Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Розробка веб-застосунку для автоматизації процесів ресторанного бізнесу

Текстова частина до курсової роботи

за спеціальністю 122 «Ком'ютерні Науки»

Керівник курсової роботи

ст.в. Салата К.В,

_____(підпис)

"___" _____ 2025 р.

Виконав студент КН-3

Рудник Б.Д.

"___" _____ 2025 р.

Київ 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
зав. кафедри інформатики,
доцент, к.ф-м.н.
Гороховський С.С.

_____
(підпис)
«_____» _____ 2025р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студену 3-го курсу факультету інформатики Руднику Богдану Дмитровичу

ТЕМА: Розробка веб-застосунку для автоматизації процесів ресторанного бізнесу

Зміст ТЧ до курсової роботи:

Перелік використаних термінів та скорочень

Анотація

Вступ

Аналіз предметної області та подібних рішень

Огляд використаних технологій та інструментів

Бізнес-вимоги та функціональність системи

Опис реалізації проєкту

Основні сторінки з демонстрацією інтерфейсу веб-застосунку

Висновки

Список використаних джерел

Дата видачі «___» _____ 2024 р. Керівник Салата К.В.

_____
(підпис)
Завдання отримав _____
(підпис)

Тема: Розробка веб-застосунку для автоматизації процесів ресторанного бізнесу

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання етапу	Примітки
1	Отримання завдання на курсову роботу.	вересень 2024	
2	Аналіз предметної області та постановка задачі.	жовтень 2024	
3	Вибір технологій і архітектури веб-застосунку.	листопад 2024	
4	Проектування структури бази даних і розробка основних модулів.	грудень 2024 - січень 2025	
5	Реалізація функціональності для користувачів та адміністраторів.	лютий - березень 2025	
6	Тестування веб-застосунку на наближених до реальних даних та виправлення помилок.	квітень 2025	
7	Написання текстової частини курсової роботи.	квітень - травень 2025	
8	Подання роботи на кафедру для перевірки на плагіат.	до _____ травня 2025	
9	Створення презентації для захисту роботи.	до _____ травня 2025	
10	Захист курсової роботи.	Згідно з розкладом роботи ЕК	

Студент _____

Керівник _____

“_____”_____2024 р

Зміст

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕРМІНІВ ТА СКОРОЧЕНЬ

- **SSR (Server-Side Rendering)** – рендеринг сторінки на сервері перед відправкою юзеру.
- **SSG (Static Site Generation)** – створення HTML-сторінок на етапі білду, а не при кожному запиті.
- **Білдити (to build)** – процес збірки та оптимізації коду перед розгортанням застосунку.
- **Імпорти** – включення зовнішніх модулів або файлів у програму через команду import.
- **ORM (Object-Relational Mapping)** – технологія для роботи з базами даних через об'єкти замість SQL-запитів.
- **CRUD (Create, Read, Update, Delete)** – основні операції для роботи з даними в базі даних.
- **БД (База даних, Database)** – організоване зберігання інформації, доступної для обробки та запитів.
- **Хешування (Hashing)** – перетворення даних у фіксовану послідовність символів для захисту.
- **Кешування (Caching)** – тимчасове збереження даних для прискореного доступу до них у майбутньому.
- **API (Application Programming Interface)** – інтерфейс, що дозволяє програмам взаємодіяти між собою.
- **Пагінація (Pagination)** – спосіб підвантаження даних частинами замість завантаження всього набору одразу.
- **HTTP-запити** – запити, які клієнт надсилає серверу для отримання або зміни даних.
- **JWT (JSON Web Token)** – стандарт для безпечного обміну інформацією у вигляді зашифрованого токена.

- **REST API (Representational State Transfer API)** – архітектурний стиль (набір правил) про те, як правильно писати серверну частину, щоб всі компоненти в системі легко обмінювалися даними. Ця структура дозволяє в подальшому масштабувати застосунок.
- **API-ендпоінти (API Endpoints)** – конкретні шляхи або URL-адреси, за якими доступні функції API.
- **Валідація** – перевірка даних на відповідність визначеним правилам або форматам.
- **Zod-схема** – структура правил для перевірки даних, створена за допомогою бібліотеки Zod.
- **Інпут (Input)** – елемент інтерфейсу для введення даних користувачем.
- **Футер (Footer)** – нижня частина вебсторінки.
- **Хедер (Header)** – верхня частина вебсторінки.

АНОТАЦІЯ

Метою цієї курсової роботи є розробка веб-застосунку для автоматизації процесів ресторанного бізнесу. У ході роботи було проведено аналіз вимог до систем автоматизації ресторанного обслуговування, сформульовано основні функціональні вимоги до вебсайту. Реалізовано основні модулі для користувачів та адміністраторів. Веб-застосунок розроблено з використанням сучасних технологій у сфері веб-програмування: Next.js, React, Prisma, Supabase, Cloudinary та низки супутніх бібліотек.

Ключові слова: веб-застосунок, вебсайт, автоматизація ресторану, Next.js, React, Prisma, Supabase, Cloudinary, TailwindCSS, Material UI, React Query, NextAuth, Zod, Axios, ESlint.

ВСТУП

Під час всього свого існування, галузь ресторанного бізнесу неодмінно стикається з викликами, що пов’язані з ефективністю внутрішніх процесів та безпосередньої взаємодії з користувачами. Сьогодення вимагає не тільки високого рівня матеріальних аспектів (комфортабельністю закладу, смак їжі та напоїв…), а й швидкого реагування на запити клієнтам, виконання яких має задовольнити ці потреби та одночасно формувати лояльність і позитивне враження від ресторану. Саме тому розробка та впровадження новітніх рішень задля автоматизації процесів, дозволяє закладам ресторанного бізнесу бути конкурентоспроможними на цьому ринку.

Загальний розвиток технологій призводить до того, що все більше сфер переходять на рішення, які пов’язані з використанням смартфону чи браузера. Саме тому ресторани, у яких відсутні рішення на таких платформах та значна частина операцій все ще пов’язана з ручними діями, як от бронювання столику через дзвінок по телефону, поступаються тим, у яких це вже впроваджено.

Саме тому було вирішено розробити веб-застосунок, який дозволить клієнтам зручно переглядати основну інформацію про ресторан, меню, відгуки інших користувачів та надати їм можливість самостійно створювати бронювання на потрібну для них дату з часом. А персоналу закладів ресторанного бізнесу система забезпечить ефективне управління внутрішніми процесами для покращення загальної ефективності та якості.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОДІБНИХ РІШЕНЬ

1.1 Аналіз предметної області

Сфера ресторанного бізнесу вимагає постійного вдосконалення процесів обслуговування клієнтів та управління закладом. Швидкість обробки замовлень, зручність комунікації з клієнтами, точність бронювання столиків і оперативність оновлення інформації про страви – є важливими факторами у цій галузі. Використання інформаційних технологій дозволяє оптимізувати ці процеси, підвищити рівень сервісу та скоротити витрати ресурсів.

Практична значущість автоматизації ресторанних процесів підтверджується прикладом успішних впроваджень таких систем. Зокрема, як зазначено на сайті «SmartCafe» [1], впровадження цифрового управління меню та процесами бронювання дає змогу значно підвищити ефективність роботи закладів харчування. Електронне управління меню спрощує процес оновлення інформації про страви, миттєво відображаючи зміни для персоналу та клієнтів через інтеграцію з онлайн-ресурсами. Автоматизоване бронювання столиків через веб-застосунок дозволяє клієнтам самостійно здійснювати резервацію, що зменшує навантаження на адміністраторів, мінімізує кількість помилок і запобігає дублюванню записів.

Практичний досвід використання таких рішень свідчить про значні переваги: використання мобільного застосунку з інтеграцією онлайн-меню та системою бронювання здатне підвищити обсяг продажів на 40% і збільшити середній чек на 15% [1]. Це свідчить про безпосередній вплив автоматизації на фінансові показники закладів та формування лояльної клієнтської бази.

Таким чином, впровадження комплексних IT-рішень для автоматизації процесів ресторанного бізнесу є актуальним напрямом розвитку сфери громадського харчування.

1.2 **Аналіз подібних рішень**

Для аналізу сучасних підходів до автоматизації ресторанного бізнесу було розглянуто декілька наявних платформ і сервісів, що пропонують онлайн-бронювання, перегляд меню та взаємодію з клієнтами.

1.2.1 «Reston.ua» [2]

Платформа «Reston.ua» є великим каталогом ресторанів України, що дозволяє користувачам здійснювати пошук закладів за різними критеріями, переглядати базову інформацію про ресторани, залишати відгуки та здійснювати онлайн-бронювання столиків. Авторизувавшись, є можливість керувати своїми бронюваннями.

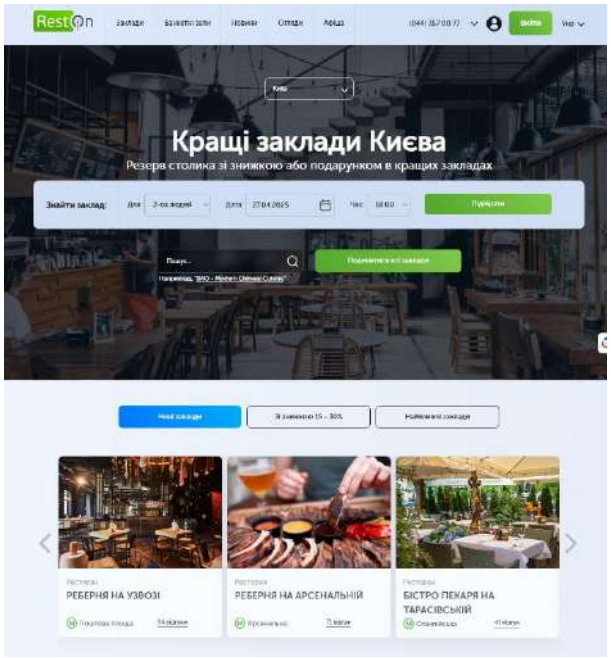

Рисунок 1.1 – Інтерфейс головної сторінки «Reston.ua»

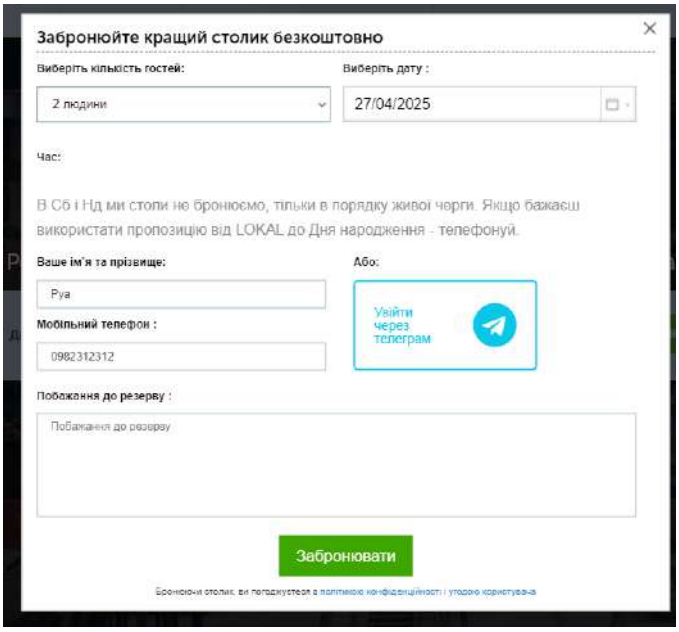

Рисунок 1.2 – Інтерфейс бронювання столику «Reston.ua»

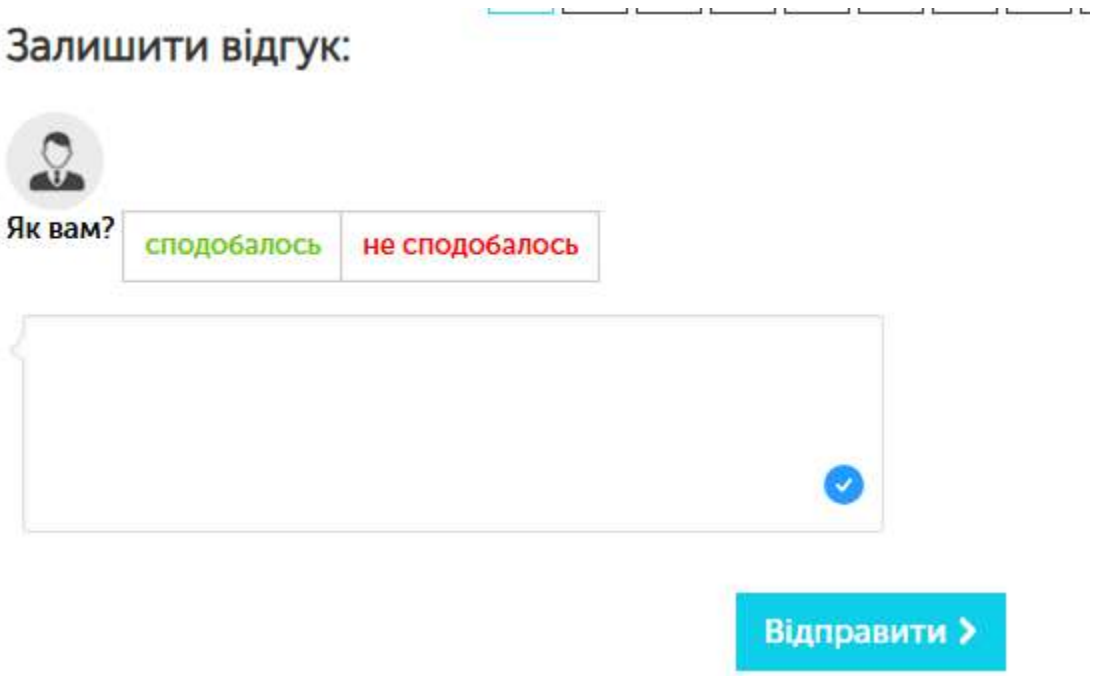

Рисунок 1.3 – Інтерфейс залишення відгуку «Reston.ua»

1.2.2 «Famiglia (Vero Vero)» [3]

«Famiglia Vero Vero» – це сайт окремого ресторану, що є частиною мережі ресторанів «Famiglia». Сайт дозволяє переглядати меню у вигляді PDF-документа та основну інформацію про ресторан. Можливість онлайн-бронювання реалізована через просту форму і має обмеження в 4-х відвідувачів.

