alt="Логотип" class="logo-img"></a>
    <div class="tabs">
        <a href="menu/viewmenu.html">Переглянути меню</a>
        <a href="menu/editmenu.html">Додати компонент меню</a>
    </div>
</div>
</body>
</html>
```

userpage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Page</title>
    <link rel="stylesheet" href="visual%20part/userpage.css">
</head>
```

```
<body>
<div class="container">
    <ul>
        <li><a href="menu/showmenu.html">Переглянути меню</a></li>
        <li><a href="adding%20an%20order/createorder.html">Створити
замовлення</a></li>
        <li><a href="login/index.html">Вихід</a></li>
    </ul>
</div>
</body>
</html>
```

editmenu.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Редагувати Меню</title>
    <link rel="stylesheet" href="../visual%20part/editmenu.css">
</head>
<body>
<div class="container">
    <h2>Введіть інформацію про страву, яку бажаєте додати до меню нашого
сервісу</h2>
    <form id="addItemForm" enctype="multipart/form-data">
        <div class="input-box">
            <label for="itemName">Страва (скільки становить 1 порція):</label>
            <input type="text" id="itemName" name="itemName" required>
        </div>
        <div class="input-box">
```

```
        <label for="itemPrice">Ціна в UAH:</label>
        <input type="text" id="itemPrice" name="itemPrice" required>
      </div>
      <div class="input-box">
        <label for="itemCategory">Виберіть категорію страви:</label>
        <select id="itemCategory" name="itemCategory">
          <option value="Сніданки">Сніданки</option>
          <option value="Супи та Бульйони">Супи та Бульйони</option>
          <option value="Салати">Салати</option>
          <option value="Основні страви">Основні страви</option>
          <option value="Піца">Піца</option>
          <option value="Снеки та закуски">Снеки та закуски</option>
          <option value="Десерти">Десерти</option>
          <option value="Напої">Напої</option>
          <option value="Алкогольні напої">Алкогольні напої</option>
          <option value="Суші-сети">Суші-сети</option>
        </select>
      </div>
      <div class="input-box">
        <label for="itemIngredients">Інгредієнти чи опис:</label>
        <textarea id="itemIngredients" name="itemIngredients" required></textarea>
      </div>
      <div class="input-box">
        <label for="itemPhoto">Додайте задля більшої привабливості з боку клієнтів
(формат JPG або PNG):</label>
        <input type="file" id="itemPhoto" name="itemPhoto" accept=".jpg, .jpeg, .png">
      </div>
      <button type="submit" class="btn">Додати до меню</button>
    </form>
</div>
```

