МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Чайковський Данило Євгенович
(прізвище, ім'я, по батькові здобувача)

на тему **Розробка програмного забезпечення замовлень клієнтів у ресторані**
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник **к.е.н., доцент** _____ **Лисенко В.С**
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 № 12
Завідувач кафедри _____
(підпис)
д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
«11» червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи Розробка програмного забезпечення замовлень клієнтів у ресторані

Керівник роботи к.е.н., доцент Лисенко В.С
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об’єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Розробка алгоритму розв’язання задачі

Розділ 3.Проектування бази даних для проекту

Розділ 4.Розробка програмного забезпечення

Об’єкт дослідження: сайт ресторану

Предмет дослідження: автоматизацію процесів сайту ресторану

Мета кваліфікаційної роботи:створення веб-сайту та адмін-панелі ресторану та автоматизація його процесів

5. Дата видачі завдання «04» квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ ___Лисенко В.С____

(підпис) (прізвище та ініціали)

Завдання одержав _____ __Чайковський Д. Є_

(підпис) (прізвище та ініціали)

ЗГОДА здобувача(чки) вищої освіти

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату та розміщення в Репозитарії Університету

Я, _____Чайковський Данило Євгенович_____(ПІП), підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота Розробка програмного забезпечення замовлень клієнтів у ресторані _____ _____ (назва роботи повністю) виконана самостійно та не містить академічного плагіату. Я не надавав(ла) і не одержував(ла) недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений(а). Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований(на), що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений(на).

Дата підпис ініціали, прізвище (власноруч)

АНОТАЦІЯ

на бакалаврську кваліфікаційну роботу

«Розробка програмного забезпечення замовлень клієнтів у ресторані»

Чайковського Данила Євгеновича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

Основним завданням дипломної роботи було створення програмного комплексу для автоматизації процесу замовлень у ресторані.

У бакалаврській дипломній роботі удосконалено теоретико-методичні підходи до створення прикладного програмного забезпечення.

Доповнено підходи до створення проєктів з урахуванням їх використання на різних операційних системах та у різних браузерах.

У розробленому програмному забезпеченні використано технології Node.js для серверної логіки, React з бібліотекою Material UI для клієнтської частини сайту, а також C# WinForms і OpenGL для створення візуального модуля адміністративної панелі.

У роботі запропоновано новий підхід до автоматизації внутрішніх процесів ресторану, що дозволяє мінімізувати участь персоналу в обробці замовлень та бронювань.

Ключові слова: автоматизація ресторану, система замовлень, Node.js, React, Material UI, C#, WinForms, OpenGL, MySQL.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	Розшифрування
API	Прикладний програмний інтерфейс — це набір чітко визначених методів для взаємодії різних компонентів.
СУБД	Система управління базми даних
UI	Користувацький інтерфейс
Вебсайт	Сукупність вебсторінок та залежного вмісту, доступних у мережі Інтернет, які об'єднані як за змістом, так і за навігацією під єдиним доменним ім'ям
Алгоритм	Набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій
SQL	Мова структурованих запитів
JSON	JavaScript Object Notation (Формат обміну даними)
MUI	Бібліотека інтерфейсів для React
OpenGL	Відкрита графічна бібліотека

ЗМІСТ

ВСТУП

Діджитал технології актуалізуються з кожним днем все більше. Програми стали вірним помічником у багатьох сферах нашого життя. А деякі бізнес-проекти отримали суттєву перевагу запровадивши в свої проекти діджитал засоби. Завдяки всесвітній павутині відкривається безліч можливостей, зокрема у сфері ресторанного бізнесу. Завдяки веб-сайтам, ресторани можуть запропонувати онлайн-замовлення доставки. При цьому сайти мають перевагу до замовлень їжі по телефону тим, що клієнти можуть побачити страву, яку замовляють, ознайомитися зі складом їжі. Переваги сучасних технологій існують не тільки для клієнтів. Програмне забезпечення також полегшує роботу самих працівників ресторану. Завдяки сучасним засобам можна швидко переглядати в який час забронюваний столик, організовувати замовлення їжі та рахувати дохід за день. Що немало важливо для бізнесу - ведення аналітики. Створення статистики в ручну займає забагато часу і не завжди є ефективним.

Метою дипломної бакалаврської роботи є створення діджитал-системи для замовлень клієнтів ресторану. Планується розробити програмне забезпечення для користувачів, яке буде доступне в онлайн режимі та розробка програмного забезпечення для робітників ресторану. Доцільно обрати напрям розробки веб-сайту для користувачів та адмін панелі для робітників. Веб-сайт не вимагає завантаження програми на комп’ютер, кожна людина матиме швидкий доступ до можливості замовлення. В той же час, програма для робітників повинна бути в обмеженому доступі і не залежати від інтернет-з’єднання.

Перед початком розробки дипломного проекту було поставлено за ціль виконати наступні завдання:

1. Розробка візуальної частини веб сайту
2. Розробка функціональної частини сайту
3. Інтеграція бази даних

4. Створення API для виконання основних операцій
5. Розробка адміністративної частини
6. Розробка модулю візуалізації залів ресторану

Тема розробки програмного забезпечення для ресторану актуальна, тому що ресторани потребують автоматизації процесу замовлень клієнтів. Завдяки цьому поток клієнтів збільшується, а час робітників економиться, що надає переваги бізнесу на конкуренському ринку. Впровадження можливості онлайн бронювання столів та замовлення їжі робить обслуговування швидшим та легшим для клієнтів. Візуалізація залів ресторану дозволяє швидко ознайомитися із схемою залу та обрати вільний на потрібний час столик. Впровадження адмін панелі дозволяє персоналу швидко реагувати на замовлення, а адміністраторам проводити аналіз та аналітику.

Об’єкт дослідження - веб сайт та адмін панель ресторану. Предмет дослідження - автоматизація процесів замовлення ресторану.

Обрані інструменти для розробки:

1. HTML
2. CSS
3. JavaScript
4. C#
5. OpenGL

РОЗДІЛ І
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Задача дипломної роботи - створення програмного забезпечення замовлення клієнтів ресторану. Для зручності роботу було поділено на дві частини: розробка адмін-панелі та розробка веб-сайту. Роботи матимуть точку перетину у частині обробки даних та матимуть спільну базу даних.

Особливістю задачі є необхідність розробки модуля візуалізації залів ресторану з використанням OpenGL, який дозволить клієнтам отримати реалістичне уявлення про розташування столиків та обрати оптимальний варіант для бронювання. Для працівників цей модуль допоможе забезпечити управління завантаженістю залів та покраще орієнтовність у розміщенні відвідувачів ресторану.

Характеристика задачі розробки веб-сайту для замовлень повинна передбачати потреби клієнтів. Основна задача сайту провести клієнта по всім етапам оформлення столику та замовлення. Відвідувач заходить на сайт, як правило, з однією або кількома цілями: ознайомитися з місцем, подивитися меню, забронювати столик або дізнатися, як зробити замовлення. Його поведінка є послідовною і прогнозованою — користувач очікує швидко знайти ключову інформацію, не вивчаючи структуру сайту занадто глибоко. Тому структура сайту повинна бути зрозуміла, доступ до важливих частин сайту повинна залишатися на видному місці і завжди доступною.

Характеристика задачі розробки адмін-панелі створюється на основі аналізу виконуваної роботи працівників ресторану. У процесі обслуговування клієнтів ресторану приймають участь персонал з різними посадами, кожен з яких виконує свої функції та може взаємодіяти з цифровою системою керування закладом. Адміністративна панель повинна бути орієнтована на спрощення виконання робочих задач адміністратора залу, менеджера,

офіціанта, кухаря. Кожна з цих ролей має масив особистих завдань, значну частину з яких на сьогодні виконується вручну або у вигляді неструктурованих нотаток, таблиць чи усного передавання інформації.

Адміністратор залу відповідає за розсадження клієнтів, координацію бронювань, контроль заповненості залу та комунікацію з відвідувачами. Зараз це здійснюється через паперові журнали, дзвінки або записи в месенджерах. Менеджер зміни слідкує за загальним навантаженням на персонал, контролює потік замовлень, розв'язує конфліктні ситуації та веде звітність. Офіціанти отримують замовлення, часто записуючи їх вручну або передаючи на кухню усно. Такий підхід не є точним, та займає багато часу. Кухарі орієнтуються на ці замовлення для приготування страв. Автоматизація допоможе офіціанту обирати їжу з існуючого переліку страв і одразу ж передавати їх кухарю, що значно економить час, та прискорює приготування страви. Для клієнтів швидкість - важливий фактор, тому таке рішення вплине позитивно на враження клієнтів про ресторан. Окрім того автоматизація більшості з цих завдань дозволяє значно зменшити навантаження на персонал. За допомогою адмін-панелі адміністратор зможе в режимі реального часу бачити на екрані план залу з позначенням вільних і зайнятих столиків, підтверджувати бронювання та призначати столи. Менеджер матиме доступ до статистики замовлень, графіка заповненості та історії бронювань, що полегшить прийняття оперативних рішень.

Характеристика створення візуального модуля залів ресторану формується на зовнішньому вигляду реального залу. Не всі об’єкти інтер’єру які є в ресторані повинні перемальовуватися, оскільки працівники потребують лише схематичне представлення залів. Столики — центральні об’єкти системи. Вони повинні відображатися з урахуванням розміру та кожен у той же частині кімнати, в якій він стоїть і в реальному залі. Саме тому потрібно передбачити малювання стін, простір повинен бути обмежений, а не відкритий. Для схематичного зображення не так важливо відтворювати деталі інтер’єру: матеріали столу, картини, люстри, тощо.

1.2. Огляд існуючих рішень автоматизації ресторанного бізнесу

Аналіз ринку показує, що заклади харчування, які впровадили цифрові технології для автоматизації процесів, демонструють зростання клієнтської бази на 15-20% та збільшення середнього чеку на 8-12% [1]. Особливо це стало відчутно після пандемії COVID-19, коли споживачі звикли до зручності онлайн-замовлень та безконтактного обслуговування.

На ринку вже існують різноманітні рішення для автоматизації ресторанного бізнесу, які можна умовно поділити на декілька категорій.

1. POS-системи (Point of Sale) — комплексні рішення для управління закладом, які включають модулі для обліку продажів, управління запасами, аналітики тощо.
2. Системи онлайн-бронювання — спеціалізовані сервіси для резервування столиків.
3. Системи доставки та онлайн-замовлень — рішення, інтегровані з сайтом ресторану або доступні через мобільні додатки.
4. Комплексні системи управління ресторанним бізнесом — рішення, що поєднують функціонал всіх попередніх категорій.

Більшість існуючих комерційних рішень (Poster, R-Keeper, iiko) пропонують інтегровані системи, однак вони мають ряд недоліків:

- Висока вартість впровадження та щомісячної підписки
- Обмежені можливості кастомізації під специфічні потреби конкретного закладу
- Надлишковий функціонал для малих та середніх закладів
- Складність інтеграції з іншими системами закладу

Таблиця 1.1 представляє порівняльний аналіз найпопулярніших систем автоматизації ресторанного бізнесу.

Таблиця 1.1

Порівняльний аналіз систем автоматизації ресторанного бізнесу

Параметр порівняння	Poster	R-Keeper	iiko	Власне рішення
Вартість впровадження	Середня	Висока	Висока	Низька
Щомісячна підписка	$50-200	$100-500	$100-400	Відсутня
Можливість кастомізації	Обмежена	Середня	Середня	Повна
Залежність від інтернет-з'єднання	Часткова	Часткова	Часткова	Налаштовувана
Візуалізація залів	Базова	Розширена	Розширена	Розширена з OpenGL
Складність інтеграції з існуючими системами	Середня	Висока	Висока	Низька

У цьому контексті важливо врахувати специфіку малих і середніх ресторанів, які часто не мають потреби в повноцінних ERP-рішеннях. Їхні ключові потреби — це:

- простий у використанні інтерфейс для адміністраторів і офіціантів;
- легке оновлення меню та управління замовленнями без потреби в технічному персоналі;
- відсутність регулярних ліцензійних платежів;
- адаптація під локальні мовні, податкові та сервісні умови;
- можливість адаптації під індивідуальний сценарій обслуговування.