До того ж, функціоналу для скасування бронювання або повноцінної взаємодії через профіль користувача не передбачено.

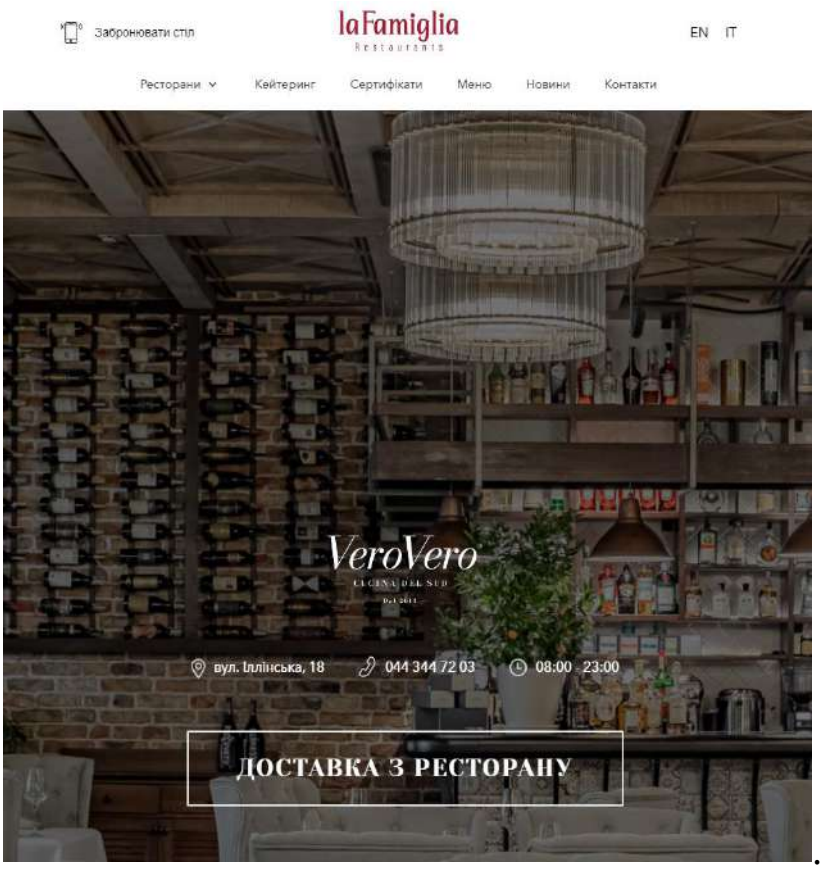

Рисунок 1.4 – Інтерфейс головної сторінки «Famiglia (Vero Vero)»

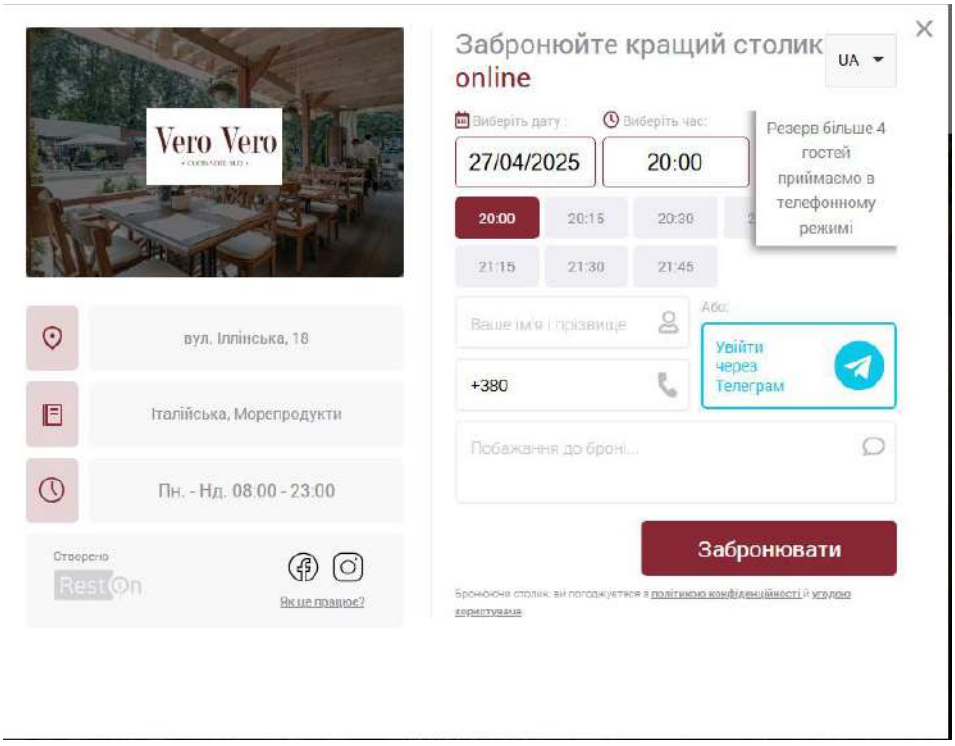

Рисунок 1.5 – Інтерфейс бронювання столику «Famiglia (Vero Vero)»

1.2.3 «Avalon» [4]

Платформа «Avalon» використовує рішення «ChoiceQR» [5] для автоматизації процесів замовлень і бронювання столиків. Сервіс дозволяє користувачам переглядати меню в інтерактивному форматі, авторизуватися, здійснювати бронювання та скасовувати їх у разі потреби. Інтерфейс орієнтований на простоту і швидкість взаємодії користувача з рестораном.

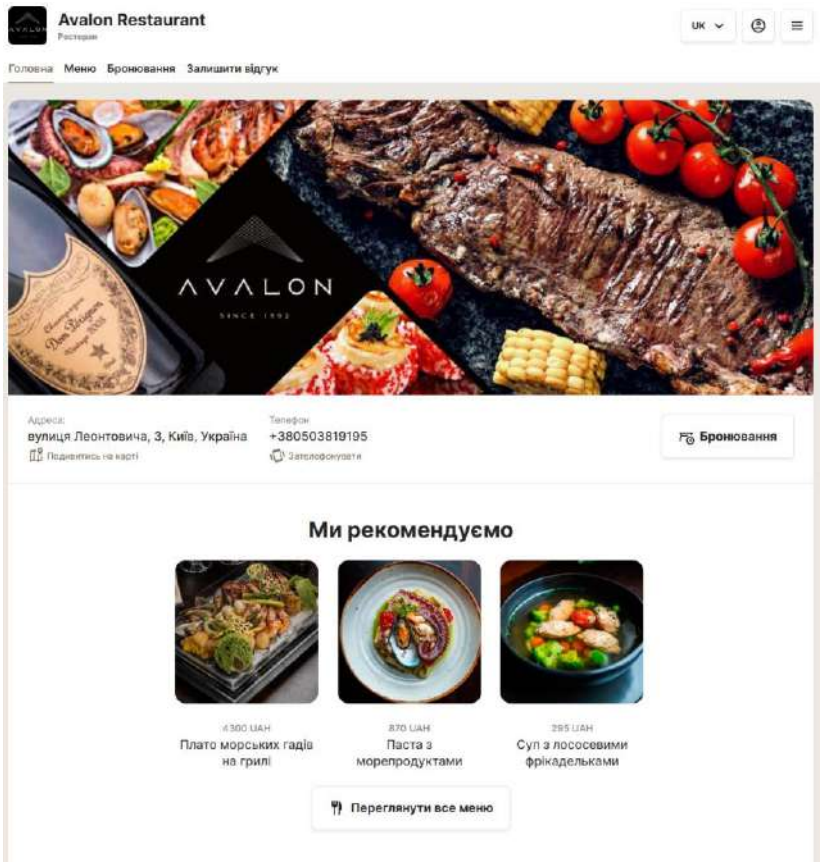

Рисунок 1.6 – Інтерфейс головної сторінки «Avalon»

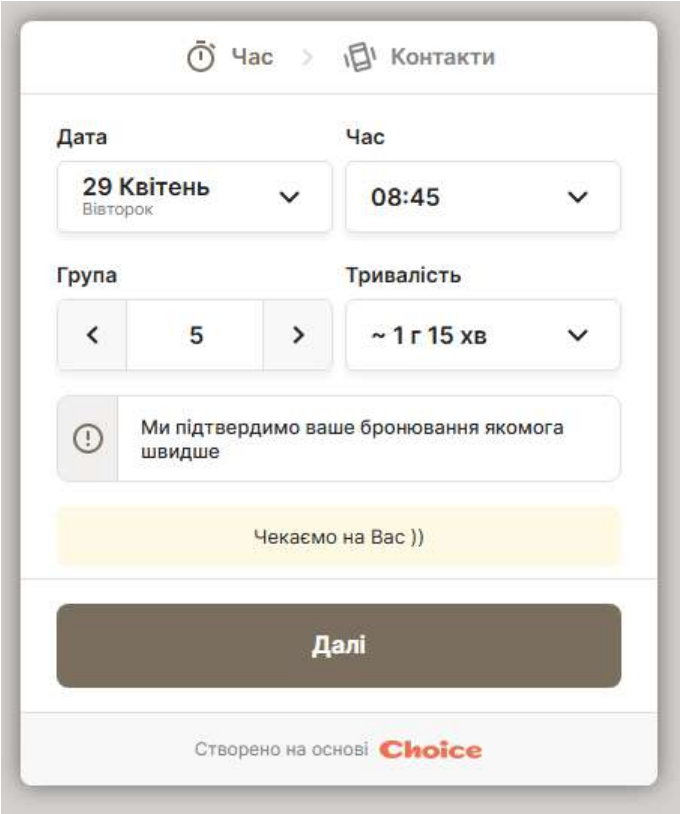

Рисунок 1.7 – Інтерфейс бронювання столику «Avalon»

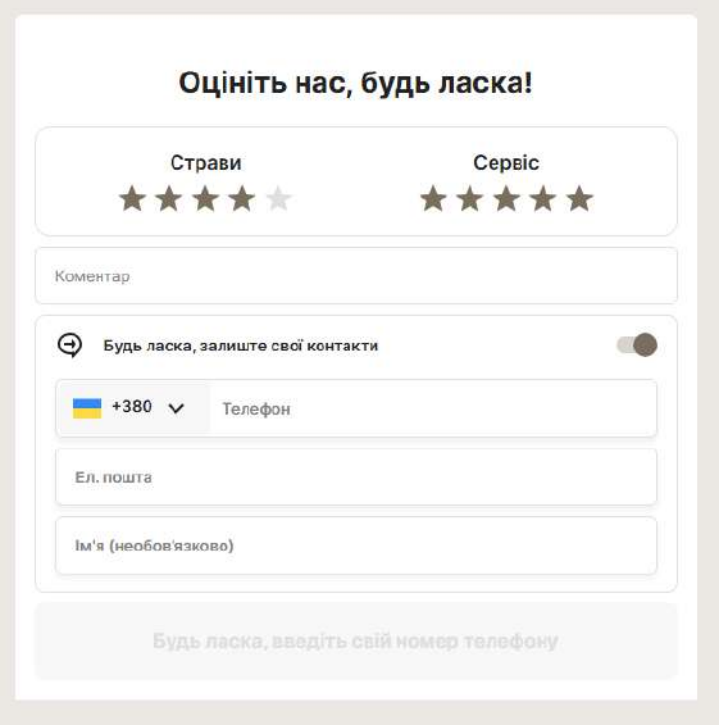

Рисунок 1.8 – Інтерфейс залишення відгуку «Avalon»

1.2.4 Порівняльна таблиця основних можливостей схожих рішень

Таблиця 1.1 – Порівняльна таблиця з альтернативними рішеннями

Функціонал	**Курсовий проєкт**	**Reston.ua [2]**	**Famiglia (Vero Vero) [3]**	**Avalon [4]**
Авторизація	+	+	–	+
Редагування профілю	+	+	-	+
Створення бронювання	+	+	+ (до 4х осіб)	+
Скасування бронювання	+	+	–	+
Перегляд своїх резервацій у профілі.	+	+	-	+
Перегляд меню	+	–	+ (через PDF)	+
Залишення відгуку	+	+	+	+
Перегляд наявних відгуків	+	+	–	–

Як можна побачити з таблиці, не всі наявні рішення забезпечують повний спектр функціональності, зокрема щодо можливості перегляду відгуків або зручного доступу до меню безпосередньо у веб-застосунку. Пропонований у курсовій роботі веб-застосунок поєднує всі ключові функції, що дозволяє підвищити зручність взаємодії користувачів з рестораном.