```
<script src="menu.js"></script>
<script>
    document.getElementById('addItemForm').addEventListener('submit', function(event) {
        event.preventDefault();
        var itemName = document.getElementById('itemName').value;
        var itemPrice = document.getElementById('itemPrice').value;
        var itemCategory = document.getElementById('itemCategory').value;
        var itemIngredients = document.getElementById('itemIngredients').value;
        var itemPhoto = document.getElementById('itemPhoto').files[0];
        addItemToMenu(itemName, itemPrice, itemCategory, itemIngredients, itemPhoto);
        window.location.href = 'viewmenu.html';
    });
</script>
</body>
</html>
```

showmenu.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>View Menu</title>
    <link rel="stylesheet" type="text/css" href="../visual%20part/showmenu.css">
    <style>
        .menu-item {
            text-align: center;
        }
    </style>
</head>
```

```
<body>
<div class="container">
    <div class="menu-header">
        <a href="../userpage.html"><img
src="https://c.files.bbci.co.uk/9017/production/_105278863_gettyimages-855098134.jpg"
alt="Логотип" class="logo-img"></a>
        <h2>Меню</h2>
    </div>
    <div id="menuList" class="menu-list"></div>
</div>

<script src="menu.js"></script>
<script>
    function displayMenuItems() {
        let menuItems = JSON.parse(localStorage.getItem('menuItems')) || [];
        let menuList = document.getElementById('menuList');
        menuList.innerHTML = '';
        let categories = {};
        menuItems.forEach(item => {
            if (!categories[item.category]) {
                categories[item.category] = [];
            }
            categories[item.category].push(item);
        });
        Object.keys(categories).sort().forEach(category => {
            let categoryItems = categories[category];
            let categoryDiv = document.createElement('div');
            categoryDiv.classList.add('menu-category');
            let categoryHeader = document.createElement('h3');
            categoryHeader.textContent = category;
```

```
        categoryDiv.appendChild(categoryHeader);
        categoryItems.sort((a, b) => a.name.localeCompare(b.name)).forEach(item => {
            let itemBlock = document.createElement('div');
            itemBlock.classList.add('menu-item');
            let itemImage = document.createElement('img');
            itemImage.src = item.photo;
            itemImage.alt = item.name;
            itemBlock.appendChild(itemImage);
            let itemName = document.createElement('p');
            itemName.classList.add('item-name');
            itemName.textContent = item.name;
            itemBlock.appendChild(itemName);
            let itemPrice = document.createElement('p');
            itemPrice.classList.add('item-price');
            itemPrice.textContent = `Ціна: ${item.price} грн`;
            itemBlock.appendChild(itemPrice);
            let itemIngredients = document.createElement('p');
            itemIngredients.classList.add('item-ingredients');
            itemIngredients.textContent = `Інгредієнти: ${item.ingredients}`;
            itemBlock.appendChild(itemIngredients);
            categoryDiv.appendChild(itemBlock);
            let divider = document.createElement('hr');
            categoryDiv.appendChild(divider);
        });

        menuList.appendChild(categoryDiv);
    });
  }
  window.onload = displayMenuItems;
</script>
```

```
</body>
</html>
```

viewmenu.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>View Menu</title>
    <<link rel="stylesheet" href="../visual%20part/viewmenu.css">

</head>
<body>
<div class="container">
    <div class="menu-header">
        <a href="../adminpage.html"><img src="…="
alt="Меню" class="menu-logo"></a>
        <a href="editmenu.html" class="add-item-link">Додати нову страву</a>
    </div>
    <ul class="menu-list" id="menuList"></ul>
</div>

<script src="menu.js"></script>
<script>
    function deleteMenuItem(index) {
        let menuItems = JSON.parse(localStorage.getItem('menuItems')) || [];
        menuItems.splice(index, 1);
        localStorage.setItem('menuItems', JSON.stringify(menuItems));
        displayMenuItems();
    }
```

```
function displayMenuItems() {
    let menuItems = JSON.parse(localStorage.getItem('menuItems')) || [];
    let menuList = document.getElementById('menuList');
    menuList.innerHTML = '';
    let categories = {};
    menuItems.forEach(item => {
        if (!categories[item.category]) {
            categories[item.category] = [];
        }
        categories[item.category].push(item);
    });
    Object.keys(categories).forEach(category => {
        let categoryItems = categories[category];
        let categoryContainer = document.createElement('div');
        categoryContainer.classList.add('category');
        let categoryTitle = document.createElement('h3');
        categoryTitle.classList.add('category-title');
        categoryTitle.textContent = category;
        categoryContainer.appendChild(categoryTitle);
        menuList.appendChild(categoryContainer);
        categoryItems.forEach(item => {
            let listItem = document.createElement('li');
            listItem.classList.add('menu-item');
            let itemImage = document.createElement('img');
            itemImage.src = item.photo;
            itemImage.alt = item.name;
            listItem.appendChild(itemImage);
            let itemName = document.createElement('span');
            itemName.textContent = item.name;
            listItem.appendChild(itemName);
```

```
            let itemPrice = document.createElement('span');
            itemPrice.textContent = `Ціна: ${item.price} грн`;
            listItem.appendChild(itemPrice);
            let itemIngredients = document.createElement('span');
            itemIngredients.textContent = `Інгредієнти: ${item.ingredients}`;
            listItem.appendChild(itemIngredients);
            let deleteButton = document.createElement('button');
            deleteButton.textContent = 'Видалити';
            deleteButton.classList.add('delete-button');
            deleteButton.addEventListener('click', () =>
deleteMenuItem(menuItems.indexOf(item)));
            listItem.appendChild(deleteButton);
            categoryContainer.appendChild(listItem);
        });
    });
  }
  window.onload = displayMenuItems;
</script>
</body>
</html>
```

menu.js