Крім того, сучасні тренди вказують на зростаючий попит на:

- мобільні версії сайтів або PWA-додатки;
- безконтактні платежі та інтерактивні QR-меню;
- автоматичні нагадування через месенджери;
- внутрішню аналітику для власника чи менеджера;

- підтримку інклюзивності для користувачів з особливими потребами.

Аналіз існуючих рішень підтверджує доцільність розробки власної діджитал-системи для ресторану, яка забезпечить необхідний функціонал з урахуванням конкретних потреб закладу та дозволить уникнути недоліків комерційних продуктів.

1.3. Вимоги до програмного забезпечення

На основі характеристики задач та аналізу існуючих рішень були проаналізовані та створені вимоги до програмного забезпечення.

Сайт повинен бути орієнтований на клієнта, та легко навігувати його по потрібним частинам сайту. Тому на сторінці повинне бути передбачено меню, яке має пункти до необхідних сторінок з меню, а також інформацією яка може бути важливим для користувача: розташування ресторану, контактні дані. Також головна сторінка повинна мати велику кнопку, яка направляє користувача до меню ресторану. Сторінка меню повинна бути гнучкою, та надавати швидкий доступ до пошуку бажаної їжі клієнта. Тому треба передбачити систему категоризації страв. Окрім того, страви повинні мати власну сторінку, щоб користувач міг легко ознайомитися з вмістом страви. Товар повинен легко додаватися до кошика, в один клік. Форма для оформлення замовлення повинна бути проста та мати тільки необхідні поля, такі як ім’я та номер телефону. Кожна дія користувача повина супроводжуватися відповідними повідомленнями: товар додано у кошик, замовлення оформлено тощо.

Адмін панель орієнтована на працівників ресторану. Програма повинна надавати інтерфейс управління замовленнями та бронюваннями. Окрім того повинні бути створені форми, за допомогою яких персонал зможе створювати замовлення вручну.

Зважаючи на можливі зміни інтер’єру закладу, модуль візуалізації залів повинен підтримувати динамічну реконфігурацію. Програмне забезпечення

має дозволяти легко додавати або змінювати розміщення столів. Координати об’єктів, а також кількість персон, на яку кожен із них розрахований, повинні зчитуватись із бази даних. На основі цих даних у 3D-просторі кімнати відтворюється відповідна схема розміщення меблів.

Для реалізації клієнтської частини системи обрано стек JavaScript, зокрема Node.js для серверної логіки та React у поєднанні з бібліотекою компонентів Material UI для побудови інтерфейсу. Такий вибір зумовлений необхідністю створити сучасний, адаптивний веб-сайт, який буде зручно використовувати як з десктопів, так і з мобільних пристроїв. Node.js забезпечує хорошу продуктивність при обробці запитів і дозволяє легко побудувати бекенд частину сайту, а React дозволяє реалізувати динамічний інтерфейс із високим рівнем взаємодії без перезавантаження сторінки. Використання Material UI прискорює розробку і дозволяє дотриматися єдиного стилю оформлення, що відповідає принципам сучасного UX/UI-дизайну.

У порівнянні з Vue.js, який також є популярним фронтенд-фреймворком, React краще підходить для невеликих застосунків і має стабільніший життєвий цикл компонентів. Vue, хоч і простіший на старті, більше підходить для проектів з більш складною логікою, тоді як React дозволяє будувати більш контрольовані структури за допомогою хуків та контексту. Крім того, екосистема React має підтримку таких бібліотек, як Redux, React Router чи Next.js, дозволяє розширити функціонал без зміни базової архітектури.

Angular, ще один фреймворк, який розглядався як інструмент розробки сайту ресторану. Він орієнтований переважно на великі корпоративні системи. Його складна структура, жорстка типізація (через обов’язкове використання TypeScript).

Бібліотеки інтерфейсу Bootstrap чи Tailwind CSS були розглянуті як альтернатива Material UI. Однак Bootstrap значно поступається у плані сучасності дизайну та компонентного підходу, а Tailwind — хоча й дуже гнучкий — вимагає більше зусиль для створення складних інтерактивних

елементів. Material UI, навпаки, надає вже готову систему компонентів, повністю сумісну з React, що дає змогу зосередитися на логіці застосунку, а не на ручному проєктуванні кожного елемента інтерфейсу.

Адміністративна панель створюється як десктопний Windows-застосунок на платформі C# із використанням WinForms. Такий підхід обумовлений потребою у стабільному середовищі для внутрішнього використання персоналом, а також доступністю WinForms як технології, яка дозволяє швидко створювати зручні форми для обробки замовлень і бронювань. Додатково до панелі інтегрується модуль візуалізації залу, побудований з використанням OpenGL. Цей модуль забезпечує графічне представлення розташування столів у реальному часі. Обрання OpenGL пояснюється його широкими можливостями візуалізації, підтримкою 2D і 3D-графіки, а також достатньою продуктивністю для завдань подібного типу.

Усі дані, що циркулюють між сайтами та панеллю адміністратора, мають зберігатися в єдиній базі даних, побудованій на MySQL. Ця СКБД була обрана через сумісність з вибраними мовами розробки. Також було важливо врахувати, щоб СКБД була сумісна з обома частинами програмного забезпечення - адмін панеллю та сайтом.

Висновки до розділу 1

У поточному розділі було проведено аналіз предметної області, визначено ключові аспекти задачі та особливості її реалізації в контексті ресторанного обслуговування. Розглянуто існуючі програмні рішення, їхні переваги та обмеження, що дозволило сформувати актуальні вимоги до майбутньої системи. Також обґрунтовано вибір технологій для розробки клієнтської частини, адмін-панелі та бази даних, відповідно до специфіки задачі та цілей проекту.

РОЗДІЛ ІІ
ПРОЕКТУВАННЯ ЗАДАЧІ

2.1. Проектування алгоритму роботи сайту

Процес функціонування веб-сайту ресторану буде базуватись на послідовності дій користувача, спрямованих на перегляд меню, вибір страв та оформлення замовлення через інтуїтивно зрозумілий інтерфейс. Уся взаємодія з клієнтом буде організована у вигляді окремих логічних етапів, кожен з яких відповідає за конкретний підпроцес: ознайомлення з пропозицією, вибір позицій, заповнення персональних даних і відправка замовлення. Для чіткої візуалізації цих етапів і взаємозв’язків між ними буде застосовано засоби моделювання UML-діаграм, що дозволяють формально описати поведінку системи з позиції користувача.

Зокрема, для опису загальної взаємодії користувача із сайтом буде створено UML-діаграму прецедентів (Use Case Diagram), яка дозволить виявити основні сценарії використання системи. Це дасть змогу структурувати функціональні вимоги, які в подальшому будуть реалізовані у вигляді компонентів клієнтської та серверної частин.

Діаграма відображатиме ключові дії, що будуть доступні користувачу:

- перегляд усіх категорій меню та окремих страв;
- додавання обраних позицій до кошика;
- можливість редагування вмісту кошика;
- оформлення замовлення з введенням персональних даних;
- надсилання фінального замовлення до серверної частини системи.

На основі цих дій буде побудовано інтерфейс сайту, що взаємодіє з сервером через API-запити до бази даних MySQL. Дані про замовлення в реальному часі будуть передаватись до десктопного застосунку адміністратора (написаного мовою C#), де ці замовлення візуалізуватимуться у 3D-просторі з прив’язкою до відповідних столів у залах ресторану.

Використання UML-діаграм на етапі проєктування дозволяє:

формалізувати вимоги до сайту у зручному для аналізу форматі;

краще узгодити взаємодію між модулями системи;

спростити подальше проєктування компонентів UI/UX;створити основу для тестування логіки роботи замовлень.

Створену діаграму можна побачити на Рис. 2.1.

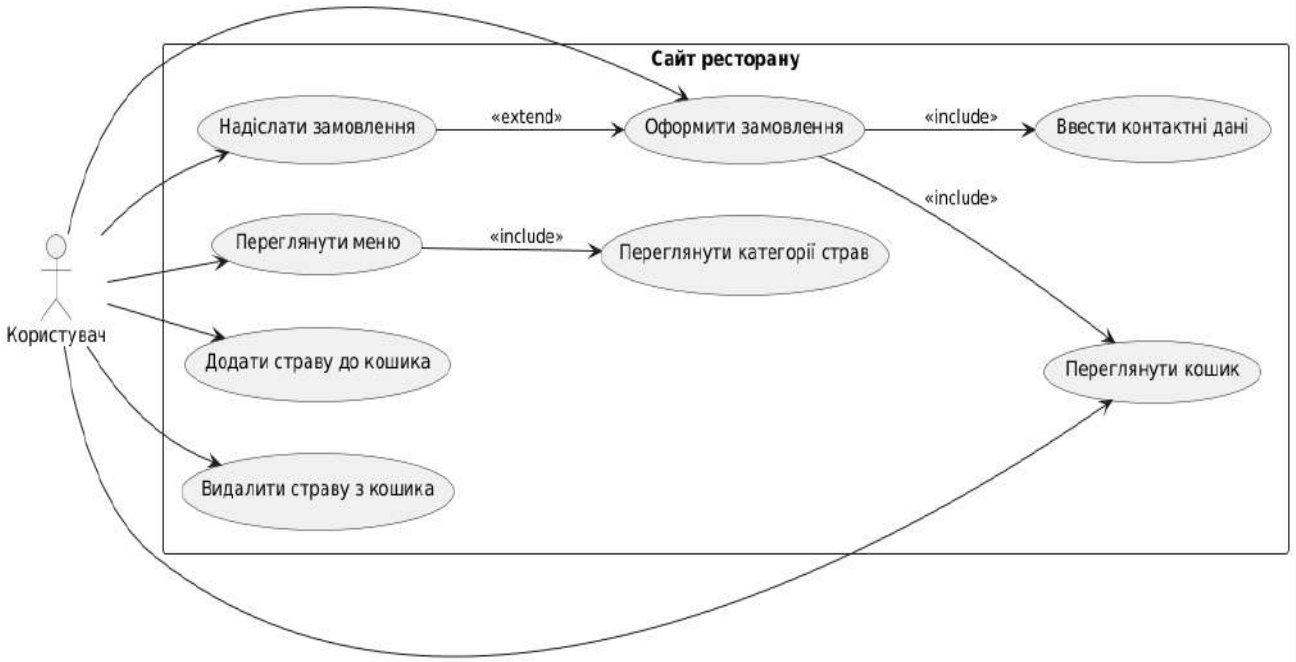

Рис. 2.1. UML-діаграма «Взаємодія користувача із сайтом ресторану»

У рамках подальшої деталізації алгоритму функціонування веб-сайту ресторану буде створено UML-діаграму активностей, яка демонструє покроковий сценарій взаємодії користувача з сайтом у процесі оформлення замовлення. На відміну від діаграми прецедентів, що лише визначає загальні функції, діаграма активностей дозволяє показати послідовність дій, включно з умовними переходами, розгалуженнями логіки та результатами виконання певних операцій.

Діаграма активностей слугуватиме для моделювання динамічної поведінки системи та забезпечить глибше розуміння процесу з боку як замовника, так і розробника. Вона дозволить проаналізувати сценарій від

початку відвідування сайту до остаточного надсилання даних до серверу. Також на ній буде видно, які дії виконуються користувачем, які — автоматично системою, і де можливі розгалуження логіки залежно від дій чи помилок користувача.