1.3 Висновки до розділу 1

Отже, у даному розділі було проаналізовано предметну область автоматизації процесів ресторанного бізнесу та обґрунтовано актуальність розробки веб-застосунку для підвищення ефективності обслуговування клієнтів і управління закладом. Було досліджено приклад практичного впровадження цифрових рішень на основі платформи «SmartCafe», що засвідчило їхню позитивну дію на фінансові показники та задоволеність клієнтів.

Крім того, проведено аналіз існуючих рішень, таких як платформа «Reston.ua», сайт ресторану «Famiglia (Vero Vero)» та «Avalon» на базі ChoiceQR. Описано функціональні можливості кожного із зазначених сервісів, проаналізовано їх сильні та слабкі сторони.

Порівняльний аналіз показав, що не всі рішення пропонують повний спектр функціональності для зручної взаємодії користувачів із рестораном, зокрема можливість перегляду меню у веб-інтерфейсі або доступ до історії бронювань. Власний веб-застосунок, що розробляється у межах курсової роботи, враховує ці недоліки та об’єднує основні функції в одному рішенні.

До того ж, на основі проведеного аналізу були сформульовані основні завдання для подальшої розробки системи.

2 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ

Для створення системи автоматизації процесів ресторанного бізнесу було обрано веб-застосунок, а не мобільний чи настільний додаток, з таких причин:

- **Доступність з будь-якого пристрою**: веб-застосунок можна відкрити через Інтернет на будь-якому пристрої з браузером, не встановлюючи додаткове програмне забезпечення.
- **Кросплатформність**: веб-застосунок не залежить від операційних систем (наприклад, Android, iOS, Windows, Ubuntu...), що розширює коло потенційних користувачів.
- **Економія часу та ресурсів:** розробка веб-застосунку вимагає менше зусиль і витрат, ніж створення кількох нативних додатків для різних платформ.
- **Зручність оновлень:** усі зміни та вдосконалення веб-застосунку вносяться централізовано, без потреби оновлювати програми на пристроях користувачів.

Для реалізації проєкту використано сучасний набір технологій: Next.js, TypeScript, Supabase, Cloudinary, WebStorm, а також низку додаткових бібліотек та інструментів.

2.1 Next.js [7]

Основною технологією для реалізації курсової роботи виступив Next.js — популярний фреймворк для React [8], що забезпечує серверний рендеринг (SSR), статичну генерацію сторінок (SSG) і має широку підтримку для побудови SEO-оптимізованих вебсайтів. Використання Next.js дозволяє реалізувати високу продуктивність застосунку завдяки можливості комбінувати рендеринг на сервері (SSR) і на клієнті. Також, однією з ключових переваг Next.js над React полягає в тому, що не потрібно робити окремо бек-енд та фронт-енд, оскільки все можна реалізувати зручно в одному проєкті [9].

У розробці застосунку використано версію **Next.js 15.2.4**, що підтримує найновіші можливості фреймворку.

2.2 TypeScript [10]

TypeScript – це мова програмування, що розширює JavaScript завдяки введенню статичної типізації. Використання типізованої мови дозволяє зменшити кількість помилок під час розробки, полегшити масштабування проєкту та підвищити надійність та якість коду.

2.3 Supabase [11]

Supabase – це безкоштовний хмарний сервіс, що надає повноцінну серверну інфраструктуру на базі PostgreSQL.

Платформа забезпечує просту інтеграцію з Next.js, що дозволило не витрачати час на налаштування власного серверного середовища.

2.4 Cloudinary [12]

Cloudinary – це хмарна платформа для управління, зберігання, оптимізації та отримання медіаконтенту за URL. Використання цього сервісу дозволяє мінімізувати навантаження на основний сервер і забезпечити швидке завантаження фотографій.

Однією з ключових переваг Cloudinary є наявність безкоштовного тарифного плану, який цілком задовольнив потреби реалізованого застосунку. Платформа відрізняється детальною та зрозумілою документацією, що містить численні приклади інтеграції різними мовами програмування, що значно спрощує процес впровадження сервісу в проєкт.

2.5 WebStorm [13]

WebStorm – це інтегроване середовище розробки (IDE) від компанії «JetBrains», оптимізоване для JavaScript, TypeScript тощо та сучасних фреймворків, зокрема React і Next.js.

Такі функції, як автодоповнення коду, перевірка синтаксису в реальному часі та інші, значно підвищили ефективність розробки.

2.6 Використання ESLint для підтримки якості коду [14]

Для забезпечення якості коду під час розробки вебзастосунку було використано інструмент ESLint. Він здійснює статичний аналіз коду, виявляє синтаксичні помилки, порушення стилю та небезпечні практики програмування.

У проєкті використовувався плагін @typescript-eslint, який дозволяє проводити перевірку коду, написаного на TypeScript.

Це розширення забороняло білдити проєкт, поки не будуть вирішені всі помилки, що пов’язанні з типізацією, не використаними імпортами тощо.

На рисунку 2.1 наведено приклад помилок, виявлених ESLint під час перевірки коду:

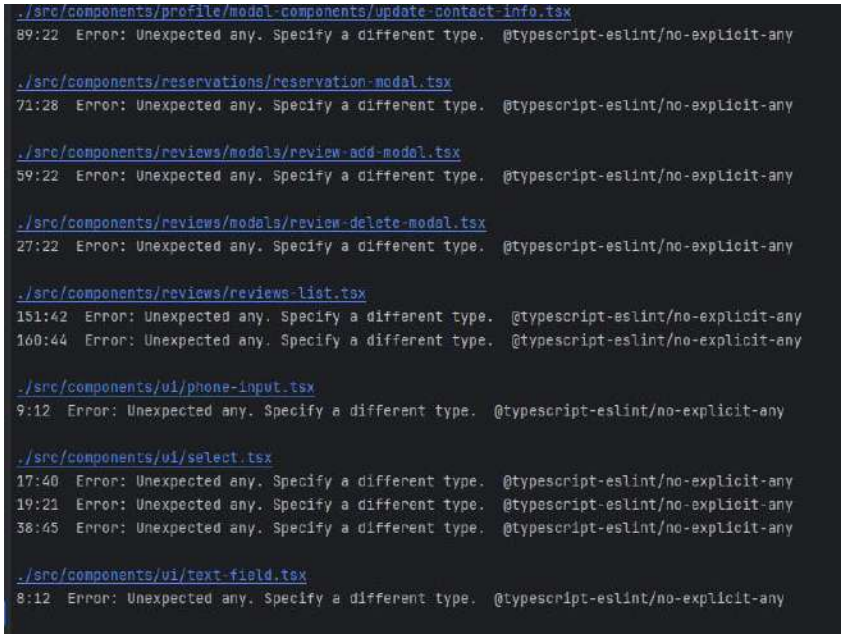

Рисунок 2.1 – Приклад повідомлень ESLint про виявлені помилки типізації

2.6 Огляд використаних бібліотек

У проєкті також використано низку додаткових бібліотек та фреймворків, що забезпечили необхідний функціонал.

2.6.1 Prisma ORM [15]

Prisma – це сучасна ORM (Object-Relational Mapping) для роботи з базами даних, яка використовується у проєкті для взаємодії з Supabase. Prisma спрощує створення запитів до бази даних, автоматизує генерацію моделей і забезпечує типізацію даних завдяки інтеграції з TypeScript.

Завдяки цій бібліотеці не потрібно писати SQL-запити вручну. Натомість просто використовувати функції, які покривають всі необхідні операції з базою даних, такі як створення, читання, оновлення та видалення записів (CRUD).

Додатковою перевагою Prisma є вбудовані міграції бази даних, що дозволяють відстежувати та контролювати зміни в структурі даних під час розробки. І якщо щось зламається, то є можливість повернутися до минулої версії БД. До того ж, завдяки строгій типізації, працюючи з Prisma в IDE видно підказки коду та помилки на етапі компіляції, що значно підвищує надійність та якість коду.

2.6.2 NextAuth.js [16]

NextAuth.js – це бібліотека для реалізації автентифікації в застосунках на основі Next.js. У проєкті вона забезпечує зручну інтеграцію авторизації через сторонні провайдери. За допомогою цієї бібліотеки у проєкті було реалізовано можливість входити в систему за допомогою Google-акаунтів. NextAuth.js підтримує безпечне управління сесіями, JWT-токени та кастомізацію процесу автентифікації.

2.6.3 MUI (Material UI) [17] та TailwindCSS [18]

Material UI (MUI) – одна з найпопулярніших бібліотек для створення інтерфейсів користувача на базі React. Вона надає велику кількість готових

компонентів, що дозволяють швидко розробляти адаптивні, естетично привабливі веб-інтерфейси.

Використовуючи цю бібліотеку, було побудовано основні компоненти інтерфейсу: кнопоки, текстові поля, модальні вікна тощо.

TailwindCSS – це утилітарний CSS-фреймворк, який дозволяє будувати інтерфейси без написання власних CSS-файлів завдяки використанню готових класів безпосередньо в HTML/JSX-розмітці.

Поєднання MUI та TailwindCSS дозволило досягти балансу між використанням потужних готових компонентів і гнучкою розробкою унікальних стилістичних рішень, що суттєво прискорило процес створення адаптивного та зручного для користувачів інтерфейсу.

2.6.4 Інші бібліотеки

Нижче наведено таблицю з іншими бібліотеками, які використовувалися в проєкті, із зазначенням їхньої функціональності:

Таблиця 2.1 – Опис інших бібліотек використаних в проєкті

Назва бібліотеки	За що відповідає
material-react-table	Створення таблиць із розширеним функціоналом (сортування, фільтрація, пагінація) на базі Material UI.
bcryptjs	Хешування паролів користувачів для безпечного зберігання в базі даних.
React Hook Form + Zod	Створення форм із підтримкою валідації введених даних, інтеграція з TypeScript.
@tanstack/react-query	Кешування та оптимізація запитів до API, управління станом завантаження даних.
axios	Виконання HTTP-запитів до серверної частини або сторонніх сервісів.

framer-motion	Додавання плавних анімаційних ефектів до елементів інтерфейсу.
date-fns	Робота з датами: форматування, обчислення різниці між датами тощо.
swiper	Створення інтерактивних слайдерів і «каруселей» для відображення контенту.
react-hot-toast	Відображення сповіщень і повідомлень користувачам у зручному форматі.

2.7 Висновки до розділу 2

У цьому розділі було обґрунтовано вибір веб-формату застосунку для автоматизації процесів ресторанного бізнесу та описано використані технології. Застосування сучасного стеку (Next.js, TypeScript, Supabase, Cloudinary, Prisma, NextAuth.js, MUI, TailwindCSS) дозволило реалізувати надійний, адаптивний та функціонально-достатній вебзастосунок. А використання ESLint сприяло підтримці високої якості коду.

3 БІЗНЕС-ВИМОГИ ТА ФУНКЦІОНАЛЬНІСТЬ СИСТЕМИ

3.1 Бізнес-вимоги

У межах розробки веб-застосунку для автоматизації процесів ресторанного бізнесу було визначено основні бізнес-вимоги, які система повинна задовольняти:

- Забезпечення онлайн-доступу до інформації про мережу ресторанів та їх меню.
- Надання користувачам можливості самостійного бронювання столиків через веб-інтерфейс.
- Реалізація багаторівневої системи доступу залежно від ролі користувача.
- Автоматизація управління резервaціями та процесами обслуговування закладів.
- Підвищення якості взаємодії із клієнтами через можливість залишати та переглядати відгуки.
- Забезпечення зручного адміністрування ресторанів, меню, категорій, столиків.
- Підтримка безпечної аутентифікації та управління обліковими записами користувачів.
- Спрощення процесу адміністрування системи через інтерфейс для менеджерів та адміністраторів.

3.2 Основні ролі користувачів

У системі передбачено такі ролі:

- **Guest** (неавторизований користувач)
- **User** (авторизований користувач)
- **Manager** (менеджер ресторану)
- **Admin** (адміністратор системи)

3.3 Функціональні вимоги

Функціональні вимоги визначають, що система повинна робити [19], відповідно до ролей користувачів:

Guest може:

- Зареєструватися.
- Авторизуватися.
- Переглядати інформацію про ресторани та меню.
- Переглядати відгуки інших користувачів.
- Бронювати столик.

User може все те, що і Guest, а також:

- Переглядати свої бронювання (активні та минулі).
- Редагувати контактну інформацію у власних бронюваннях.
- Скасовувати власні майбутні бронювання.
- Редагувати власний профіль (пароль, ім'я, прізвище, телефон).
- Додавати, редагувати та видаляти власні відгуки.
- Видалити свій акаунт.