```
function displayMenuItems() {
  let menuItems = JSON.parse(localStorage.getItem('menuItems')) || [];
  let menuList = document.getElementById('menuList');
  menuList.innerHTML = '';
  menuItems.forEach(item => {
    let listItem = document.createElement('li');
    listItem.textContent = `${item.name}: ${item.price}`;
    menuList.appendChild(listItem);
  });
```

```
}
function addItemToMenu(name, price, category, ingredients, photo) {
    let reader = new FileReader();
    reader.onload = function(event) {
        let photoDataUrl = event.target.result;
        let menuItem = { name: name, price: price, category: category, ingredients:
ingredients, photo: photoDataUrl };
        let menuItems = JSON.parse(localStorage.getItem('menuItems')) || [];
        menuItems.push(menuItem);
        localStorage.setItem('menuItems', JSON.stringify(menuItems));
    };
    reader.readAsDataURL(photo);
}
window.onload = displayMenuItems;
```

createorder.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Create Order</title>
    <link rel="stylesheet" href="../visual%20part/createorder.css">
</head>
<body>
<div class="container">
    <a href="../userpage.html"><img
src="https://c.files.bbci.co.uk/9017/production/_105278863_gettyimages-855098134.jpg"
alt="Логотип" class="logo-img"></a>
    <h2>Створення смачнючого перекусу</h2>
    <div id="menuList"></div>
```

```
    <label for="customerName">Як до Вас звертатися</label>
    <input type="text" id="customerName" name="customerName">
    <label for="customerPhone">Ваш телефон: +380</label>
    <input type="tel" id="customerPhone" name="customerPhone">
    <label for="customerAddress">Куди доставити ваше замовлення:</label>
    <input type="text" id="customerAddress" name="customerAddress">
    <button id="submitOrder">Оформити замовлення</button>
    <div id="orderSummary"></div>
</div>

<script src="order.js"></script>
</body>
</html>
```