Основні етапи, які буде відображено на діаграмі активностей:

1. Початок сесії — користувач відкриває головну сторінку сайту.
2. Перегляд меню — можливість переглянути категорії страв і доступні позиції.
3. Додавання страв до кошика — вибрані позиції зберігаються в локальному сховищі або контексті стану.
4. Перехід до кошика — користувач переходить до сторінки перевірки замовлення.
5. Редагування замовлення — користувач може змінювати кількість, видаляти позиції або повернутись до меню.
6. Заповнення форми — користувач вводить ім’я, контактні дані, примітки.
7. Підтвердження замовлення — надсилання форми через HTTP POST-запит до сервера.
8. Отримання відповіді від сервера — відображення повідомлення про успішне створення або помилку.
9. Кінець процесу — завершення взаємодії, очищення кошика.

Побудовану діаграму яка демонструє покроковий сценарій взаємодії користувача з сайтом у процесі оформлення замовлення можна побачити на Рис. 2.2.

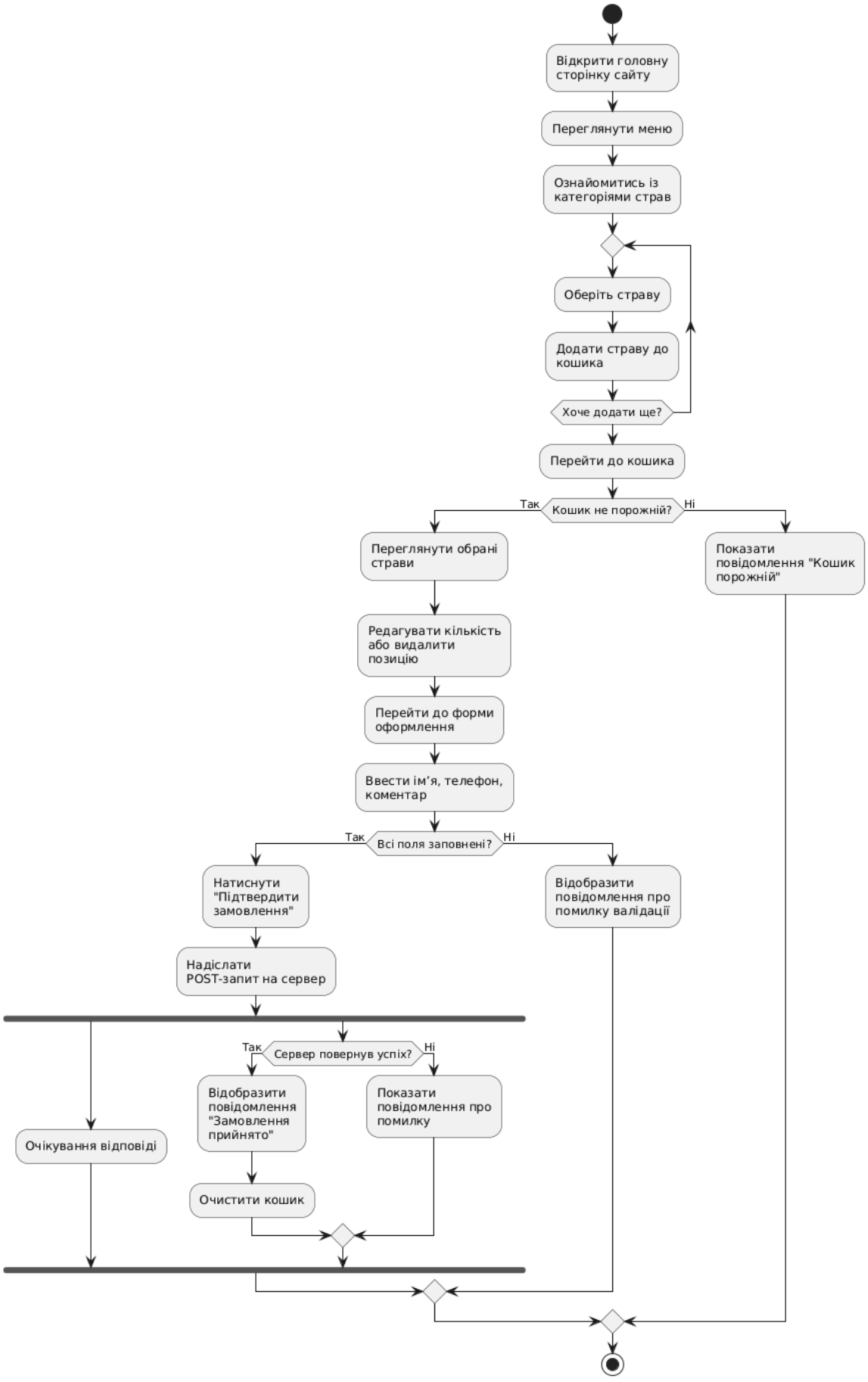

Рис. 2.2. UML-діаграма «Створення замовлення на сайті»

2.2. Проектування алгоритму роботи адміністративної панелі

Основою роботи адміністративної панелі є алгоритм — чітка та послідовна інструкція для системи щодо виконання операцій, необхідних для управління рестораном. Алгоритм складається з декількох основних етапів які представлені на Рис. 2.3.

Підготовчий етап:

- ініціалізація підключення до бази даних;
- доступ до основних операцій керування замовленнями, бронюваннями, столами, візуалізації залів;
- ініціалізація головного вікна додатку.

Основний етап:

відображення панелі керування з навігацією по ключових розділах;

- обробка замовлень;
- управління бронюваннями;
- створення нових замовлень та бронювань вручну;
- візуалізація залів: зчитування координат столів із бази даних, відображення в 3D-просторі.

Заключаючий етап:

- збереження змін у базі даних;
- підтвердження успішного виконання операцій через повідомлення користувачу;
- візуальне оновлення рендеренгу відносно оновлених даних.

Алгоритм взаємодії адміністративної панелі з базою даних базується на використанні сінглтон-патерну для встановлення єдиного з’єднання з базою даних протягом усього часу роботи програми. Це дозволяє уникнути створення надлишкових підключень та забезпечує ефективне управління ресурсами.

Рис. 2.3. UML-діаграма «Послідовність етапів роботи додатку»

На початку роботи адміністративної панелі ініціалізується екземпляр сінглтон-класу, який відповідає за підключення до бази даних MySQL. Цей об’єкт зберігає активне з’єднання, що використовується всіма компонентами системи. Взаємодія з базою даних здійснюється через окремий клас, який відповідає за певний функціональний модуль: замовлення, бронювання, додавання нових столів, візуалізація залів. Ці класи використовують сінглтон-

підключення для виконання SQL-запитів — отримання даних або внесення змін.

При виконанні операції клас формує відповідний запит, відправляє його через сінглтон-з’єднання, отримує результат та передає його в інтерфейс адміністратора. У разі внесення змін (додавання, оновлення, видалення) результати операцій підтверджуються відповідними повідомленнями (Рис. 2.4.).

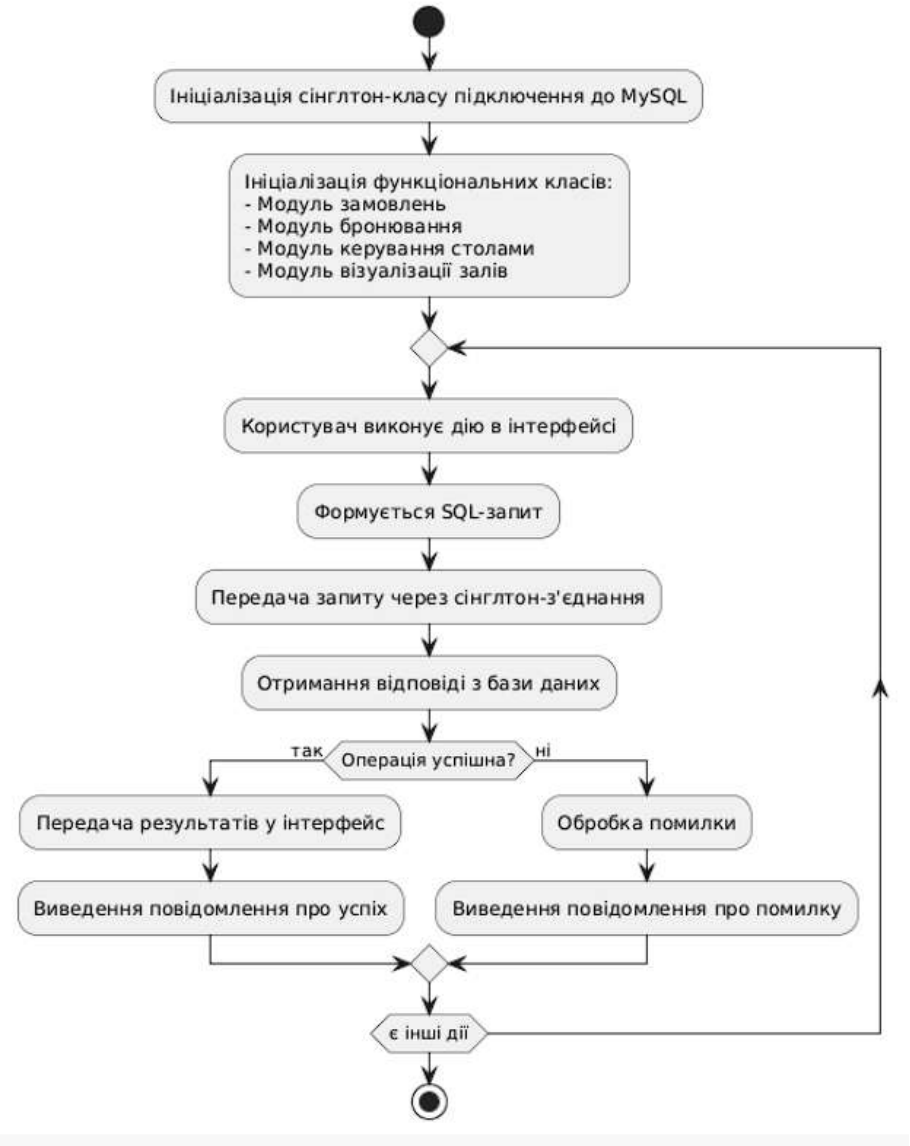

Рис. 2.4. UML-діаграма «Обробка запитів БД»

Алгоритм роботи візуального модуля (Рис. 2.5.) починається з ініціалізації інтерфейсу для взаємодії з базою даних, який відповідає за отримання параметрів столів: координат у просторі, кількості персон, на яку

розрахований стіл, а також кількості посадкових місць, для точного відтворення розташування меблів у залі.

Рис. 2.5. UML-діаграма «Робота візуального модуля»

Після отримання параметрів, інформація передається в клас рендерингу, який відповідає за створення тривимірних моделей столів. У цьому класі для кожного столу генерується геометрична модель відповідної форми паралелепіпеду з урахуванням заданих розмірів.

Далі відбувається ініціалізація контексту OpenGL, що включає налаштування камери, джерел світла, та проекційних матриць, які забезпечують коректне відображення сцени у тривимірному просторі. OpenGL використовує пайплайн рендерингу, де вершини моделей трансформуються у

координати екрану, після чого здійснюється растеризація та застосування текстур або кольорів.

Кожен стіл позиціонується в 3D-просторі згідно з координатами, зчитаними з бази даних. Для покращення візуального сприйняття реалізуються базові ефекти освітлення, що додають реалістичності об’єктам. Також використовується буфер глибини для коректного відображення перекривань між столами.