Manager може все те, що і User, а також:

- Переглядати, редагувати, скасовувати та завершувати всі резервації.
- Додавати, редагувати та видаляти страви та категорії меню.
- Додавати, редагувати та видаляти міста, адреси та столики.
- Переглядати, видаляти користувачів із роллю User.
- Видаляти або редагувати відгуки будь-яких користувачів.
- Отримувати вільні столики на певний час.

Admin може все те, що і Manager, а також:

- Видаляти облікові записи користувачів із ролями User і Manager.

- Призначати роль Manager користувачам із роллю User.
- Переглядати повний список усіх зареєстрованих користувачів.

3.4 Нефункціональні вимоги

До нефункціональних вимог, що висуваються до системи, належать:

- **Доступність:** Веб-застосунок має бути доступним цілодобово (24/7) з будь-якого пристрою через Інтернет.
- **Безпека:** Забезпечення захисту персональних даних користувачів, шифрування паролів та безпечної аутентифікації.
- **Масштабованість:** Можливість розширення системи при зростанні кількості користувачів або закладів.
- **Продуктивність:** Час завантаження сторінок має бути мінімальним (менше 2 секунд при нормальному навантаженні).
- **Юзабіліті:** Інтерфейс має бути інтуїтивно зрозумілим для користувачів без необхідності проходити спеціальне навчання.
- **Підтримка мобільних пристроїв:** Адаптивний дизайн для роботи на смартфонах і планшетах.

3.5 Висновки до розділу 3

У цьому розділі було визначено бізнес-вимоги до розроблюваного веб-застосунку та сформульовано функціональні й нефункціональні вимоги до його реалізації. Описано основні ролі користувачів системи та їх доступні можливості.

Формулювання вимог забезпечує чітке розуміння архітектури системи, необхідного функціоналу та критеріїв якості, яким має відповідати розроблений продукт.

4 ОПИС РЕАЛІЗАЦІЇ ПРОЄКТУ

4.1 Загальна структура проєкту

Файлова структура веб-застосунку побудована відповідно до стандартів розробки застосунків на базі Next.js та TypeScript [21] [22]. Основними елементами структури є такі каталоги та конфігураційні файли:

Кореневий каталог проєкту:

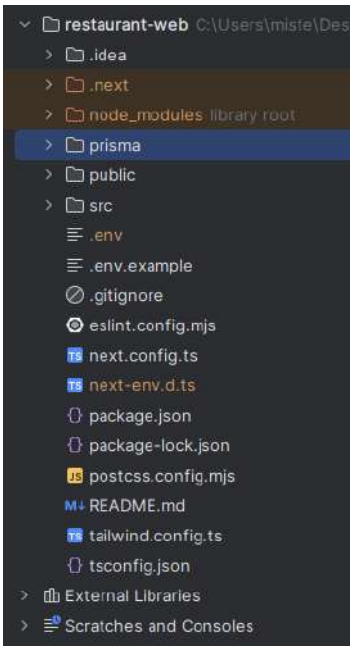

Рисунок 4.1 – Скриншот корневого каталогу проєкту

- **.env, .env.example** – файли змінних середовища, що містять ключову інформацію для підключення до бази даних, автентифікації через Google, роботи із Cloudinary та API-роутами.
  - DATABASE_URL – підключення до бази даних PostgreSQL.
  - NEXTAUTH_SECRET, AUTH_GOOGLE_ID, AUTH_GOOGLE_SECRET – змінні для налаштування автентифікації через NextAuth.js.
  - NEXT_PUBLIC_API_URL – публічна змінна для доступу до API ендпоінтів.
  - CLOUD_NAME, CLOUD_API_KEY, CLOUD_API_SECRET – облікові дані для роботи з сервісом Cloudinary.

- **package.json** – файл конфігурації залежностей та скриптів проєкту.
- **tsconfig.json** – налаштування компілятора TypeScript.
- **tailwind.config.ts**, **postcss.config.mjs** – файли конфігурації для роботи з TailwindCSS.
- **eslint.config.mjs** – конфігурація ESLint для підтримки якості коду.

Ключові каталоги:

- **/prisma –** містить файли для налаштування бази даних:
  - **schema.prisma** - файл, у якому визначено структуру бази даних (таблиці, зв'язки, типи даних).
  - Файл скрипту (**seed.ts**) для початкового заповнення бази тестовими даними.
- **/public –** містить статичні зображення для головної сторінки, які можуть бути доступні напряму через сервер.
- **/src-** головна директорія вихідного коду проєкту. Її структура буде детальніше описана у наступних підрозділах.
- **/.next**, **/node_modules –** системні каталоги, які створюються автоматично під час зборки або встановлення залежностей проєкту.

4.1.1 Структура папки /src

Основний код застосунку розташований у директорії **/src**.

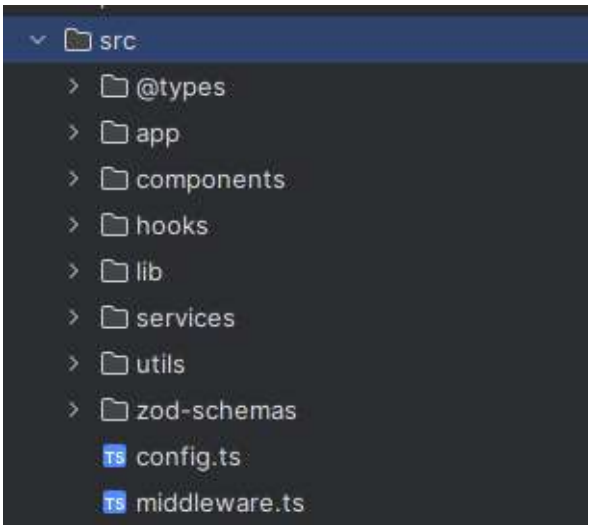

Рисунок 4.2 – Скриншот папки /src

Основні підкаталоги та їх призначення:

- **/src/@types –** папка для зберігання глобальних **типів TypeScript**, які використовуються в усьому проєкті.
- **/src/app –** корнева папка для файлової системи маршрутизації в Next.js, де структура директорій визначає URL-маршрути застосунку.
- **/src/components –** папка з основними візуальними компонентами. Містить як базові елементи (**/ui**) (кнопки, інпут-поля, селектори), так і складніші компоненти (наприклад, компоненти для списку резервацій, меню або відгуків). Компоненти згруповані за логікою їх використання.
- **/src/hooks –** папка для зберігання кастомних React-хуків.
- **/src/lib –** допоміжні функції, утиліти або налаштування для сторонніх бібліотек. Тут налаштовано інтеграції з NextAuth.js, ініціалізація Prisma-клієнта та створений глобальний axiosInstance.
- **/src/services –** папка для **роботи з API-запитами**. Містить функції для взаємодії із сервером через HTTP (Axios-запити). Логіка звернення до API винесена у окремі сервіси.

- **/src/utils** – допоміжні функції для роботи з датами, форматуванням, перевіркою прав доступу тощо.
- **/src/zod-schemas** – схеми валідації даних, побудовані на базі бібліотеки **Zod**. Визначають правила валідації для всіх форм, де потрібно щось заповнити користувачу. Забезпечують перевірку правильності введених даних до надсилання їх на сервер.

Файли у корені /src:

- **config.ts** – глобальні налаштування застосунку. Тут визначена назва сайту: export const SITE_NAME = "Dineva";
- **middleware.ts** – у проєкті реалізує обробку запитів на рівні маршрутизації перед передачею їх у систему. Він забезпечує захист маршрутов і контроль доступу до сторінок та API залежно від ролі користувача та статусу авторизації.

Каталог app/api/:

У цій папці реалізовані маршрути API, що обробляють запити клієнтів через HTTP-методи (GET, POST, PATCH, PUT, DELETE)

Рисунок 4.3 – Скриншот папки /src/app/api

Для безпеки деякі маршрути винесені окремо у папку protected/. Для обробки запитів в цій папці, потрібні підвищені права доступу (Manager, Admin).

4.2 Схема бази даних

База даних веб-застосунку побудована на основі Supabase (PostgreSQL) із використанням Prisma ORM. Структура бази даних організована відповідно до основних бізнес-процесів системи: керування користувачами, ресторанами, меню, бронюваннями та відгуками.

Між сутностями встановлені зв'язки, що забезпечують цілісність та логічну взаємодію даних у системі. Таблиця 4.1 – Таблиці застосунку в базі даних

Назва таблиці	Опис
users	Користувачі системи
accounts	Зовнішні акаунти (Google)
sessions	Сесії авторизації користувачів
cities	Міста
addresses	Адреси ресторанів
restaurants	Ресторани
tables	Столики в ресторанах
menu_items	Страви в меню
categories	Категорії страв
reservations	Бронювання столиків
reviews	Відгуки користувачів

Таблиця 4.2 – Зв’язки між таблицями

Таблиця 1	Таблиця 2	Тип зв'язку	Опис
users	reservations	1:N	Користувач може мати багато бронювань
users	reviews	1:N	Користувач може залишити багато відгуків

users	accounts	1:N	Один користувач може мати кілька акаунтів (Google, Facebook…)
users	sessions	1:N	Один користувач може мати кілька сесій авторизації
users	reservations (completedReservations)	1:N	Користувач може завершувати багато бронювань
users	reservations (cancelledReservations)	1:N	Користувач може скасовувати багато бронювань
cities	addresses	1:N	Місто має багато адрес
addresses	restaurants	1:1	Кожна адреса належить одному ресторану
restaurants	tables	1:N	Ресторан має багато столиків
tables	reservations	1:N	Столик має багато бронювань
categories	menu_items	1:N	Категорія містить багато страв

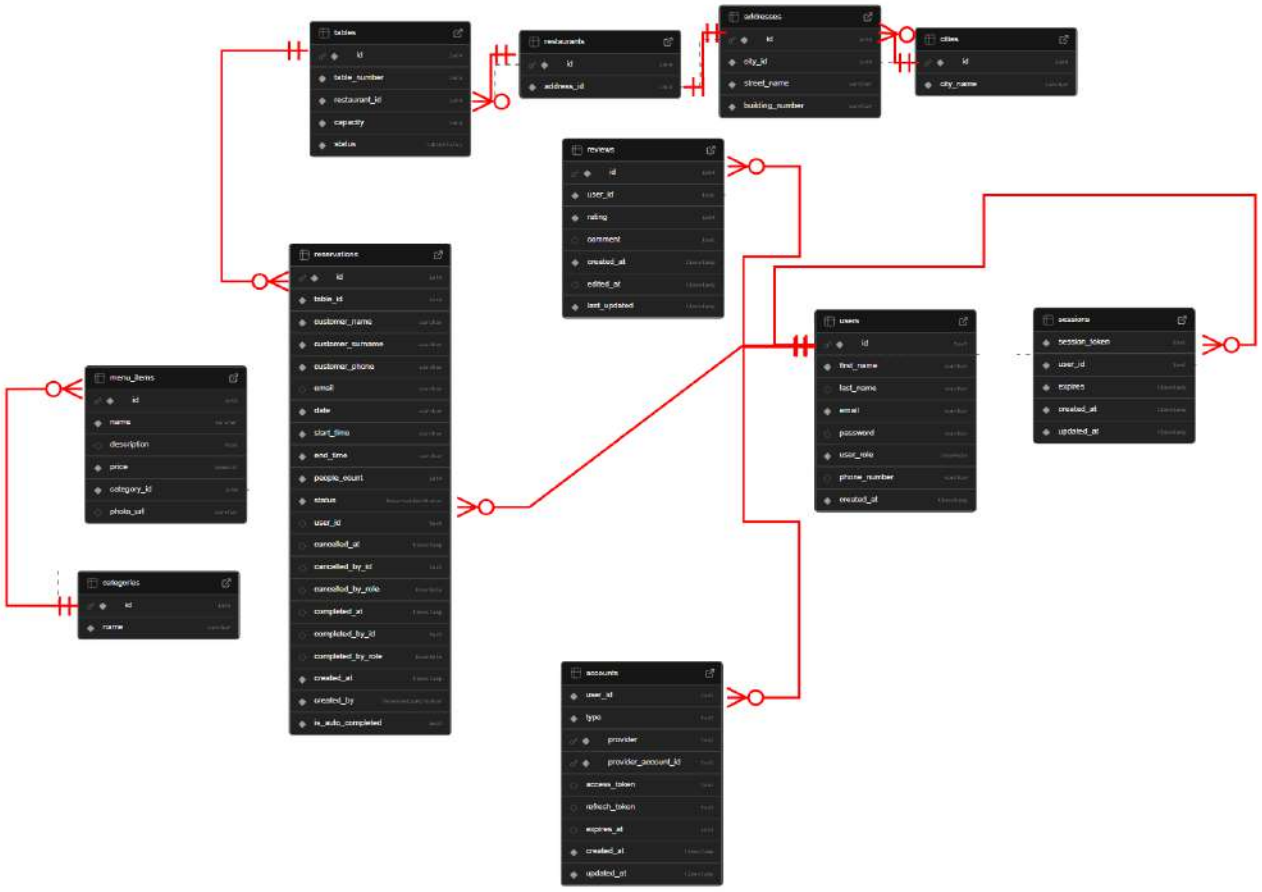

Рисунок 4.4 – Зв’язки в базі даних

Структури основних таблиць:

Таблиця 4.3 – Таблиця users (користувачів)

Назва поля	Тип даних	Опис
id	String, PK	Ідентифікатор користувача
name	varchar(50), NOT NULL	Ім’я користувача
surname	varchar(50)	Прізвище користувача
email	varchar(150), UNIQUE, NOT NULL	Електронна пошта
password	varchar(255)	Хеш пароля
phoneNumber	varchar(13)	Номер телефону
userRole	enum (Admin, Manager, User)	Роль користувача
createdAt	timestamp	Дата створення профілю

Таблиця 4.4 – Таблиця restaurants (ресторанів)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор ресторану
addressId	integer, UNIQUE, FK	Зв'язок із таблицею addresses

Таблиця 4.5 – Таблиця cities (міст)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор міста
city_name	varchar(100), UNIQUE, NOT NULL	Назва міста, унікальна в системі

Таблиця 4.6 – Таблиця addresses (адрес)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор адреси
city_id	integer, NOT NULL, FK	Зовнішній ключ на таблицю cities
street_name	varchar(150), NOT NULL	Назва вулиці
building_number	varchar(20), NOT NULL	Номер будинку

Таблиця 4.7 – Таблиця tables (столиків)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор столика
tableNumber	integer	Номер столика у ресторані, унікальний в межах ресторану

restaurantId	integer, FK	Зв'язок із рестораном
capacity	integer	Кількість місць
status	enum (free, reserved, occupied)	Статус столика

Таблиця 4.8 – Таблиця menu_items (компонентів меню)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор страви
name	varchar(50)	Назва страви
description	text (опційно)	Опис страви
price	decimal(10,2)	Ціна страви
categoryId	integer, FK	Категорія страви
photoUrl	varchar(255)	Посилання на фото страви

Таблиця 4.9 – Таблиця categories (категорій меню)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор категорії
name	varchar(50), UNIQUE	Назва категорії

Таблиця 4.10 – Таблиця reservations (бронювань)

Назва поля	**Тип даних**	**Опис**
id	serial, PK	Ідентифікатор бронювання
table_id	integer, NOT NULL, FK	Зовнішній ключ на таблицю tables
customer_name	varchar(50), NOT NULL	Ім’я особи, яка бронює
customer_surname	varchar(50), NOT NULL	Прізвище особи, яка бронює
customer_phone	varchar(12), NOT NULL	Телефон для зв'язку

email	varchar(150)	Електронна пошта
date	DATE, NOT NULL	Дата бронювання
start_time	varchar(5), NOT NULL	Час початку бронювання
end_time	varchar(5), NOT NULL	Час завершення бронювання
people_count	integer, NOT NULL	Кількість людей для бронювання
status	enum (active, completed, cancelled)	Статус бронювання
created_by	enum (Admin, Manager, User, Guest)	Хто створив бронювання
user_id	string, FK	Зовнішній ключ на таблицю users (якщо користувач авторизований)
created_at	timestamp, NOT NULL	Дата і час створення запису
completed_at	timestamp	Час завершення бронювання
completed_by_id	string, FK	Ідентифікатор користувача, який завершив бронювання
completed_by_role	enum (Admin, Manager, User)	Роль користувача, який завершив бронювання
cancelled_at	timestamp	Час скасування бронювання
cancelled_by_id	string, FK	Ідентифікатор користувача, який скасував бронювання
cancelled_by_role	enum (Admin, Manager, User)	Роль користувача, який скасував бронювання
is_auto_completed	boolean, default FALSE	Чи завершене бронювання автоматично системою

4.3 Взаємодія клієнта із сервером

Веб-застосунок в курсовій роботі побудований за архітектурним принципом **REST API**" де клієнтська частина взаємодіє з сервером шляхом надсилання HTTP-запитів та отримання відповідей [23].

Основні HTTP-методи, що використовуються у проєкті:

- **GET** – використовується для отримання даних із сервера.
- **POST** – застосовується для створення нових записів.
- **PATCH** – використовується для часткового оновлення наявних даних.
- **PUT** – застосовується для повного оновлення об'єкта.
- **DELETE** – використовується для видалення записів.

Усі маршрути відповідних API-ендпоінтів визначені у файлах **route.ts**. До того ж, в одному файлі не може бути більше одного, однакового за методом, ендпоінту. Тому окрім того, що розбиття по директоріям забезпечує кращу читаємість та модульованість системи, так ще й просто заборонено системою визначати всі шляхи в одному файлі.

Для відправлення HTTP-запитів з клієнтської частини використовується окремо налаштований екземпляр axiosInstance, який підтримує такі методи:

axiosInstance.get() / .post() / .put() / .patch() / .delete()

На рисунку 4.5 зображено принцип роботи клієнт-серверної взаємодії через REST API та використання HTTP-методів:

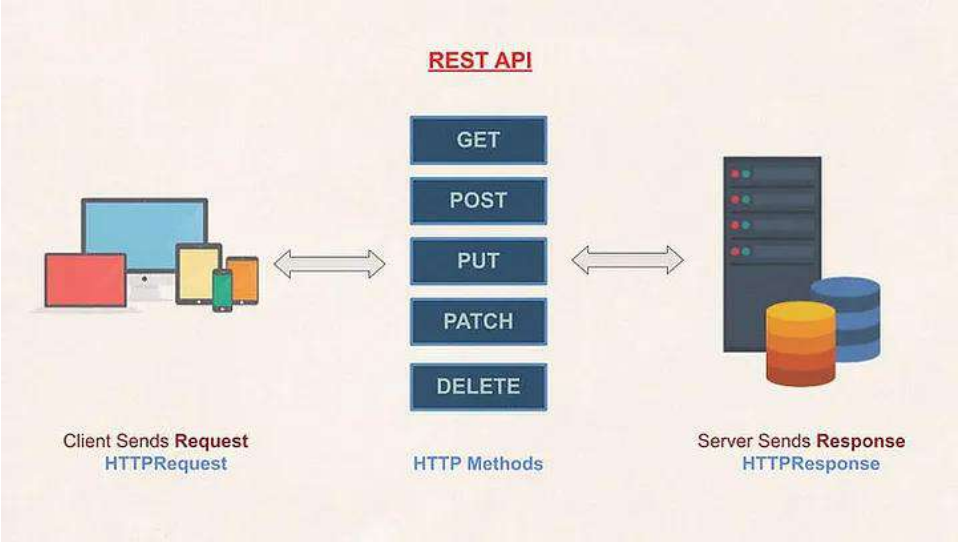

Рисунок 4.5 – Взаємодія клієнта із сервером через HTTP-методи

4.6 Функціонування системи

Для забезпечення стабільної роботи веб-застосунку та гарантування правильності введених даних реалізовано двоступеневу систему валідації: на клієнтському та серверному рівнях. Також сама база даних, якщо намагатися вставити в неї дані, які визначені інакше (не той тип, формат чи розмір), не дозволить цього і поверне помилку.

4.6.1 Валідація на клієнтській стороні

На стороні клієнта валідація введених користувачем даних здійснюється за допомогою бібліотек **React Hook Form** та **Zod**.

Zod-схеми визначають правила перевірки полів форми (наприклад, обов'язковість введення імені, правильність формату електронної пошти чи телефону, мінімальну кількість символів у паролі тощо).

У разі виявлення помилки валідації форма не надсилається на сервер, а користувач одразу отримує повідомлення про некоректно заповнене поле.

На рисунку 4.6 продемонстровано приклад такої схеми:

```
export const AddressSchema : ZodObject<{ cityId: ZodNumber; streetName... = z.object({
  cityId: z.number({ required_error: "Місто обов'язкове" }),
  streetName: z
    .string()
    .min(1, "Назва вулиці обов'язкова")
    .max(150, "Максимум 150 символів"),
  buildingNumber: z
    .string()
    .min(1, "Номер будинку обов'язковий")
    .max(20, "Максимум 20 символів"),
});
```

Рисунок 4.6 – Приклад Zod схеми для валідації адреси.

4.6.2 Валідація на серверній стороні.

На серверній стороні додатково проводиться валідація даних відправлянням запиту безпосередньо в базу даних. Zod-схема використовується безпосередньо і в API-роутах наступним чином:

const validatedData = UpdateUserSchema.parse(body);

У цьому прикладі перед оновленням профілю користувача дані, отримані у запиті, проходять перевірку за допомогою Zod-схеми.

Це дозволяє убезпечити систему від шкідливих або некоректних запитів, які можуть бути відправлені за межами інтерфейсу застосунку (через Postman або інші зовнішні інструменти).

Крім того, це дозволяє гарантувати, що тільки валідні та очікувані дані потрапляють до бази даних,

Саме тому, валідація на бекенді є критично важливою частиною загальної безпеки веб-застосунку.

4.6.3 Обробка помилок та повідомлення користувачу

У випадках, коли під час виконання запитів виникають помилки, система обробляє їх і виводить користувачеві зрозумілі повідомлення через механізм Toast-повідомлень (використовуючи бібліотеку react-hot-toast).

На рисунку 4.7 продемонстрований приклад повідомлення у разі помилки.

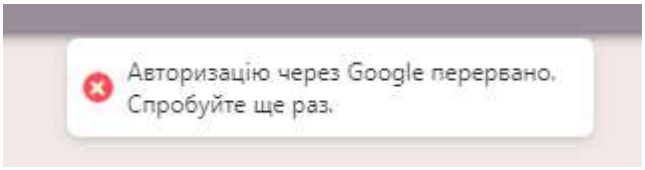

Рисунок 4.7 – Приклад Toast-повідомлення у разі помилки при авторизації

Окрім цього, ще перед надсиланням запиту, у місцях, де використовуються форми з полями для введення, виконується валідація на стороні клієнту, і під відповідним інпутом виводиться текстова підказка про помилку.

На рисунку 4.8 продемонстровано приклад локальних повідомлень про помилки введення:

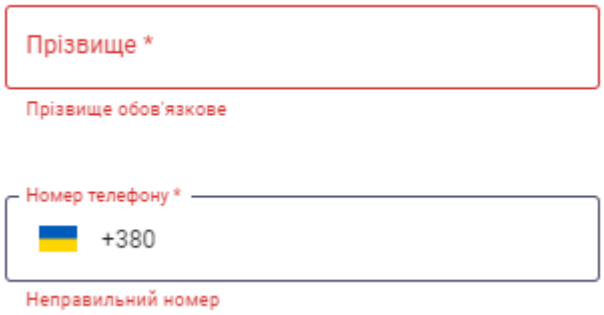

Рисунок 4.8 – Приклад локальних повідомлень про помилки під час заповнення форми

4.7 Висновки до розділу 4

У цьому розділі було детально описано архітектуру та реалізацію веб-застосунку: структуру проєкту, організацію бази даних, принципи взаємодії клієнта з сервером, механізми валідації даних та обробки помилок.

Розроблене рішення забезпечує надійність, безпеку обробки інформації та зручність користування.

5 ОСНОВНІ СТОРІНКИ З ДЕМОНСТРАЦІЄЮ ІНТЕРФЕЙСУ ВЕБ-ЗАСТОСУНКУ

У цьому розділі представлено основні сторінки та функціональні можливості веб-застосунку у вигляді скріншотів інтерфейсу.

Для кожної сторінки буде наведено короткий опис її призначення та ключового функціоналу.

5.1 Головна сторінка веб-застосунку

– це наче «обгортка книги». Вона виконує вітальної панелі для користувачів. На ній міститься інформація про ресторан, пропозиції для гостей, а також швидкий доступ до основних розділів системи.

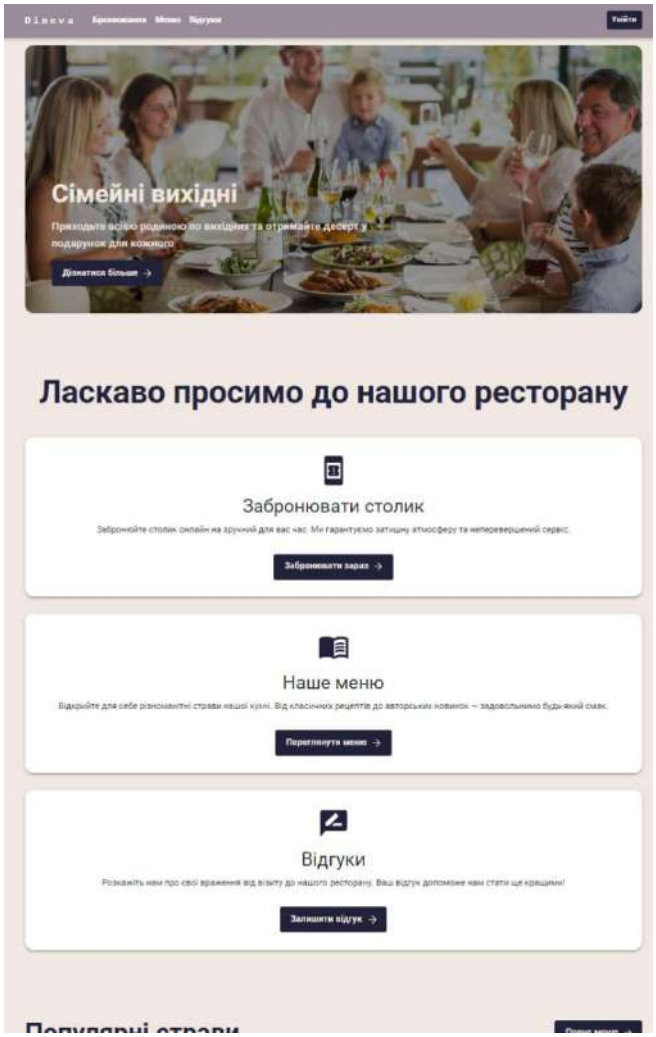

Рисунок 5.1 – Головна сторінка застосунку

5.2 Сторінка бронювання

дозволяє користувачу самостійно створити резервацію у вибраному ресторані, обравши ресторан, кількість осіб, дату та час, на який вони хочуть забронювати столик.