order.js

```
window.onload = function() {
    let menuItems = JSON.parse(localStorage.getItem('menuItems')) || [];
    let menuList = document.getElementById('menuList');
    let orderSummary = document.getElementById('orderSummary');
    let groupedMenuItems = {};
    menuItems.forEach(item => {
        if (!groupedMenuItems[item.category]) {
            groupedMenuItems[item.category] = [];
        }
        groupedMenuItems[item.category].push(item);
    });
    Object.keys(groupedMenuItems).forEach(category => {
        let categoryItems = groupedMenuItems[category];
        categoryItems.sort((a, b) => a.name.localeCompare(b.name));
        let categoryHeader = document.createElement('h2');
        categoryHeader.textContent = category;
```

```
    menuList.appendChild(categoryHeader);
    categoryItems.forEach(item => {
        let menuItem = document.createElement('div');
        menuItem.classList.add('menuItem');
        let itemName = document.createElement('h3');
        itemName.textContent = item.name;
        menuItem.appendChild(itemName);
        let itemDescription = document.createElement('p');
        itemDescription.textContent = item.description;
        menuItem.appendChild(itemDescription);
        let itemIngredients = document.createElement('des');
        itemIngredients.textContent = `Інгредієнти: ${item.ingredients}`;
        menuItem.appendChild(itemIngredients);
        let itemImage = document.createElement('img');
        itemImage.src = item.photo;
        itemImage.alt = item.name;
        menuItem.appendChild(itemImage);
        let itemPriceText = item.price;
        let itemPrice = parseFloat(itemPriceText);
        if (itemPrice > 0) {
            let itemPriceDisplay = document.createElement('p');
            itemPriceDisplay.textContent = `Ціна: ${item.price} грн`;
            menuItem.appendChild(itemPriceDisplay);
        }
        let quantityInput = document.createElement('input');
        quantityInput.type = 'number';
        quantityInput.min = '0';
        quantityInput.value = '0';
        menuItem.appendChild(quantityInput);
```

```
        menuList.appendChild(menuItem);
    });
});
document.getElementById('submitOrder').addEventListener('click', submitOrder);
function submitOrder() {
    let orderedItems = [];
    let totalCost = 0;
    let itemCosts = {};
    let menuItems = document.querySelectorAll('.menuItem');
    menuItems.forEach(menuItem => {
        let itemName = menuItem.querySelector('h3').textContent;
        let itemPriceText = menuItem.querySelectorAll('p')[1].textContent.split(' ')[1];
        let itemPrice = parseFloat(itemPriceText);
        let quantity = parseInt(menuItem.querySelector('input').value);
        if (quantity > 0) {
            let itemTotalCost = quantity * itemPrice;
            if (!itemCosts[itemName]) {
                itemCosts[itemName] = 0;
            }
            itemCosts[itemName] += itemTotalCost;
            totalCost += itemTotalCost;
            orderedItems.push({ name: itemName, quantity: quantity });
        }
    });
    totalCost += 40;
    let allQuantitiesZero = orderedItems.every(item => item.quantity === 0);
    let customerName = document.getElementById('customerName').value;
    let customerPhone = document.getElementById('customerPhone').value;
    let customerAddress = document.getElementById('customerAddress').value;
    if (allQuantitiesZero || customerPhone.trim() === '' || customerAddress.trim() === '') {
```

```
            alert('Замовлення не може бути створене. Будь ласка, виберіть страви та введіть ваші дані перед оформленням замовлення.');
            return;
        }
        orderSummary.innerHTML = '';
        let customerInfo = document.createElement('p');
        customerInfo.textContent = `Замовив: ${customerName}, Телефон: +380${customerPhone}, Адреса доставки: ${customerAddress}`;
        orderSummary.appendChild(customerInfo);
        let orderList = document.createElement('ul');
        Object.keys(itemCosts).forEach(itemName => {
            let listItem = document.createElement('li');
            listItem.textContent = `${itemName}: ${itemCosts[itemName]} грн`;
            orderList.appendChild(listItem);
        });
        let deliveryItem = document.createElement('li');
        deliveryItem.textContent = `Доставка: 40 грн`;
        orderList.appendChild(deliveryItem);
        orderSummary.appendChild(orderList);
        let totalCostElement = document.createElement('p');
        totalCostElement.textContent = `Загальна вартість: ${totalCost} грн`;
        orderSummary.appendChild(totalCostElement);
        let orders = JSON.parse(localStorage.getItem('orders')) || [];
        orders.push({ name: customerName, phone: customerPhone, address: customerAddress, items: orderedItems, totalCost: totalCost, timestamp: Date.now() });
        localStorage.setItem('orders', JSON.stringify(orders));
    }
};
```

orders.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Orders</title>
    <link rel="stylesheet" href="../visual%20part/orders.css">
</head>
<body>
<div class="container">
    <div class="logo-container">
        <a href="../userpage.html"><img
src="https://c.files.bbci.co.uk/9017/production/_105278863_gettyimages-855098134.jpg"
alt="Логотип"></a>
    </div>
    <h2>Деталі поточних замовлень</h2>
    <div class="order-details" id="orderDetails"></div>
</div>