Висновки до розділу 2

У цьому розділі було розглянуто, як саме працюють ключові частини системи для клієнтів та для персоналу ресторану. Було описано алгоритм роботи сайту, взаємодію користувача з меню, кошиком та оформленням замовлення, а також особливості побудови інтерфейсу з використанням сучасного JavaScript-стека. Окремо розглянуто логіку функціонування адміністративної панелі — від підключення до бази даних до візуалізації залів у 3D.

Було побудовано алгоритми роботи програм, які допомогли зробити логічний поділ відповідальностей між класами, застосування шаблонів проектування, роботу з OpenGL-сценою. На основі цих підходів реалізовано механізм, який автоматизує внутрішні процеси й надає персоналу зручний інструмент для щоденної роботи, який покриває всі робочі потреби.

РОЗДІЛ ІІІ
ПРОЕКТУВАННЯ БАЗИ ДАНИХ ДЛЯ ПРОЕКТУ

3.1. Структура бази даних

Побудова бази даних є критичним етапом при розробці інформаційної системи для будь-якого проєкту. Вона визначає, як інформація зберігатиметься, опрацьовуватиметься та передаватиметься між усіма компонентами програмного забезпечення. База даних забезпечує узгоджене, централізоване та ефективне управління ключовими бізнес-процесами: обробкою замовлень, управлінням залами та столами, адмініструванням працівників, категоризацією меню, збереженням історії бронювань та динамікою виконання замовлень. Саме база даних дозволяє зібрати всі ці сутності в єдину цілісну систему, яка може бути масштабована, підтримувана та інтегрована з іншими модулями.

Основою даної реалізації є реляційна база даних, побудована на СУБД MySQL. Вибір саме MySQL був зумовлений її надійністю, широким розповсюдженням, відкритою ліцензією, високою продуктивністю у багатокористувацькому середовищі, а також її сумісністю з мовою програмування JavaScript, яка використовується на серверній частині (Node.js). У порівнянні з іншими СУБД, такими як Microsoft Access, SQLite або Oracle, MySQL показує кращі результати при обробці великої кількості одночасних з’єднань, має розвинену систему безпеки, підтримку реплікації, журналювання транзакцій, а також є достатньо гнучкою для інтеграції з іншими платформами та мовами програмування. Особливо важливою є її здатність працювати в умовах реального часу, коли велика кількість запитів надходить одночасно, що характерно для ресторанного бізнесу в години пік.

Однією з ключових переваг MySQL у даному проєкті є те, що вона слугує надійним централізованим сховищем даних, до якого одночасно мають доступ як вебклієнти (через сервер на Node.js), так і локальна система.

Десктопний застосунок працює безпосередньо зі структурою бази даних, отримуючи актуальну інформацію про бронювання та замовлення, а також надає змогу візуалізувати простір ресторану у тривимірному форматі для огляду розміщення столиків у залі. Завдяки цьому досягається синхронізована взаємодія між офіціантами, адміністраторами, кухнею та клієнтами, що значно підвищує загальну ефективність обслуговування.

Логічна структура бази даних спроєктована відповідно до принципів нормалізації до третьої нормальної форми (3НФ), що дозволяє усунути надмірність даних, підвищити їх цілісність та забезпечити ефективний доступ до інформації. На етапі проєктування була проведена декомпозиція предметної області на окремі сутності, кожна з яких отримала власну таблицю. Наприклад, інформація про столи й зали зберігається в окремих таблицях tables та rooms, пов’язаних між собою зовнішніми ключами. Така організація дає змогу уникнути дублювання однакових даних у різних частинах системи та забезпечує гнучкість при подальших змінах структури. Принцип 3НФ передбачає, що кожна таблиця містить лише атрибути, безпосередньо пов’язані з її первинним ключем, а всі зв’язки між сутностями реалізуються через зовнішні ключі. Наприклад, таблиця orders зберігає зв’язок між замовленням і стравою, фіксуючи ідентифікатори бронювання (booking) і страви (dish), а також кількість порцій, що дозволяє точно визначити склад кожного замовлення.

Кожна таблиця в базі містить набір полів, де визначено тип даних, обмеження на значення, а також технічні атрибути, такі як дата створення або останньої модифікації запису. Наприклад, поля created_at та changed_at забезпечують можливість автоматичного відстеження змін, що є важливим для реалізації контролю версій та аудиту. Деякі таблиці, наприклад, таблиця замовлень, містять додаткові обмеження (наприклад, перевірку на позитивність кількості порцій), що дозволяє зменшити ризик введення некоректних даних.

Особливу роль відіграють зв’язки між таблицями, які встановлені через зовнішні ключі. Наприклад, кожне замовлення пов’язане з конкретним бронюванням, а кожна страва — з певною категорією меню. Важливо відзначити, що реалізовані обмеження на рівні бази даних забезпечують автоматичне видалення пов’язаних записів у разі видалення батьківського елементу, наприклад, при видаленні залу автоматично видаляються пов’язані з ним столи.

Додатковим важливим елементом побудови бази стало забезпечення безпеки та стабільності системи. У MySQL реалізована можливість керування правами доступу до таблиць і окремих операцій (читання, вставка, оновлення, видалення), що дозволяє обмежити потенційно небезпечні дії для користувачів з різними ролями. З боку серверної частини (Node.js) використовуються перевірки на стороні сервера, що додатково захищає систему від SQL-ін’єкцій і несанкціонованих змін.

Загалом, обрана структура бази даних забезпечує повну підтримку функціоналу системи: керування залами, столами, меню, замовленнями та працівниками; обробку інформації про бронювання; динамічне оновлення стану замовлень у реальному часі; синхронізацію між сайтом і десктопним інтерфейсом. Вона побудована з урахуванням масштабованості, що дозволяє у майбутньому розширити функціональність або інтегрувати систему з іншими сервісами (наприклад, зовнішніми платіжними платформами чи сервісами аналітики), не порушуючи при цьому основної структури. Такий підхід забезпечує стабільність, гнучкість та надійність усієї цифрової інфраструктури ресторану.

3.2. Розробка бази даних застосунку

Процес побудови структури бази даних розпочинається з опису основних загальних сутностей, що формують фундамент інформаційної моделі системи. Однією з таких первинних сутностей є зали, які є просторовою

основою організації обслуговування клієнтів у ресторані. Кожна зала має власну назву, кількість столів, що можуть бути розміщені в ній, та служить основою для розміщення бронювань.

У таблиці rooms зберігається інформація про фізичні приміщення (зали) ресторану, кожна з яких є логічною одиницею, де можуть перебувати столи. Така абстракція дозволяє системі не лише гнучко реагувати на зміни розташування чи кількості столів, а й дає змогу візуалізувати простір у десктопному застосунку, де адміністратор може розставити столи у 3D-просторі відповідно до поточного розташування у реальному залі. Опис сутності room знаходиться в табл. 3.1

Таблиця 3.1

Опис сутності «Зала»

Назва поля	**Тип даних**	**Is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор зали, первинний ключ таблиці
name	VARCHAR(63)	Ні	Назва зали, яка відображається в інтерфейсах
max_tables	INT	Ні	Максимальна кількість столів, які можуть бути розміщені у залі
created_at	DATETIME	Так	Дата й час створення запису. Присвоюється автоматично при додаванні
changed_at	DATETIME	Так	Дата й час останнього оновлення запису. Автоматично оновлюється при зміні даних

Ця таблиця є ключовою для моделювання фізичного простору закладу. Завдяки розділенню залів і столів на окремі сутності досягається високий рівень масштабованості — у випадку змін у приміщенні (додавання або видалення залів) не потрібно змінювати інші таблиці, що значно спрощує обслуговування та розвиток системи. Поле max_tables дозволяє контролювати максимально допустиме заповнення кожної зали, що може використовуватись під час візуального розміщення в застосунку для перевірки коректності конфігурації. Поля created_at і changed_at важливі для ведення журналу змін і аудиту

Процес моделювання просторових елементів ресторану продовжується таблицею tables, яка безпосередньо представляє окремі столи, доступні для бронювання та обслуговування клієнтів. Столи є невід'ємною частиною логістики обслуговування в залі — кожен стіл має фіксовану місткість та розташовується у певній залі. Саме ця сутність поєднує просторову структуру з функціональністю бронювання, оскільки всі майбутні замовлення клієнтів будуть опосередковано пов'язані з конкретним столом.

В таблиці кожен стіл має свою унікальну назву, яка може відповідати номеру або позначенню в межах зали, до якої він належить. Зв'язок між столом і залою реалізовано через зовнішній ключ room, що дозволяє системі обробляти інформацію про розташування столів, групувати їх та, при потребі, видаляти всі столи, пов'язані з конкретною залою. Поле max_people відіграє важливу роль при бронюванні, оскільки дозволяє попередньо оцінити, чи достатньо місць у конкретному столі для вказаної кількості клієнтів. Опис сутності table знаходиться в табл. 3.2

Ця таблиця є основою для організації посадочних місць у ресторані. Зв'язок з таблицею rooms дозволяє адаптувати простір залежно від планування закладу — наприклад, переносити столи з однієї зали в іншу або динамічно формувати схеми розміщення в інтерфейсі адміністрування. Додатково, поле max_people забезпечує контроль допустимої кількості клієнтів за кожним столом, що є важливою функціональністю для перевірки

коректності під час бронювання через клієнтський інтерфейс чи десктопну частину.

Таблиця 3.2

Опис сутності «Стіл»

Назва поля	**Тип даних**	**Is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор стола, первинний ключ таблиці
name	VARCHAR (255)	Ні	Назва або номер стола, відображається в інтерфейсі
room	INT	Ні	Ідентифікатор зали, до якої належить стіл. Зовнішній ключ на rooms(id)
max_people	INT	Ні	Максимальна кількість клієнтів, які можуть одночасно сидіти за цим столом
created_at	DATETIME	Так	Дата й час створення запису. Встановлюється автоматично
changed_at	DATETIME	Так	Дата останнього оновлення запису. Оновлюється автоматично
x	INT	Ні	Координати розташування столу по x
y	INT	Ні	Координати розташування столу по y

Таблиця workers відповідає за зберігання інформації про працівників, які взаємодіють із системою керування рестораном — зокрема, здійснюють обробку бронювань, перегляд замовлень і змінюють стан системи. Ця сутність є важливою частиною як організаційної, так і безпекової моделі системи. Кожен працівник має унікальний логін та пароль, що забезпечує

персоналізований доступ до системи та дозволяє вести журнал активностей кожного користувача окремо.

У таблиці передбачено базові поля, необхідні для автентифікації працівника: login та password, а також fullname — повне ім’я, яке відображається в інтерфейсі адміністратора. Значення логіна повинно бути унікальним, аби виключити можливість конфліктів між обліковими записами, тому воно має відповідне обмеження у структурі таблиці. Опис сутності worker знаходиться в табл. 3.3

Таблиця 3.3

Опис сутності «Робітник»

Назва поля	**Тип даних**	**is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор працівника, первинний ключ таблиці
fullname	VARCHAR(255)	Ні	Повне ім’я працівника, що відображається в інтерфейсі
login	VARCHAR(255)	Ні	Унікальний логін, що використовується для входу в систему
password	VARCHAR(255)	Ні	Пароль у зашифрованому вигляді для автентифікації користувача
created_at	DATETIME	Так	Дата й час створення запису. Встановлюється автоматично
changed_at	DATETIME	Так	Дата останнього оновлення запису. Оновлюється автоматично

Ця таблиця лежить в основі контролю доступу до функцій системи ресторану. Вона дозволяє призначити відповідального за конкретне бронювання або замовлення, забезпечити розмежування прав та створити логіку відстеження активності працівників. Надалі, при потребі, до таблиці можуть бути додані додаткові поля — наприклад, роль або рівень доступу працівника для реалізації більш детального управління правами в системі.

Таблиця bookings є центральним елементом модуля управління бронюваннями. Вона містить всю необхідну інформацію про записи клієнтів: дату й час бронювання, контактні дані, кількість осіб, додаткові коментарі, а також інформацію про працівника, який прийняв бронювання.