Після вибору часу відкривається модальне вікно для введення особистої інформації, де всі поля є обов’язковими.

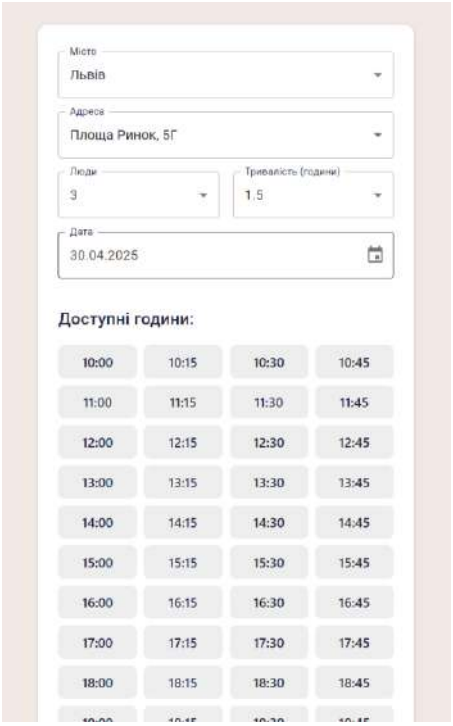

Рисунок 5.2 – Форма вибору параметрів бронювання

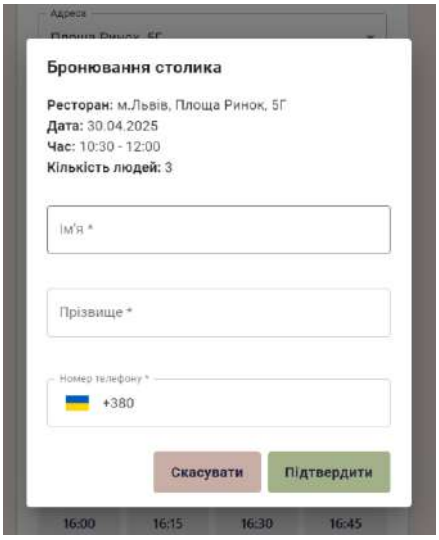

Рисунок 5.3 – Модальне вікно створення бронювання

5.3 Сторінка перегляду та керування меню

Тут користувачі можуть переглядати доступні категорії страв та окремі позиції, згруповані за категоріями. А менеджери можуть редагувати категорії та пункти меню за допомогою відповідних вікон.

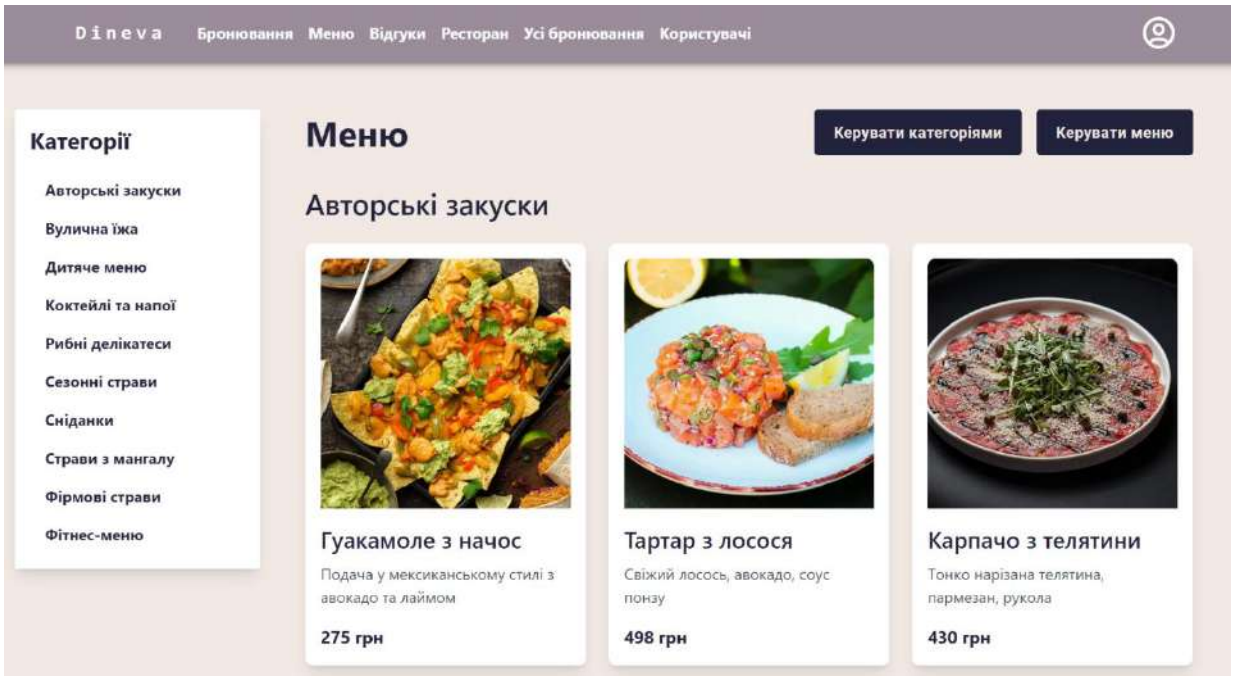

Рисунок 5.4 – Перегляд меню з категоріями як менеджер

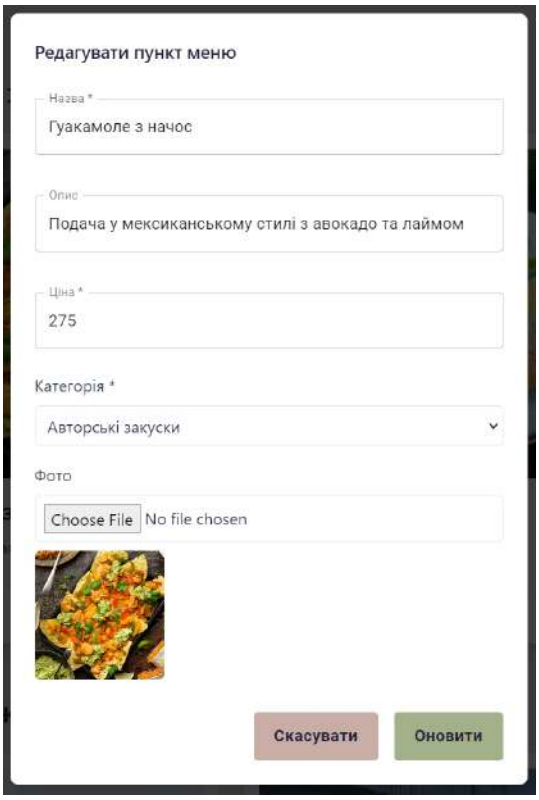

Рисунок 5.5 – Модальне вікно редагування страви меню

5.4 Сторінка перегляду та керування відгуками

Використовуючи цю сторінку користувачі можуть: переглядати залишені відгуки інших користувачів та писати нові відгуки. Реалізовано модальні вікна для редагування та видалення коментарів.

Окрім того, є можливість фільтрувати відгуки за рейтингом, переглядати лише свої власні відгуки та сортувати відгуки за датою (новіші або старіші).

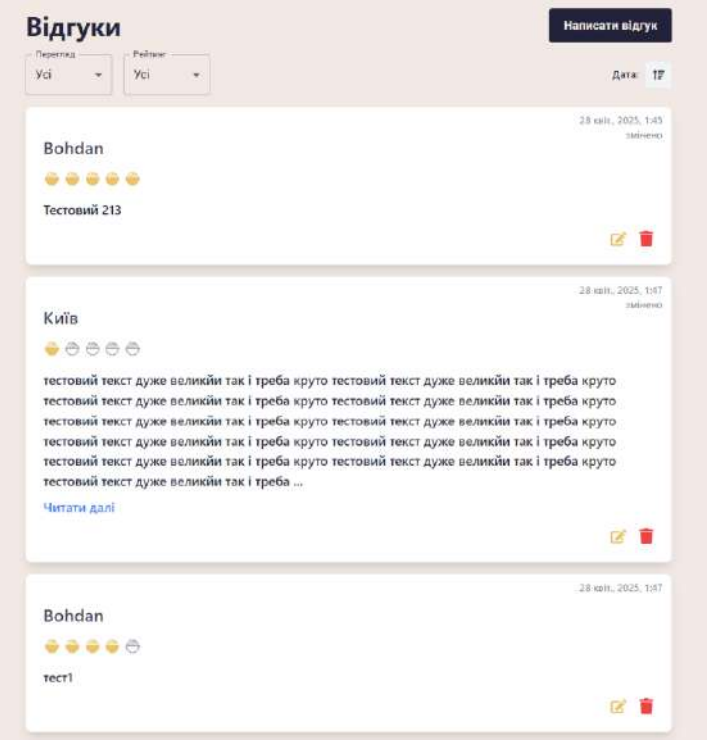

Рисунок 5.6 – Сторінка перегляду відгуків з фільтрами

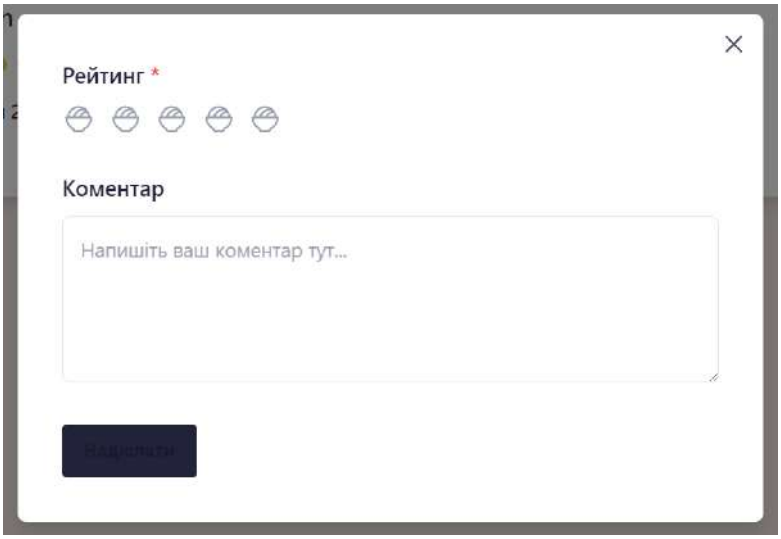

Рисунок 5.7 – Модальне вікно створення\редагування відгуку

5.5 Сторінки авторизації та реєстрації користувача

На сторінці входу користувач може авторизуватися за допомогою логіна та пароля або увійти через Google аккаунт.

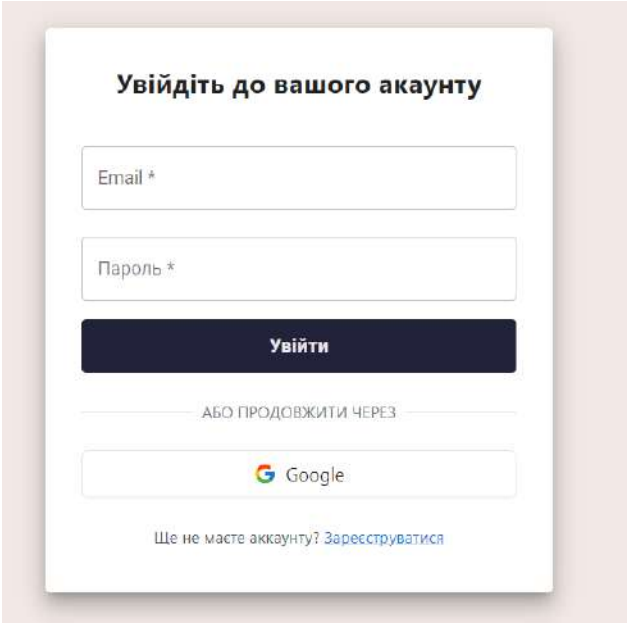

Рисунок 5.8 – Сторінка входу користувача

А для того, щоб зареєструватися користувач має заповнити всі обов'язкові поля, які позначені «*».