<script src="orders.js"></script>
</body>
</html>
```

orders.js

```
window.onload = function() {
    let orders = JSON.parse(localStorage.getItem('orders'));
    let orderDetails = document.getElementById('orderDetails');
    if (orders && orders.length > 0) {
        orders.forEach((orderData, index) => {
            let orderContainer = document.createElement('div');
            orderContainer.classList.add('order-container');
```

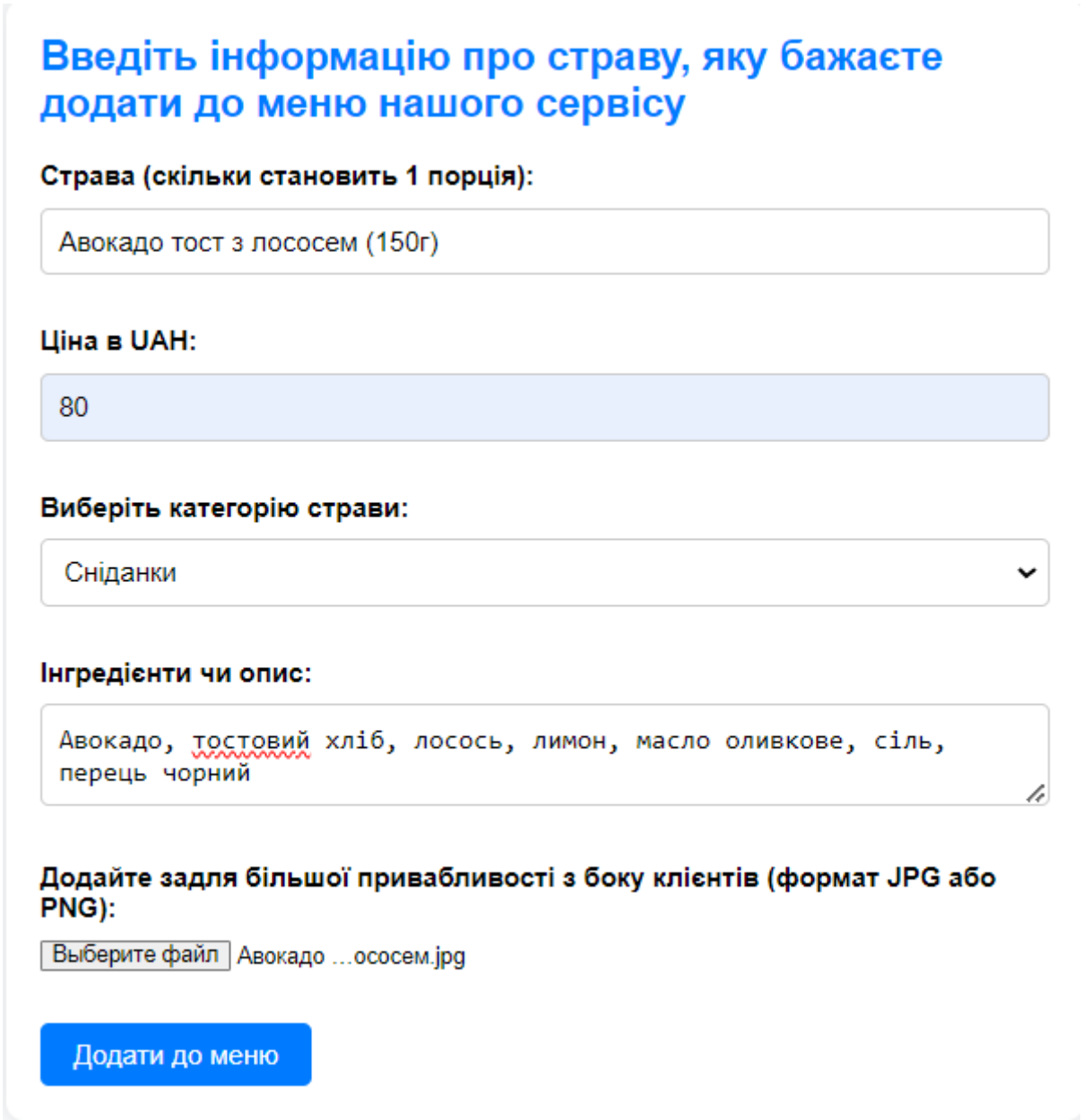
```
        let customerInfo = document.createElement('p');
        customerInfo.textContent = `Замовив: ${orderData.name}, Номер телефону:
+380${orderData.phone}, Адреса: ${orderData.address}, Дата: ${new
Date(orderData.timestamp).toLocaleString()}`;
        orderContainer.appendChild(customerInfo);
        let orderList = document.createElement('ul');
        orderData.items.forEach(item => {
            let listItem = document.createElement('li');
            listItem.textContent = `${item.name}: ${item.quantity}`;
            orderList.appendChild(listItem);
        });
        orderContainer.appendChild(orderList);
        let totalCostElement = document.createElement('p');
        totalCostElement.textContent = `Вартість за усе замовлення:
${orderData.totalCost} грн`;
        orderContainer.appendChild(totalCostElement);
        if (orderData.status === 'accepted') {
            orderContainer.classList.add('accepted');
        } else if (orderData.status === 'rejected') {
            orderContainer.classList.add('rejected');
        }
        let acceptButton = document.createElement('button');
        acceptButton.textContent = 'Прийняти';
        acceptButton.addEventListener('click', () => acceptOrder(index, orderContainer));
        orderContainer.appendChild(acceptButton);
        let rejectButton = document.createElement('button');
        rejectButton.textContent = 'Відхилити';
        rejectButton.addEventListener('click', () => rejectOrder(index, orderContainer));
        orderContainer.appendChild(rejectButton);
        let completeButton = document.createElement('button');
```