Завдяки наявності зовнішнього ключа на worker, система може точно визначити відповідального працівника за кожну бронь, що підвищує прозорість обслуговування та забезпечує можливість контролю якості роботи персоналу. Додаткове текстове поле description дозволяє вільно зазначати побажання клієнта або внутрішні примітки персоналу. Контактний номер (phone) є важливим з точки зору зворотного зв’язку з клієнтом. Опис сутності bookings знаходиться в табл. 3.4

Таблиця 3.4

Опис сутності «Бронювання»

Назва поля	**Тип даних**	**is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор бронювання, первинний ключ таблиці
booking_datetime	DATETIME	Ні	Дата і час, на які здійснено бронювання

Продовження таблиці 3.4

worker	INT	Ні	Ідентифікатор працівника, який створив запис (зовнішній ключ на workers)
name	VARCHAR(255)	Ні	Ім’я клієнта, який здійснив бронювання
phone	VARCHAR(13)	Ні	Контактний телефон клієнта
people_count	INT	Ні	Кількість осіб, зазначена у бронюванні
description	VARCHAR(500)	Так	Додаткові побажання чи примітки до бронювання
created_at	DATETIME	Так	Дата й час створення запису. Встановлюється автоматично
changed_at	DATETIME	Так	Дата останнього оновлення запису. Оновлюється автоматично

Завдяки структурі цієї таблиці система може гнучко обробляти клієнтські запити та підтримувати історію замовлень. Всі поля підібрано з урахуванням мінімально необхідного для якісної організації бронювань, однак структура залишається відкритою до розширення — наприклад, у майбутньому можна буде додати поля для зазначення обраного столу або статусу виконання замовлення.

Наступною таблицею в структурі бази даних ресторану є dish_categories, яка відповідає за класифікацію страв. Ця таблиця дозволяє систематизувати меню, забезпечуючи логічне групування страв за категоріями (наприклад, «Суші», «Супи», «Напої» тощо). Це не лише покращує користувацький досвід на клієнтському інтерфейсі, а й оптимізує внутрішні процеси пошуку, додавання та редагування страв.

Наявність окремої таблиці для категорій забезпечує високий рівень масштабованості — у разі зміни назви чи опису категорії не потрібно оновлювати всі пов’язані страви. Зовнішній ключ у таблиці dishes буде посилатися на цю таблицю, що забезпечує логічну зв’язаність даних та зменшує дублювання. Опис сутності dish_categories знаходиться в табл. 3.5

Таблиця 3.5

Опис сутності «Категорія страв»

Назва поля	**Тип даних**	**is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор категорії, первинний ключ
name	VARCHAR(255)	Ні	Назва категорії (наприклад, «Гарячі страви», «Десерти»)
description	VARCHAR(255)	Так	Короткий опис категорії (наприклад, особливості або тип страв)
created_at	DATETIME	Так	Дата створення запису. Автоматично встановлюється при створенні
changed_at	DATETIME	Так	Дата останнього оновлення. Оновлюється автоматично при зміні запису

Ця таблиця є важливим інструментом для гнучкого адміністрування меню. Вона дозволяє швидко змінювати структуру меню без впливу на основні дані про страви. Це особливо зручно при сезонних оновленнях або впровадженні спеціальних розділів (наприклад, «Новинки тижня», «Страви до посту»). Усі поля передбачають достатню довжину для описових назв та

пояснень, що важливо для зручної навігації як у внутрішньому застосунку, так і на клієнтському інтерфейсі.

Наступною в структурі бази даних є таблиця dishes, яка зберігає інформацію про всі доступні страви в меню ресторану. Вона є однією з центральних сутностей у системі, оскільки безпосередньо відображає асортимент, з яким взаємодіє як персонал, так і користувачі клієнтського інтерфейсу. Кожна страва належить до певної категорії, що забезпечує її логічне групування, і може мати додаткову інформацію, зокрема опис, зображення та статус доступності.

Така структура дозволяє легко масштабувати меню, додавати нові позиції або тимчасово приховувати страви без їх повного видалення. Для зручності адміністрування та виводу даних у фронтенді передбачене поле з посиланням на зображення, яке може бути використане для візуального представлення страв у вебінтерфейсі. Наявність поля is_available дає змогу оперативно керувати відображенням страв на сайті, наприклад, приховати позицію, яка тимчасово недоступна. Опис сутності dishes знаходиться в табл. 3.6

Таблиця 3.6

Опис сутності «Категорія страв»

Назва поля	**Тип даних**	**is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор категорії, первинний ключ
name	VARCHAR(255)	Ні	Назва категорії (наприклад, «Гарячі страви», «Десерти»)
description	VARCHAR(255)	Так	Короткий опис категорії (наприклад, особливості або тип страв)

Продовження таблиці 3.6

created_at	DATETIME	Так	Дата створення запису. Автоматично встановлюється при створенні
changed_at	DATETIME	Так	Дата останнього оновлення. Оновлюється автоматично при зміні запису

Завдяки структурі таблиці можна не лише зручно виводити меню на сайті, а й адаптувати його до внутрішніх процесів, таких як складання замовлень, аналіз популярності позицій чи створення акцій. Зв’язок з таблицею dish_categories забезпечує гнучке групування, а наявність зображень сприяє покращенню користувацького досвіду при перегляді меню.

Наступною сутністю є таблиця orders, яка відповідає за збереження інформації про замовлені страви в рамках конкретного бронювання. Ця таблиця виконує роль зв’язуючої ланки між стравами та бронюваннями, реалізуючи зв’язок «багато до багатьох» через додаткову інформацію — кількість замовлених одиниць. Таким чином, для кожного бронювання можна зберігати перелік страв, які були замовлені клієнтом, а також їхню кількість.

Ця структура дає змогу аналітично обробляти дані про замовлення, формувати рахунки, відстежувати популярність страв, контролювати навантаження на кухню та забезпечувати швидке оновлення замовлень через клієнтську або десктопну систему. Завдяки чітко визначеним зовнішнім ключам до таблиць bookings і dishes, забезпечується цілісність даних і виключається можливість прив’язки неіснуючої страви або бронювання. Опис сутності orders знаходиться в табл. 3.7

Таблиця orders відіграє роль у формуванні конкретних замовлень на основі попередньо створених бронювань. Її структура дозволяє зберігати детальну інформацію про кожну позицію замовлення, а також забезпечує

точне формування підсумкових чеків, що особливо важливо як для обслуговуючого персоналу, так і для фінансового обліку. Всі дані можуть передаватись до десктопного застосунку, де замовлення відображаються в режимі реального часу та можуть бути прив’язані до конкретних столів у візуальному інтерфейсі.

Таблиця 3.7

Опис сутності «Замовлення»

Назва поля	**Тип даних**	**is NULL**	**Короткий опис**
id	INT, AUTO_INCREMENT	Ні	Унікальний ідентифікатор замовлення, первинний ключ
booking	INT	Ні	Зовнішній ключ на таблицю bookings, вказує, до якого бронювання належить
dish	INT	Ні	Зовнішній ключ на таблицю dishes, вказує на замовлену страву
count	INT, CHECK > 0	Ні	Кількість одиниць страви, замовлених у межах одного бронювання
created_at	DATETIME	Так	Дата створення запису. Встановлюється автоматично
changed_at	DATETIME	Так	Дата останнього оновлення. Оновлюється автоматично при зміні запису

Це були основні сутності бази даних, для створення програмного забезпечення ресторану.

Висновки до розділу 3

У цьому розділі було детально розглянуто структуру бази даних, яка застосовується у системі обліку бронювань і замовлень ресторану азійської кухні. Основну увагу приділено поетапному опису кожної сутності — від загальних понять до конкретних таблиць. Усі таблиці спроєктовані згідно з принципами реляційного підходу та нормалізовані до третьої нормальної форми, що дає змогу уникнути надмірності, підвищити цілісність даних та забезпечити ефективну обробку інформації. В рамках опису надано вичерпну характеристику кожного поля: тип даних, можливість бути порожнім (Is NULL), а також функціональне призначення.

Проєкт передбачає інтеграцію з клієнтською частиною через Node.js та підтримує передачу даних до десктопного застосунку, реалізованого на C#, що дозволяє візуалізувати замовлення та бронювання в тривимірному просторі. Така архітектура забезпечує гнучкість, масштабованість і зручність адміністрування, а також формує надійну основу для подальшого розвитку системи. Ретельно розроблена структура таблиць створює надійний фундамент для ефективного функціонування бази даних у межах інформаційної системи ресторану.

РОЗДІЛ IV
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Розробка адміністративної моделі ресторану

Початок роботи з програмою починається через головне вікно з меню. Персоналу пропонується переглянути зали, бронювання або ж створити новий стіл (Рис. 4.1.).

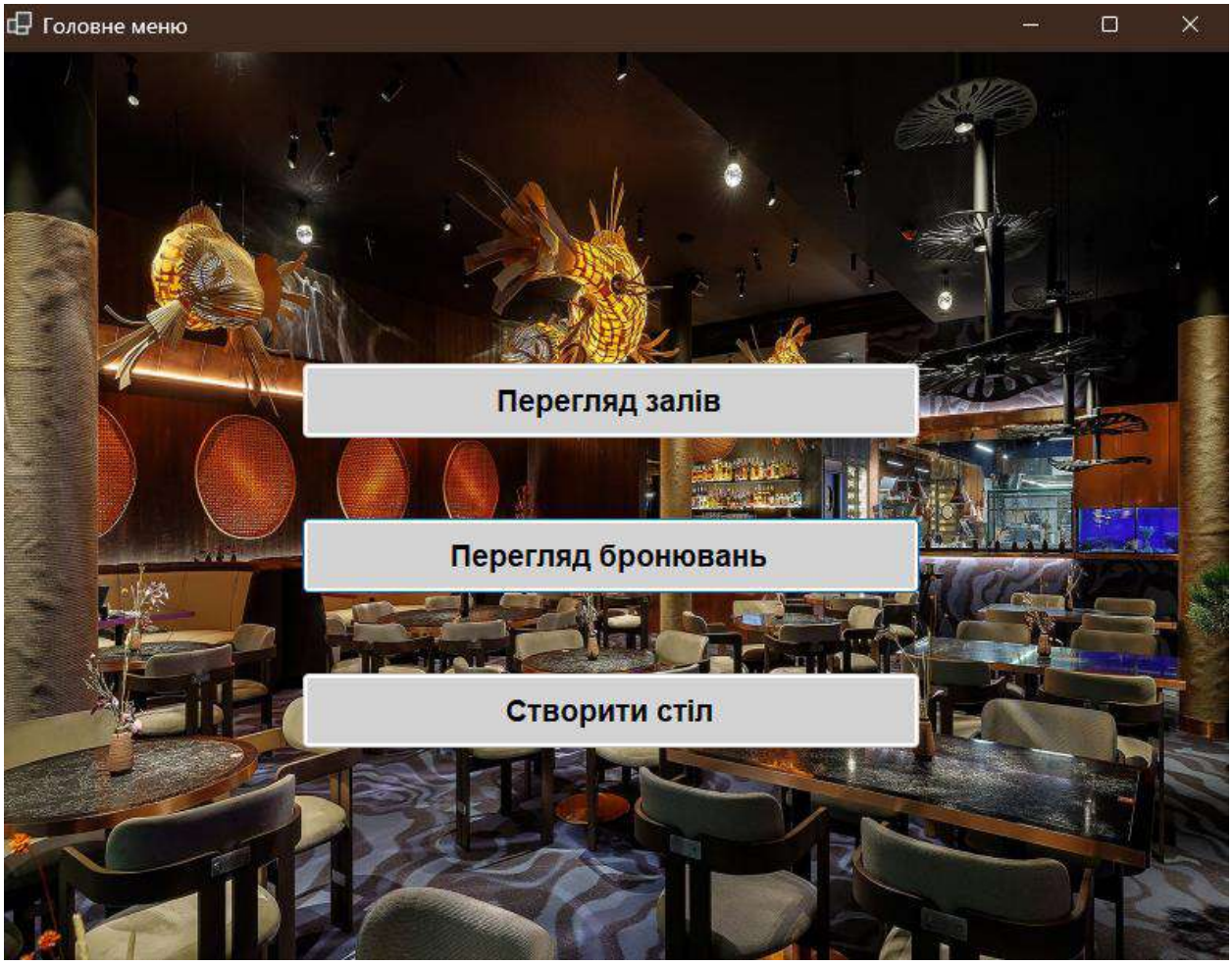

Рис. 4.1. Головне меню адмін-панелі

Перегляд залів — це візуальний модуль застосунку, який забезпечує графічне представлення інтер'єру ресторану. Головний елемент вікна з графічним представленням - openGL Control. Це компонент, вбудований у форму, який слугує вікном рендерингу тривимірної сцени. Для реалізації було

використано бібліотеку OpenTK (Open Toolkit Library) — це .NET-обгортка над OpenGL, що забезпечує зручний доступ до функцій графічного API у середовищі C# [2].