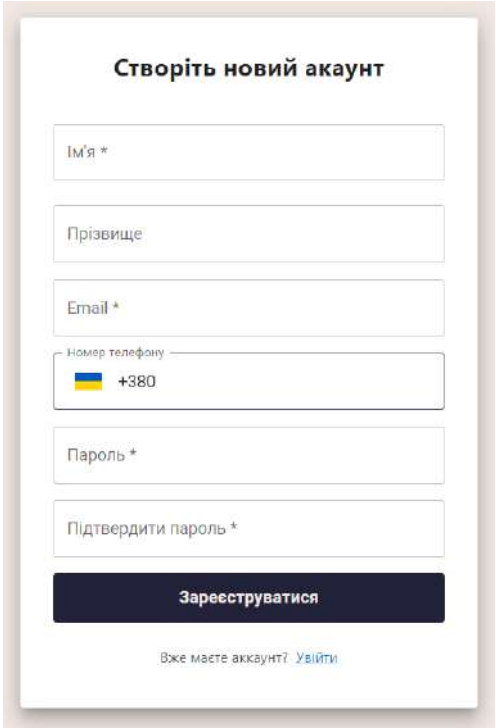

Рисунок 5.9 – Сторінка реєстрації нового користувача

5.6 Сторінка профілю користувача

Тут юзер може переглянути та змінити інформацію про свій профіль. Якщо користувач зареєструвався через гугл, то для видалення аккаунту, йому потрібно просто ввести слово «ВИДАЛИТИ», а якщо користувач має пароль (реєструвався через форму), то ввести свій пароль. Також користувач з паролем, може його змінити у відповідній формі.

Є окрема вкладка для перегляду бронювань користувача з можливістю фільтру лише активних та змінювати контактні дані чи майбутніх резервацій чи скасовувати їх.

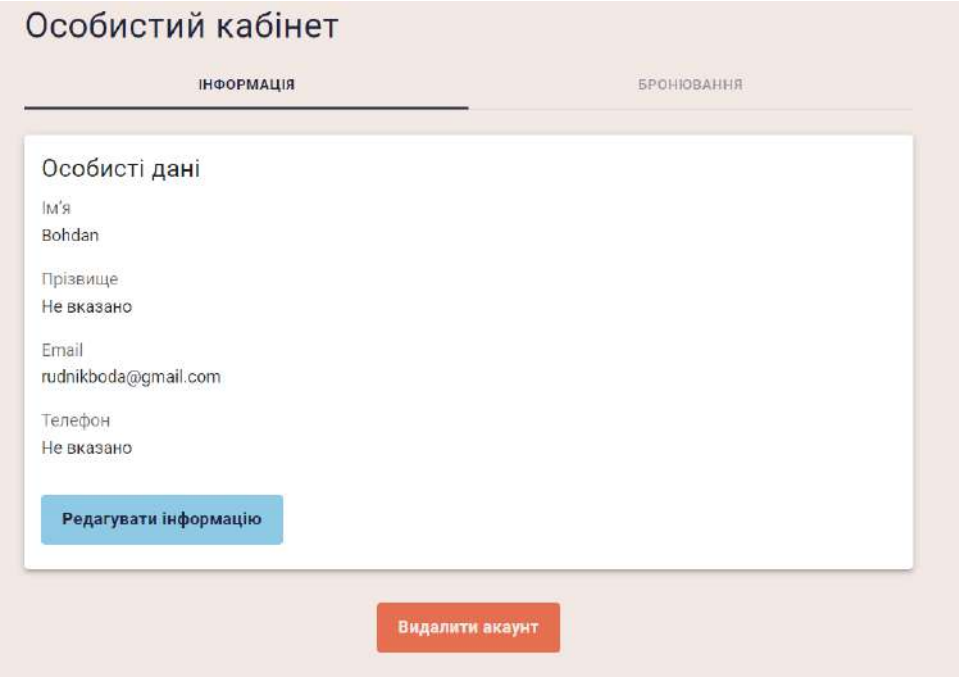

Рисунок 5.10 – Сторінка особистого кабінету

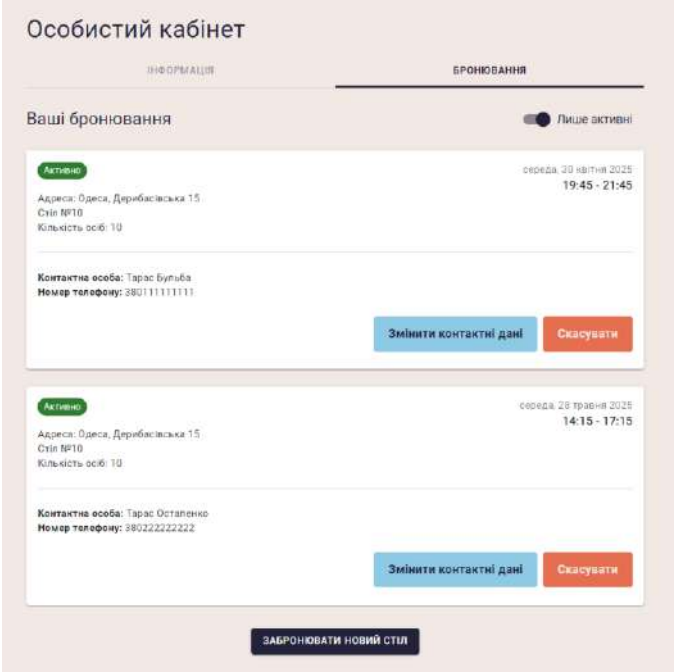

Рисунок 5.11 – Вкладка бронювань в профілі

5.7 Сторінка всіх резервацій

Доступ до цієї сторінки є лише у Адміна та Менеджера. Тут можна бачити повну інформацію про резервації, фільтрувати та сортувати їх по доступних полях. Є окрема колонка з діями, використовуючи яку, можна керувати активними (завершувати, скасовувати, редагувати інформацію) бронюваннями та переглядати детальну інформацію про вибране бронювання.

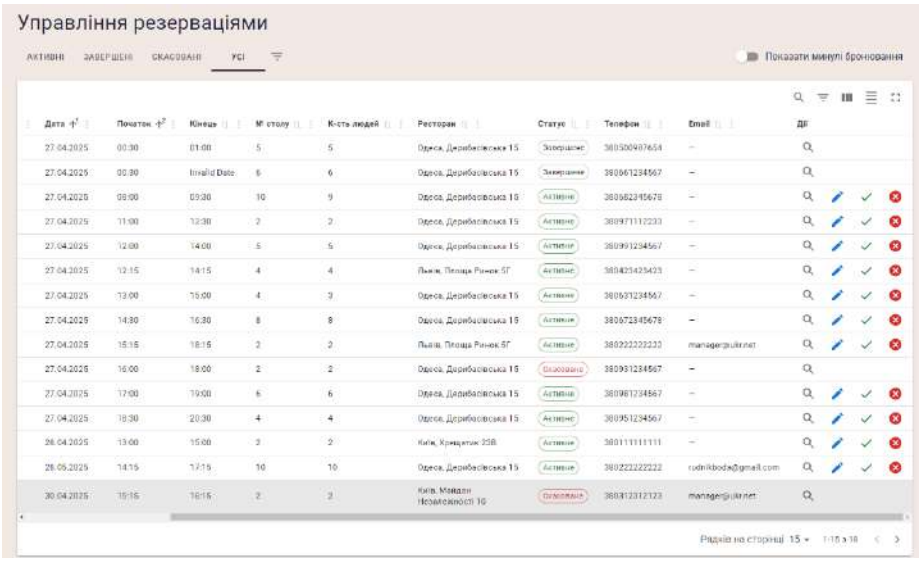

Рисунок 5.12 – Сторінка всіх резервацій

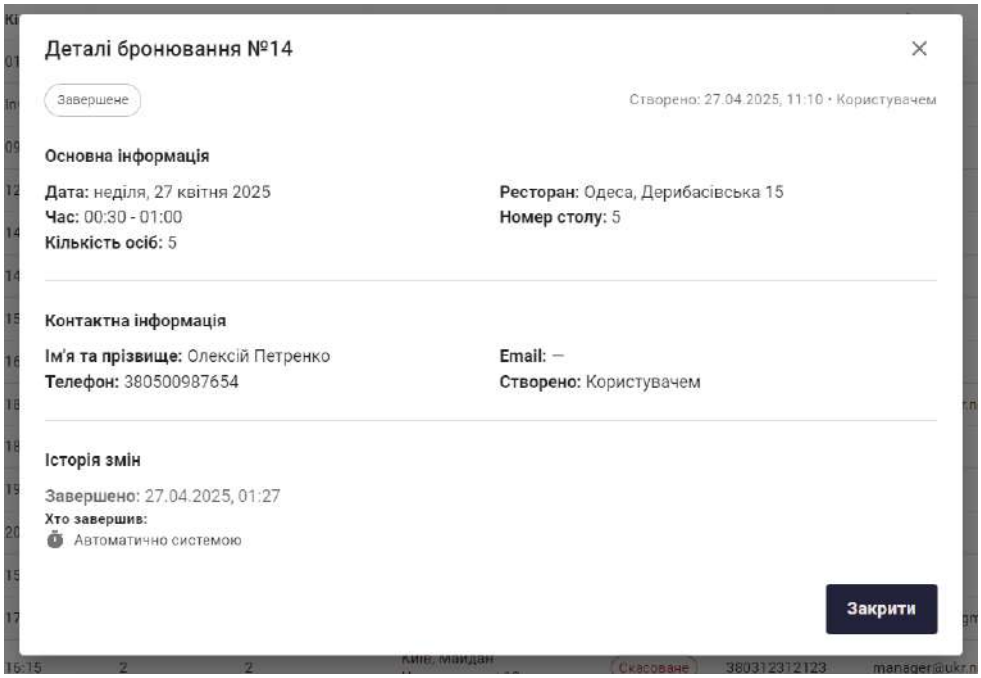

Рисунок 5.13 – Модальне вікно детальної інформації про бронювання

5.8 Сторінка зі списком користувачів

І тут доступ є лише у менеджера та адміна. При тому, менеджер бачить лише користувачів з роллю User й тільки ними може керувати. У той час адміністратор бачить всіх та може керувати User-ами (видаляти та підвищувати роль до менеджера) та Manager-ами (видаляти). При тому при видалені можна вибрати, чи скасовувати майбутні резервації користувача.

Також для цієї сторінки реалізоване модальне вікно для перегляду резервацій певного користувача.

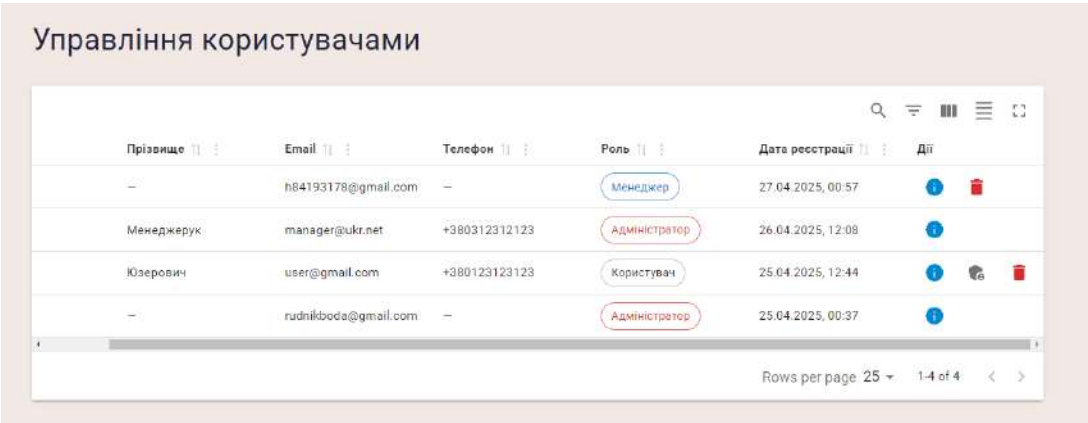

Рисунок 5.14 – Сторінка зі списком користувачів

Рисунок 5.15 – Модальне перегляду бронювань певного користувача

5.9 Сторінка управління компонентами ресторану

Доступ до цієї частини застосунку є лише у користувачів з особливими правами (Manager/Admin). На цій сторінці можна вибрати потрібний ресторан за адресою, після чого відобразиться таблиця столиків відповідного закладу. При розгортанні підтаблиць менеджером, під кожним столом відобразяться сьогоднішні бронювання, які відносяться до нього. Кожні 5хв автоматично оновлюються статуси столиків. За потреби, можна натиснути на кнопку «оновити дані», яка викликає відповідний метод одразу, без очікування. Якщо починається резервація, то статус столику стає «забронювано». Якщо менеджер закінчує чи скасовує активне бронювання (для цього в таблиці передбачені відповідні кнопки), то столик автоматично стає вільним.

Додатково є кнопка «завершити минулі резервації», яка завершує всі активні бронювання, у яких час закінчення минув.

Також є окремий статус для столику – «зайнятий». Він використовується, коли столик зайняв клієнт, який безпосередньо прийшов в ресторан без резервації заздалегідь. Щоб побачити, за який стіл можна посадити користувача є кнопка «Доступні столики», при натисканні якої відобразяться вільні столики на теперішній час.