```
            completeButton.textContent = 'Виконано';
            completeButton.addEventListener('click', () => completeOrder(orderContainer));
            orderContainer.appendChild(completeButton);
            orderDetails.appendChild(orderContainer);
        });
    } else {
        let errorMessage = document.createElement('p');
        errorMessage.textContent = 'Наразі замовлень немає.';
        orderDetails.appendChild(errorMessage);
    }
};
function acceptOrder(index, orderContainer) {
    alert('Замовлення прийнято!');
    let orders = JSON.parse(localStorage.getItem('orders'));
    orders[index].status = 'accepted';
    localStorage.setItem('orders', JSON.stringify(orders));
    orderContainer.classList.add('accepted');
}
function rejectOrder(index, orderContainer) {
    let orders = JSON.parse(localStorage.getItem('orders'));
    orders.splice(index, 1);
    localStorage.setItem('orders', JSON.stringify(orders));
    orderContainer.classList.add('rejected');
    setTimeout(() => {
        orderContainer.style.display = 'none';
    }, 50);
}
function completeOrder(orderContainer) {
    orderContainer.classList.add('completed');
    setTimeout(() => {
```

```
        orderContainer.style.display = 'none';
    }, 800);
}
```

ДОДАТОК Б

Приклад



Тут ми створюємо нову страву від імені Administrator

Додати нову страву

СНІДАНКИ

Авокадо тост з лососем (150г)

Ціна: 80 грн

Інгредієнти: Авокадо, тостовий хліб, лосось, лимон, масло оливкове, сіль, перець чорний

Видалити

Саме так вона виглядає в меню від імені Адміністратора

Сніданки

Авокадо тост з лососем (150г)

Ціна: 80 грн

Інгредієнти: Авокадо, тостовий хліб, лосось, лимон, масло оливкове, сіль, перець чорний

А так виглядає в меню від імені дефолтного користувача