Компонент GLControl, який надається OpenTK, інтегрується у WinForms-додаток і дозволяє відображати 3D-графіку без необхідності запуску окремого рендерера. Він забезпечує низькорівневий доступ до графічного контексту OpenGL, у якому відбувається вся побудова сцени: ініціалізація проекції, налаштування освітлення, малювання примітивів, текстурування [3].

Після ініціалізації GLControl викликаються події OnLoad, OnResize та OnPaint, які відповідають за завантаження ресурсів, налаштування перспективної проекції відповідно до розмірів вікна та сам процес відображення сцени. У методі малювання (OnPaint) OpenGL отримує дані про об’єкти — їх координати, розміри та параметри — і використовує ці дані для побудови геометрії у тривимірному просторі.

Рендеринг сцени починається зі створення замкнутого простору, що представляє собою кімнату ресторану [Додаток Б]. Цей етап передбачає побудову базової геометрії приміщення: підлоги і стін, які задають межі сцени у віртуальному просторі. З ціллю гарної перегляданості кімнати, малюється лише три задні і бокові стіни кімнати. Після цього алгоритм надсилає запит до бази даних для отримання координат розташування столів та кількості посадкових місць за кожним із них.

На основі кількості місць виконується обчислення розміру столу. Це дозволяє динамічно розрахувати розмір моделі. Столи на 2, 4 чи 6 осіб мають різні розміри та форму. Отримані дані передаються до рендер-модуля через інтерфейс, який описує базову поведінку об'єктів візуалізації. Залежно від розміру або типу столу викликається відповідна реалізація інтерфейсу — клас, що відповідає за побудову конкретного розміру столу.

Використовуючи OpenGL, ці класи виконують побудову примітивів через quads, triangles, triangle fans, відповідно трансформуючи їх за

координатами та масштабом. Кожен стіл відображається в заданому місці кімнати з урахуванням своїх розмірів (Рис. 4.2.).

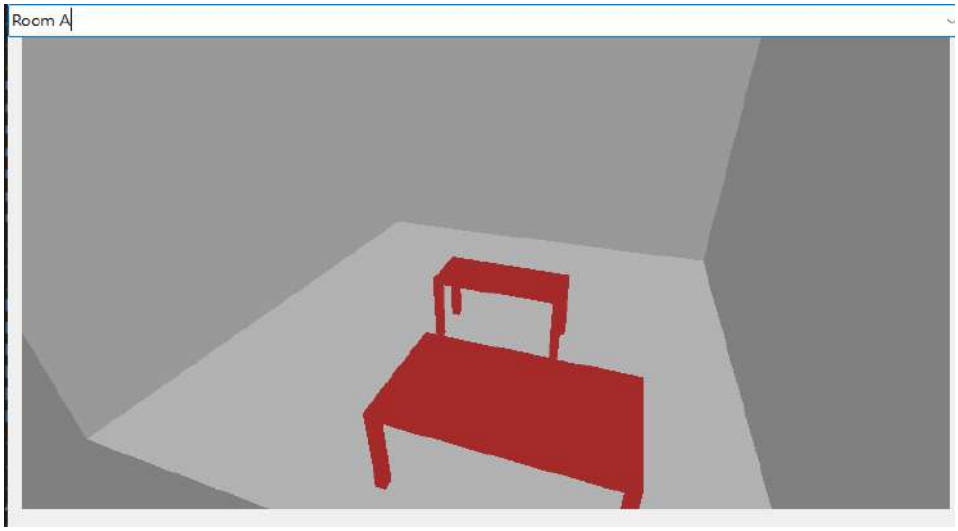

Рис. 4.2. Візуальний модуль залу ресторану

У верхній частині вікна розташований елемент керування ComboBox, який завантажує перелік доступних залів з бази даних. Кожен стіл у системі має прив’язку до конкретного залу, тому при виборі користувачем певної кімнати здійснюється запит до бази даних на отримання масиву відповідних координат та характеристик столів саме цього залу. Після отримання даних візуальний модуль очищує попередню сцену та динамічно відтворює нову конфігурацію — на тривимірній моделі кімнати з’являються лише ті столи, що належать до обраного залу (Рис. 4.3.).

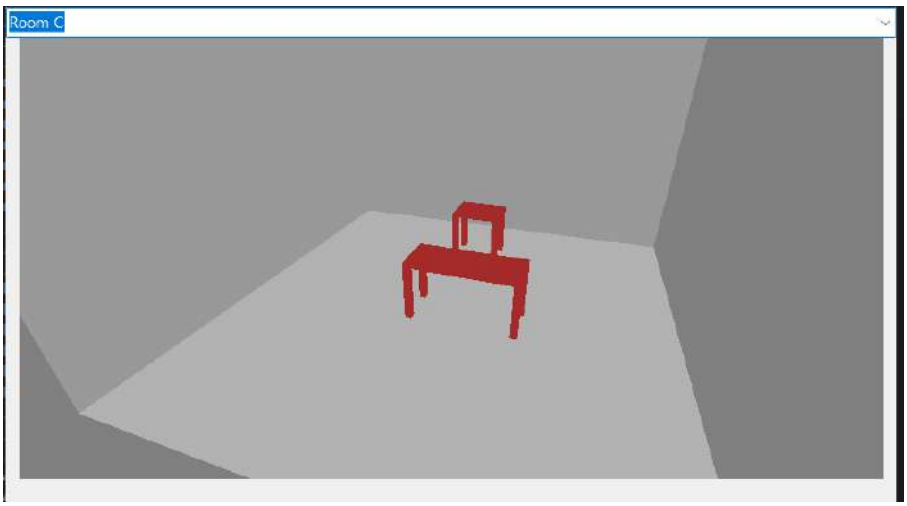

Рис. 4.3. Динамічна перемальовка столів в залежності від обраного залу

Важливо зазначити, що візуальний модуль відтворює лише схематичну модель залу. Тобто реалізація дуже проста: малюються тільки столи у відповідному до реального місці [Додаток А]. Такі деталі як проходи, вікна, аксесуари інтер'єру у реалізації не передбачені. У майбутньому планується додати обрабку наведення миші та кліку по столу, та надання інформації про час та дати бронювання столику.

Передбачено, що в залах може відбуватися перестановка, і столи отримуватимуть нові координати. Або ж можливо навіть будуть додаватися нові столи. Тоді персонал може додати стіл через відповідну форму (Рис. 4.4.).

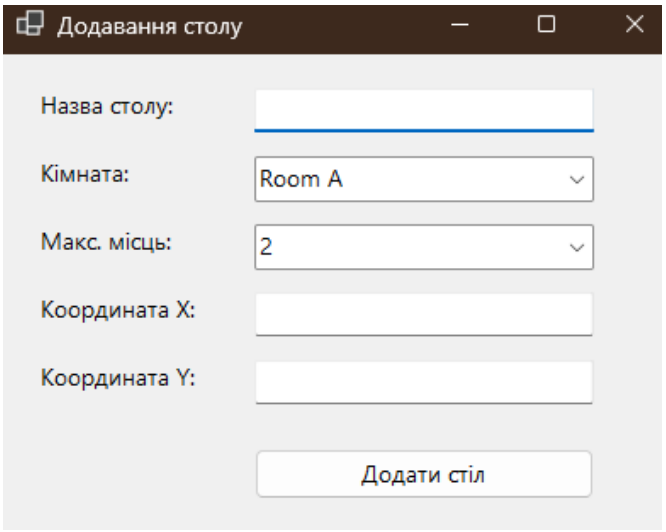

Рис. 4.4. Форма для створення нового столу

Вводиться уся необхідна для рендеру інформація:

- назва столу
- кімната в якій розташовується стіл
- координата по горизонталі (x)
- координата по вертикалі (y)

Форма також перевіряє коректність введення даних. Якщо працівник вводить не усі обов'язкові поля (Рис. 4.5.), або ж вводить координати які знаходяться за межами кімнати (Рис. 4.6.), виводиться відповідне повідомлення з проханням ввести коректні дані.

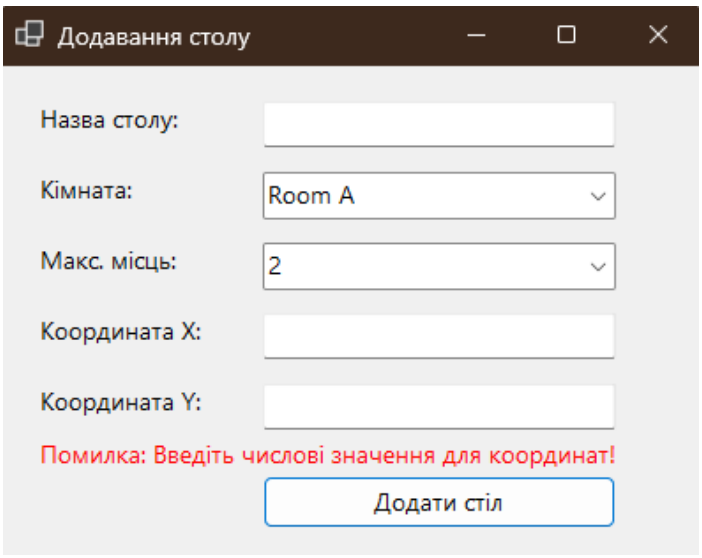

Рис. 4.5. Спроба додати стіл з порожніми даними

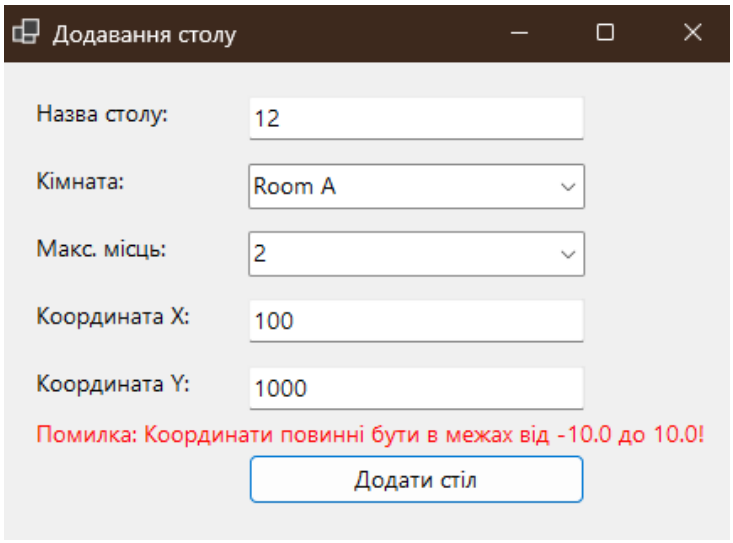

Рис. 4.6. Спроба додати стіл за межами кімнати

Якщо ж усі дані були введені вірно, працівник отримує повідомлення про успішне додавання столу, а форма закривається (Рис. 4.7.).