Для того, щоб редагувати компоненти ресторану, потрібно переключити світч, і тоді з’являться кнопки для виконання цих дій. Можна редагувати,

видаляти, додавати міста, адреси, ресторани та столики. Але є обмеження: не можна створювати міста з однаковими назвами. Ресторан створюється лише на основі адреси. Тому при видалені ресторану, видаляється і адреса, а якщо й у міста не залишилось адрес, то зникає і саме місто. Ресторан не можна видалити, якщо до нього прив'язані столики.

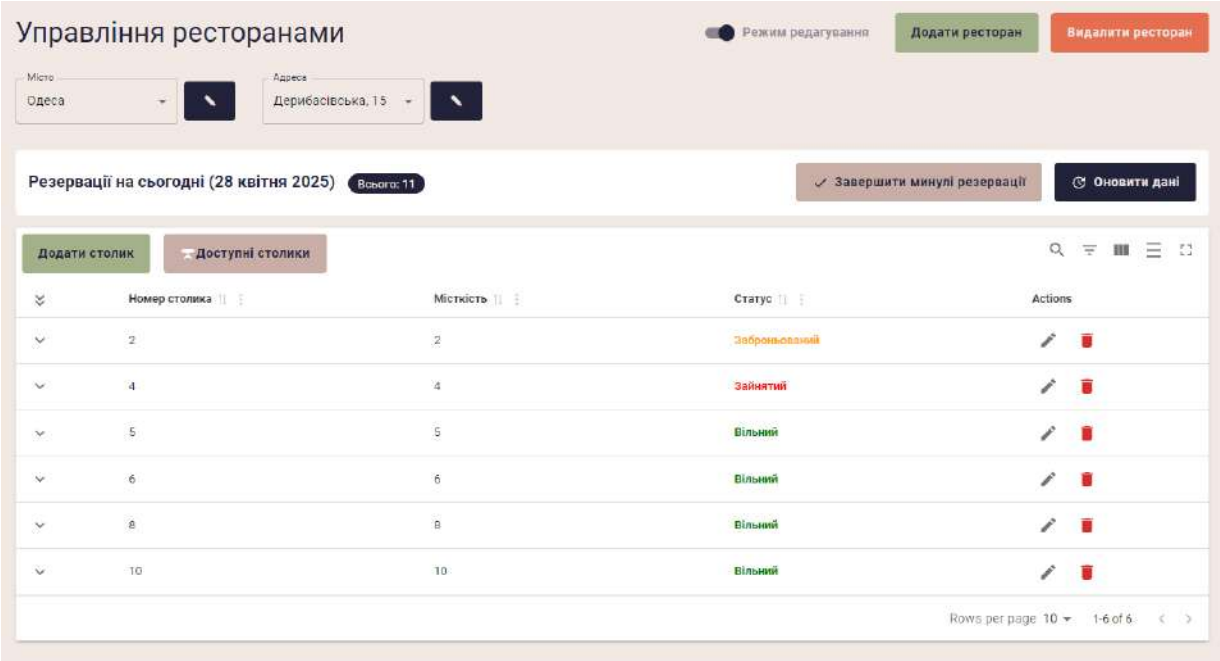

Рисунок 5.16 – Сторінка керуванням ресторану в режимі редагування

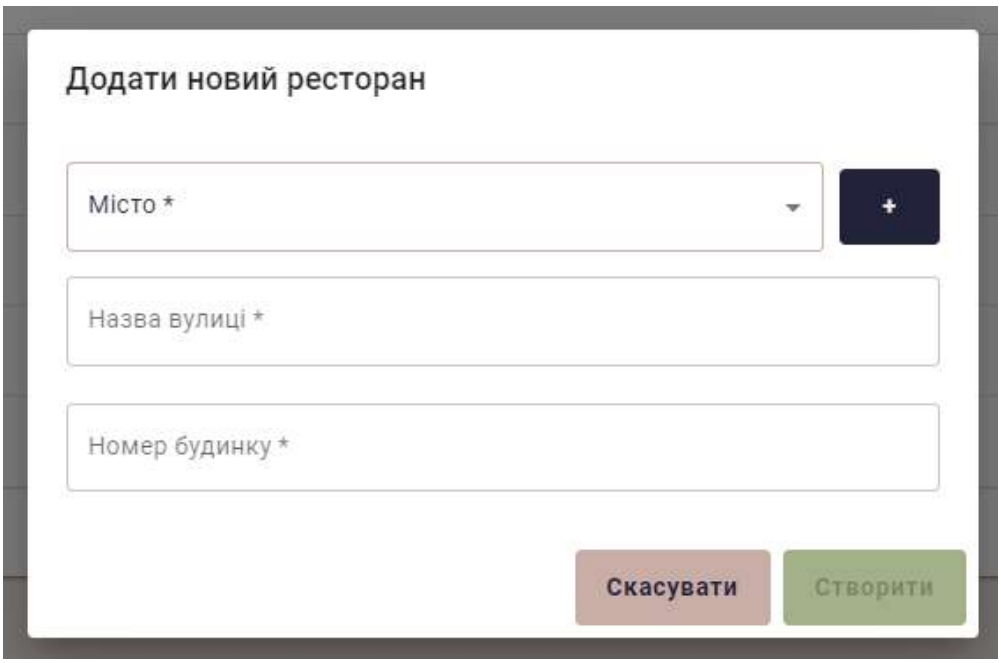

Рисунок 5.17 – Модальні вікно додавання нового ресторану

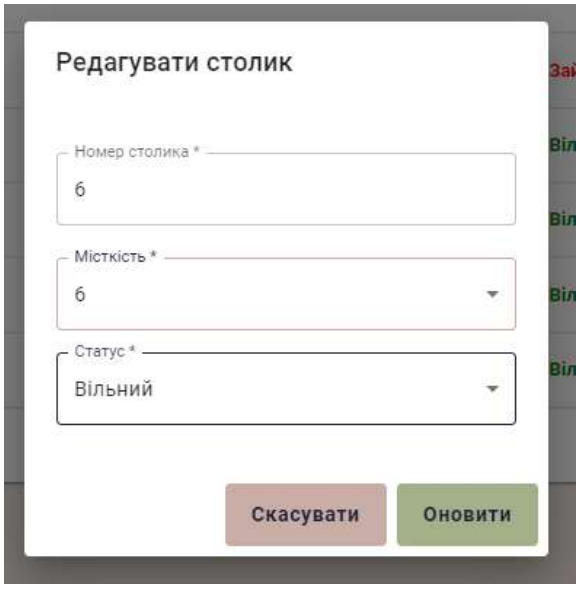

Рисунок 5.18 – Модальні вікно редагування\додавання столику

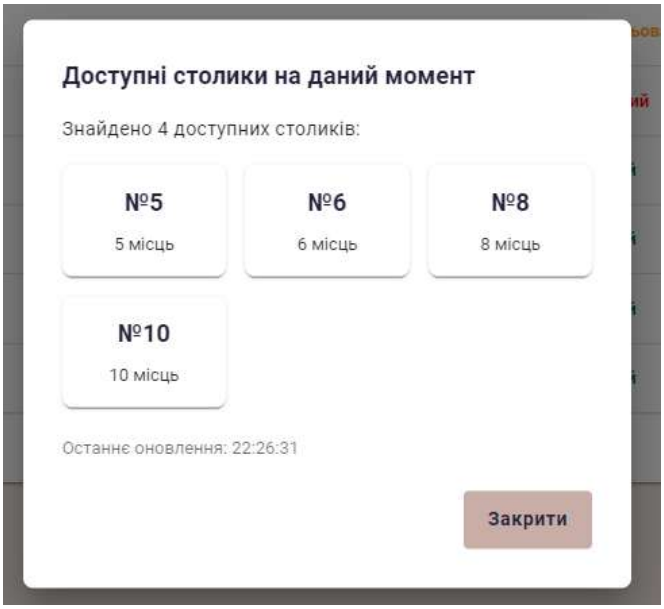

Рисунок 5.19 – Модальні вікно для перегляду вільних столиків

5.10 Висновки до розділу 5

У цьому розділі було описано всі сторінки реалізованого веб-застосунку та його функціональні можливості. Для кожної частини надані ілюстрації зовнішнього вигляду системи.

Було висвітлено можливості користувачів різних ролей (не-/ та авторизованих, менеджерів та адмінів). Частково згадані процеси реєстрації, бронювання, перегляду й редагування профілю, управління меню, відгуками та адміністрування ресторанами.

Додатково можна побачити, яка палітра кольорів використовувалась в інтерфейсі для візуальної орієнтації та зрозумілості на підсвідомому рівні функціоналу залежно від забарвлення. Більшість – це нейтральні, щоб не напружувати очі. А важливі функціональні кнопки, було виділено забарвлено наступним чином:

- Червоний колір – дії, яка пов’язані з видаленням, скасуванням.
- Зелений – для додавання нових елементів чи підтвердження успішних дій.
- Синій – дії, що пов’язані з інформативною ціллю та зміною наявних даних.

ВИСНОВКИ

Основним завданням в цій курсовій роботі було створення справного веб-застосунку, який би зміг автоматизувати процеси ресторанного бізнесу. Реалізоване програмне забезпечення дозволяє значно спростити бронювання столиків як для користувачів, так і для персоналу, надає можливість зручно відслідковувати резервації та керувати ними. Система з наявною авторизацією та аунтифікацієї надає можливості для легкого управління ресторанами, користувачами, компонентами меню, відгуками тощо. Окрім того інтуїтивно зрозумілий та кольорово-привабливий інтерфейс забезпечує комфортну взаємодію всіх користувачів із веб-застосунком.

Під час виконання роботи було проведено аналіз предметної області та аналіз схожих рішень у першому розділі задля розуміння специфіки сфери. Було описано використані технології та бібліотеки для побудови веб-застосунку в розділі 2. У розділі під номером 3 було сформульовано бізнес-вимоги та визначено основний функціонал, який і був реалізований в кінцевому варіанті.

Сам процес реалізації був детально описаний у 4-тому розділі. Було розглянути загальну структуру й зазначено особливості функціонування системи. Насамкінець, було наочно показано основний функціонал, що наявний на вебсторінках реалізованого застосунку, у 5-тому розділі.

Отже, використання створеного веб-застосунку дійсно може автоматизувати певні процеси ресторанного бізнесу та підвищити їх ефективність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SmartCafe. [Електронний ресурс].
   URL: https://smartcafe.com.ua/uk
2. Reston.ua. [Електронний ресурс].
   URL: https://reston.ua
3. Famiglia (Vero Vero). [Електронний ресурс].
   URL: https://famiglia.com.ua/vero-vero/
4. Avalon. [Електронний ресурс].
   URL: https://avalon1.choiceqr.com
5. ChoiceQR. [Електронний ресурс].
   URL: https://choiceqr.com/uk/
6. Mobile Website vs Mobile Apps: Pros and Cons - HSW Solutions. [Електронний ресурс].
   URL:https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/
7. Next.js Documentation. [Електронний ресурс].
   URL: https://nextjs.org/docs
8. React Documentation. [Електронний ресурс].
   URL: https://react.dev
9. Front-end без Next.js - як спорткар без двигуна: аналізуємо переваги та недоліки. [Електронний ресурс].
   URL: https://wezom.com.ua/ua/blog/front-end-bez-nextjs-yak-sportkar-bez-dviguna-analizujemo-perevagi-ta-nedoliki
10. TypeScript Documentation. [Електронний ресурс].
    URL: https://www.typescriptlang.org/docs/
11. Supabase Documentation. [Електронний ресурс].
    URL: https://supabase.com/docs
12. Cloudinary Documentation. [Електронний ресурс].
    URL: https://cloudinary.com/documentation

13. WebStorm – The Smartest JavaScript IDE. [Електронний ресурс]. URL: https://www.jetbrains.com/webstorm/
14. ESLint Documentation. [Електронний ресурс]. URL: https://eslint.org/docs/latest/
15. Prisma Documentation. [Електронний ресурс]. URL: https://www.prisma.io/docs
16. NextAuth.js Documentation. [Електронний ресурс]. URL: https://next-auth.js.org/
17. Material UI (MUI) Documentation. [Електронний ресурс]. URL: https://mui.com/
18. TailwindCSS Documentation. [Електронний ресурс]. URL: https://tailwindcss.com/
19. Чарльз Мур: Функціональні та нефункціональні вимоги - Guru99. [Електронний ресурс]. URL: https://www.guru99.com/uk/functional-vs-non-functional-requirements.html
20. Biplob Hosen: HTTP Request Methods - LinkedIn. [Електронний ресурс]. URL: https://www.linkedin.com/pulse/http-request-methodswe-can-perform-all-operation-post-biplob-hosen-xloec
21. Next.js Project Structure. [Електронний ресурс]. URL: https://nextjs.org/docs/app/getting-started/project-structure
22. Mert Enercan: Next.js 13 Folder Structure Overview - Medium. [Електронний ресурс]. URL: https://medium.com/@mertenercan/nextjs-13-folder-structure-c3453d780366
23. Eric Okemwa: HTTP Methods: A Comprehensive Guide - Medium. [Електронний ресурс]. URL: https://erokemwa.medium.com/http-methods-a-comprehensive-guide-b2c61b3d1788

ДОДАТОК А

Посилання на репозиторій GitHub з кодом курсової роботи:

https://github.com/cas08/Dineva