Фруктовий смузі (400 мл)

Інгредієнти: Банан, полуниця, ягоди, молоко або йогурт, мед або цукор

Ціна: 55 грн

3

Суші-сети

Філадельфія (24 ролів)

Інгредієнти: Рол "Філадельфія" (8 шт): лосось, авокадо, огірок, філадельфія сир, рис, норі, кунжут. Суші "Філадельфія" (6 шт): лосось, філадельфія сир, рис. Суші "Урамакі" (6 шт):

креветки, авокадо, огірок, рис, норі, кунжут. Суші "Нігірі" (4 шт): лосось, креветки, рис.

Ціна: 300 грн

1

Як до Вас звертатися

Ярослав

Ваш телефон: +380

84311831

Куди доставити ваше замовлення:

Шевченка, 88

Оформити замовлення

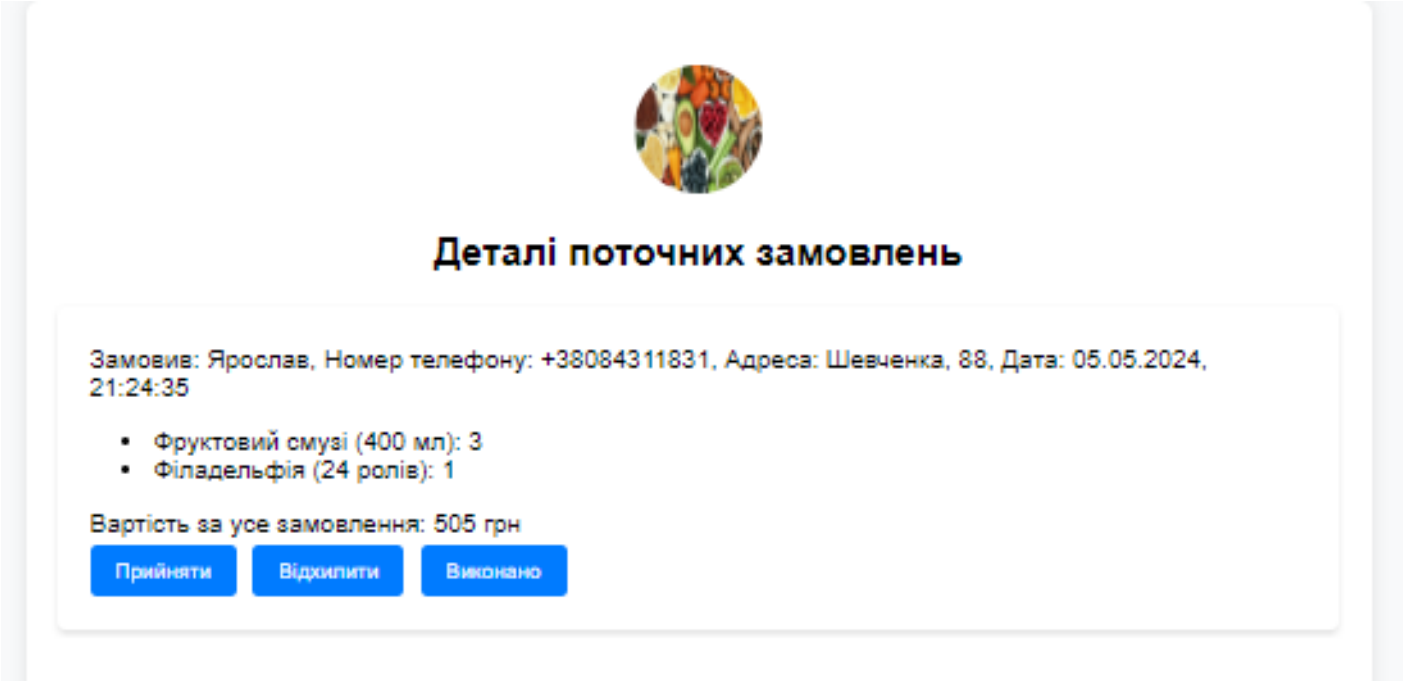
Таким чином ми створюмо замовлення, обираючи скільки страв якого виду беремо та які особисті дані вводимо

Замовив: Ярослав, Телефон: +38084311831, Адреса доставки: Шевченка, 88

- Фруктовий смузі (400 мл): 165 грн
- Філадельфія (24 ролів): 300 грн
- Доставка: 40 грн

Загальна вартість: 505 грн

А тут отримуємо чек



Таким чином виглядає сторінка з замовленнями, де є все необхідні дані: інформація про замовника, кількість замовлених товарів та прибуток.

Деталі поточних замовлень

Замовив: Ярослав, Номер телефону: +38084311831, Адреса: Шевченка, 88, Дата: 05.05.2024, 21:24:35

- Фруктовий смузі (400 мл): 3
- Філадельфія (24 ролів): 1

Вартість за усе замовлення: 505 грн

Прийняти Відхилити Виконано

Так буде виглядати замовлення, якщо ми його приймемо

Деталі поточних замовлень

Замовив: Ярослав, Номер телефону: +38084311831, Адреса: Шевченка, 88, Дата: 05.05.2024, 21:24:35

- Фруктовий смузі (400 мл): 3
- Філадельфія (24 ролів): 1

Вартість за усе замовлення: 505 грн

Прийняти Відхилити Виконано

А так якщо виконаємо. Майже одразу воно зникне

Деталі поточних замовлень

Наразі замовлень немає.

Після того як те замовлень зникне, наш список буде виглядати так.