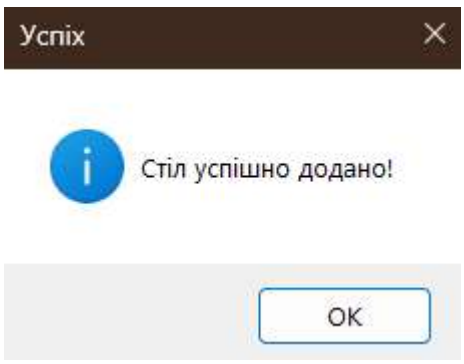

Рис. 4.7. Успішне додавання нового столу

Після цього новий стіл одразу же малюється у візуальному модулі, а дані про стіл записуються в базу даних (Рис. 4.8.).

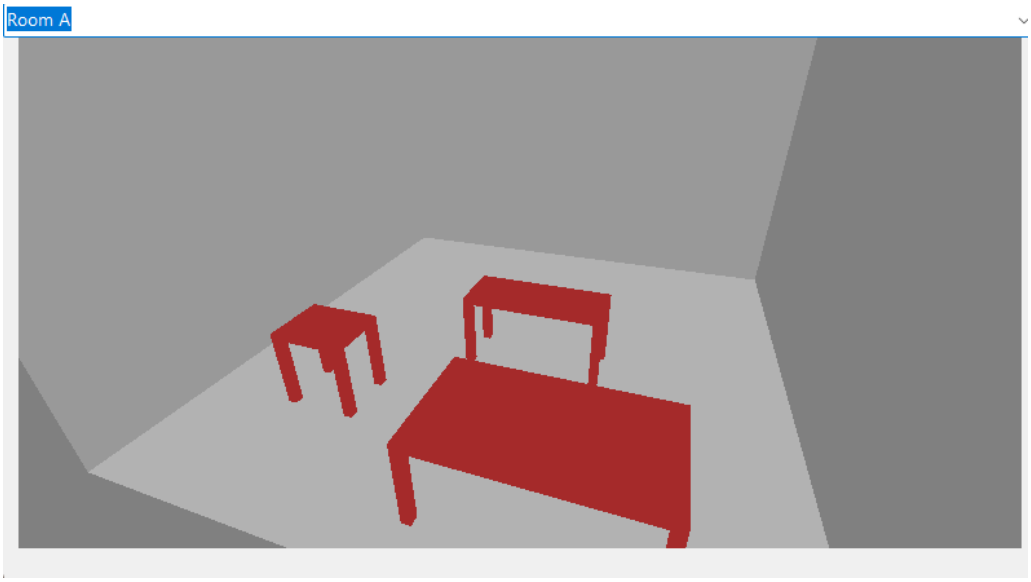

Рис. 4.8. Новий стіл додано у візуальний модуль

Окрім керування інтер’єром, працівники ресторану мають змогу переглядати інформацію про поточні бронювання та пов’язані з ними замовлення страв через окреме вікно з таблицею (гридом). Це вікно містить дві вкладки: одна відповідає за перегляд бронювань (Рис. 4.9.), інша — за відображення замовленої їжі для кожного з них (Рис. 4.10.). При перемиканні між вкладками вміст таблиці оновлюється динамічно, підвантажуючи дані з відповідних таблиць бази даних.

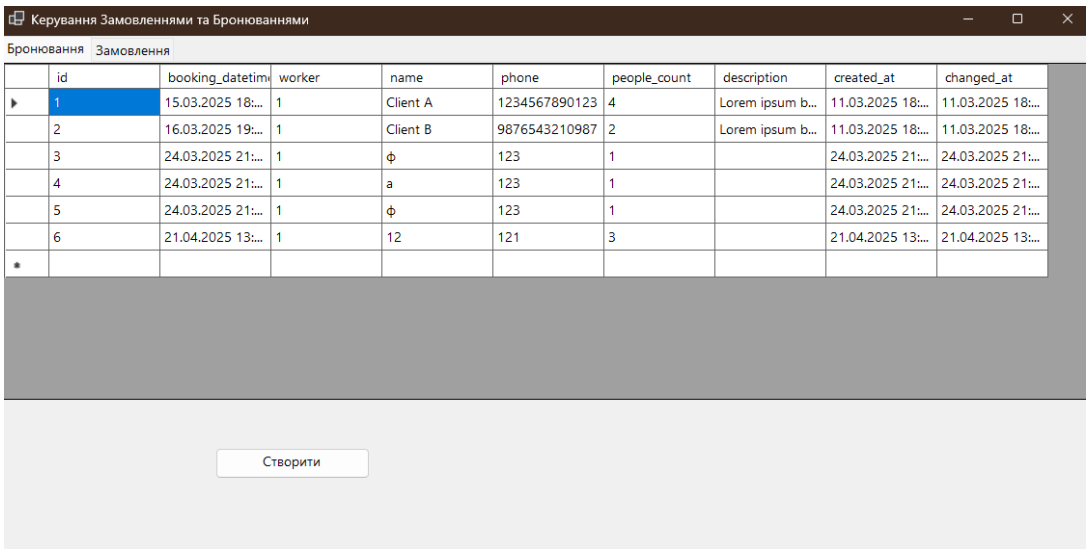

Рис. 4.9. Панель керування бронюваннями

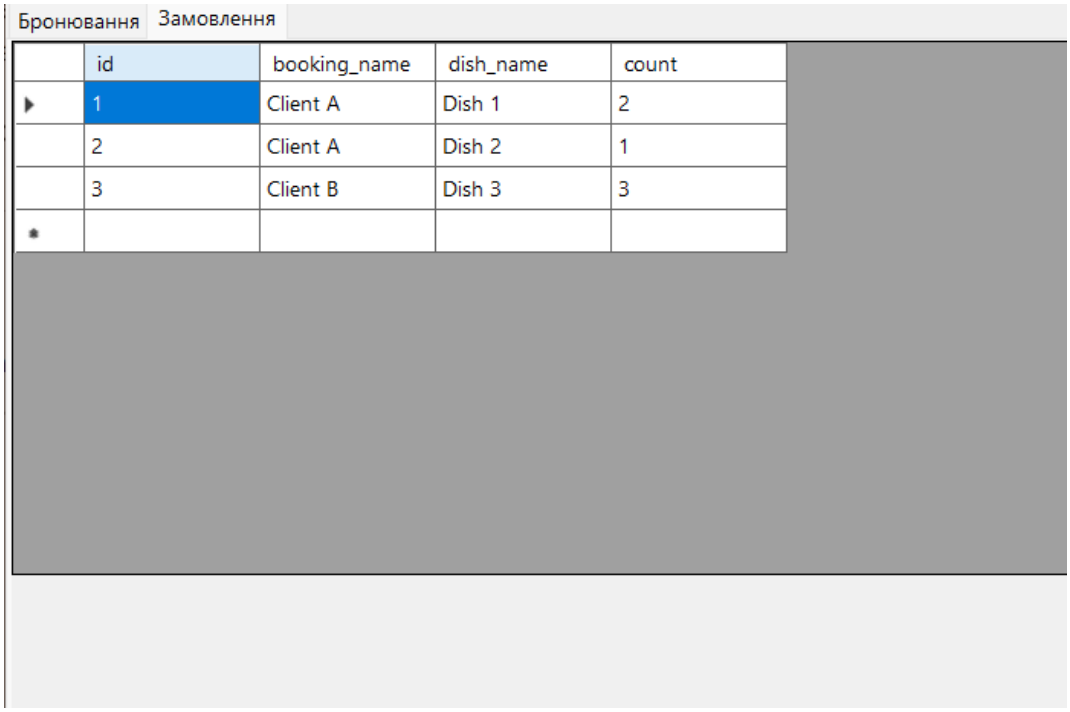

Рис. 4.10. Панель керування замовленнями

На вкладці керування бронюваннями можна помітити кнопку “Створити замовлення”. Ця кнопка відкриває діалогове вікно реєстрації бронювання яке відбувається в кілька етапів (Рис. 4.11.)

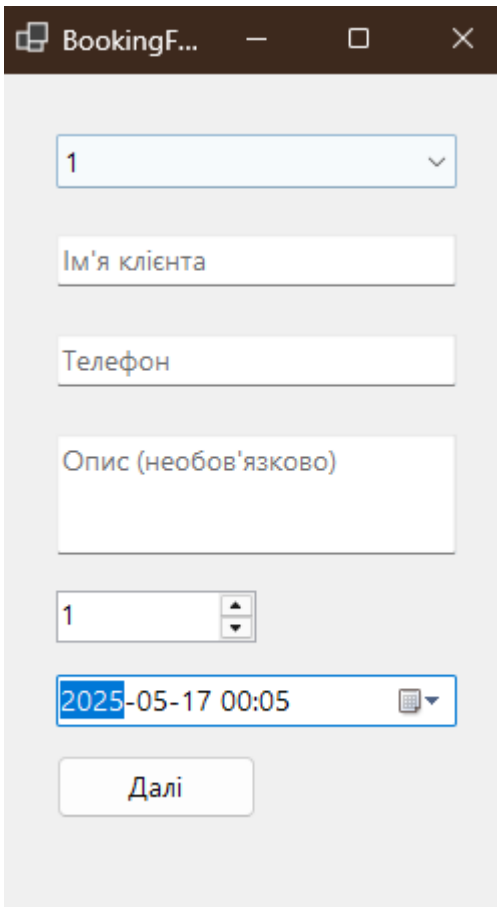

Рис. 4.11. Додавання нової броні

Спочатку безпосередньо заповнюється інформація про бронь:

робітник який створює замовлення

- ім’я клієнта
- телефон
- на яку кількість розрахований стіл
- дата та час бронювання

Коли працівник ввів валідні дані, він переходить до наступного етапу - створення замовлення до бронювання (Рис. 4.12.).

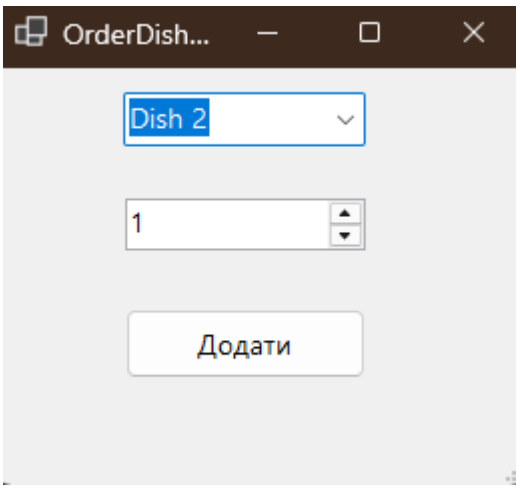

Рис. 4.12. Створення замовлення

Після створення бронювання та замовлення працівник отримує повідомлення про успіх (Рис. 4.13.).

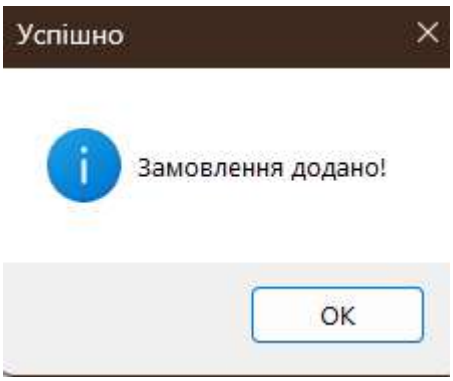

Рис. 4.13. Успішне створення замовлення

Після цього, нове замовлення одразу ж записується у базу даних та відображається на панелі (Рис. 4.14.).

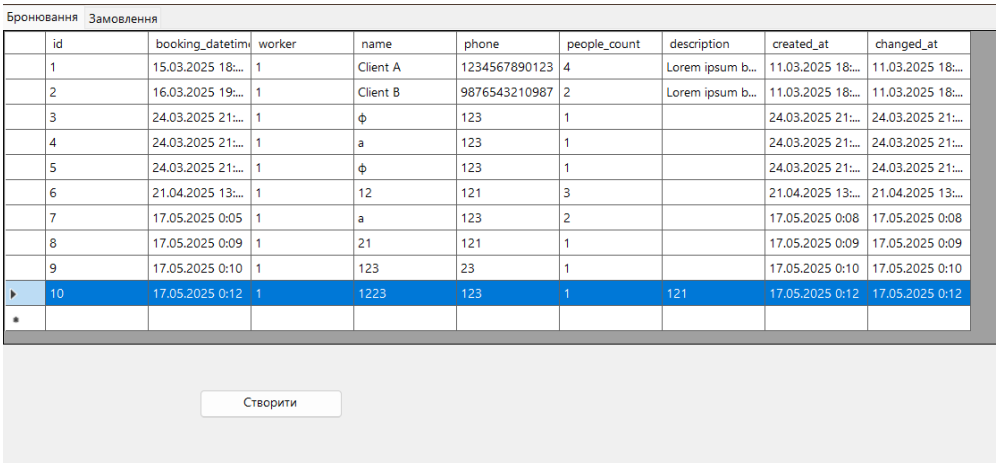

Рис. 4.14. Нове замовлення додано у базу даних

4.2. Розробка клієнтської частини сайту ресторану

Клієнтська частина сайту була реалізована з використанням бібліотеки React. Інтерфейс був створений інтуїтивно зрозумілим, візуально привабливим і зручним для користувачів як на мобільних, так і на десктопних пристроях. Основна логіка взаємодії реалізовувалась через набір візуальних компонентів, кожен з яких відповідав за окремий функціональний блок сайту.

На головній сторінці (MainPage) [Додаток Г] користувача зустрічає фоновий екран, реалізований у компоненті BackgroundScreen. У ньому використовуються анімовані лампи, слоган бренду та кнопка переходу до сторінки меню. Для плавних анімацій використовувалася бібліотека framer-motion (Рис 4.15.)

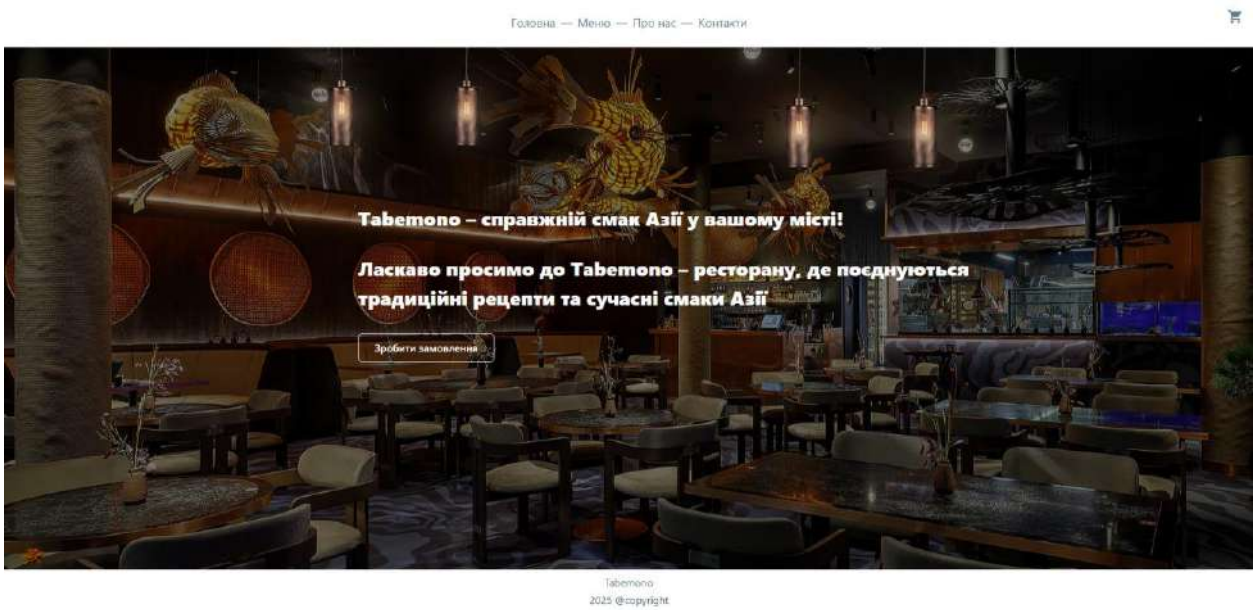

Рис. 4.15. Головна сторінка сайту з фоновим екраном та анімацією

На сторінці «Про нас» [Додаток Д] написана інформація про ресторан та його цінності. Нижче розміщуються картки (PhilosophyCard), які коротко презентують цінності закладу. Кожна картка містить заголовок, іконку та опис ідеї (Рис. 4.16.).

Рис. 4.16. Картки на сторінці «Про нас»

На сторінці меню (MenuPage) [Додаток Е] виводяться категорії страв за допомогою компонентів MenuCategoryItem. Після вибору певної категорії користувача автоматично направляє до блоку відповідних страв. Кожна страва відображається за допомогою компонента MenuItemCard, який містить зображення, назву, опис та ціну (Рис. 4.17.).

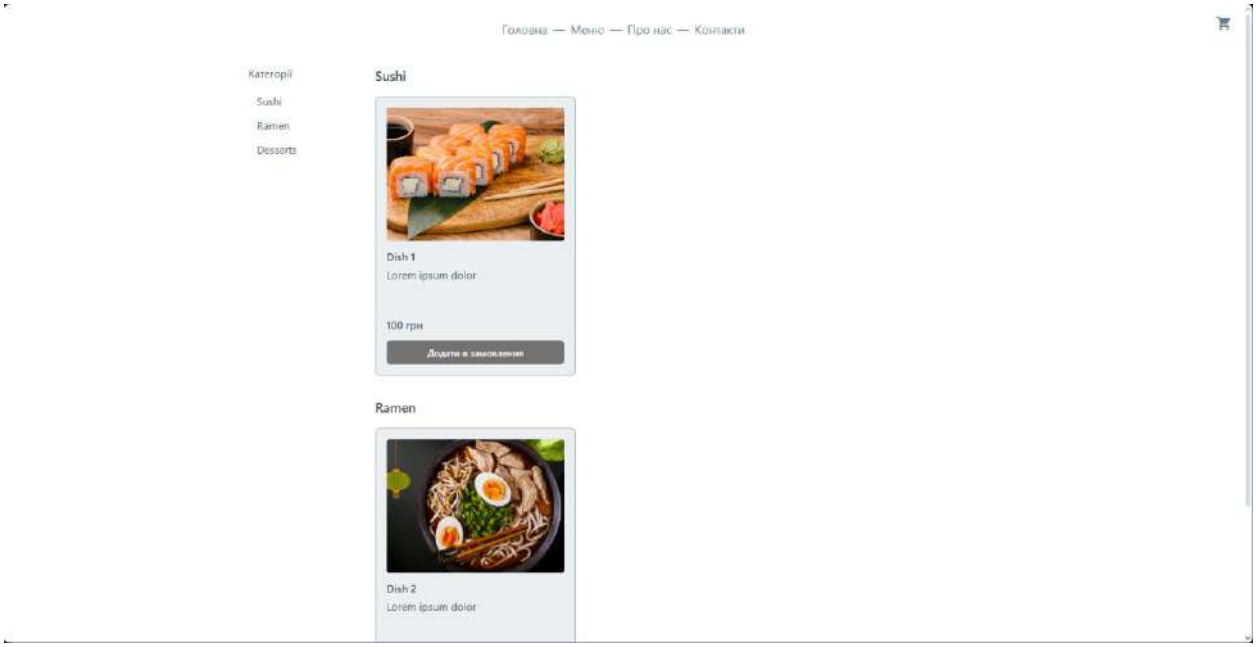

Рис. 4.17. Відображення меню з категоріями та стравами

Після вибору страв користувач може перейти до кошика (CartDrawer), який реалізований як бічна панель. У ньому можливо змінити кількість обраних страв, видалити позиції або перейти до оформлення замовлення. Кошик доступний з будь-якої сторінки, завдяки глобальному компоненту, що працює у парі з CartProvider (Рис. 4.18).

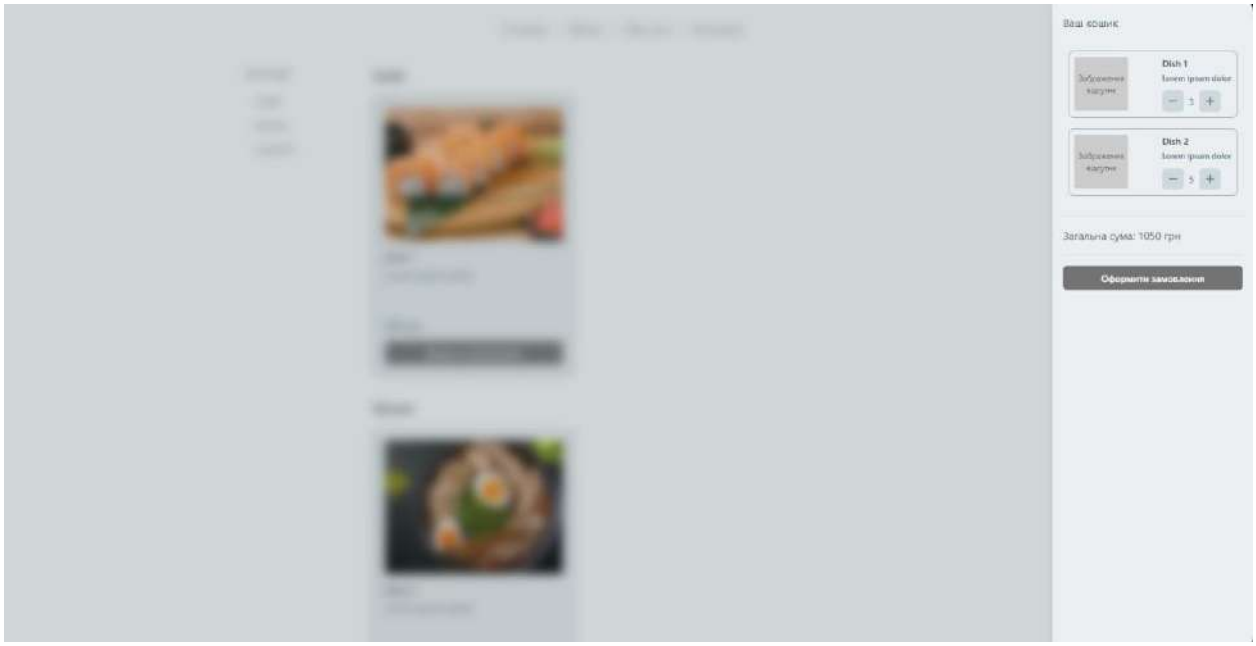

Рис. 4.18. Кошик з вибраними стравами

Оформлення замовлення здійснюється через модальне вікно (OrderModal), яке відкривається поверх інтерфейсу. У цьому вікні користувач має ввести ім’я, номер телефону та коментар. Реалізована клієнтська валідація — у випадку некоректного вводу поля підсвічується з відповідним повідомленням. Після підтвердження форма надсилає POST-запит на сервер, і у разі успіху виводиться повідомлення про успішне оформлення (Рис. 4.19.).

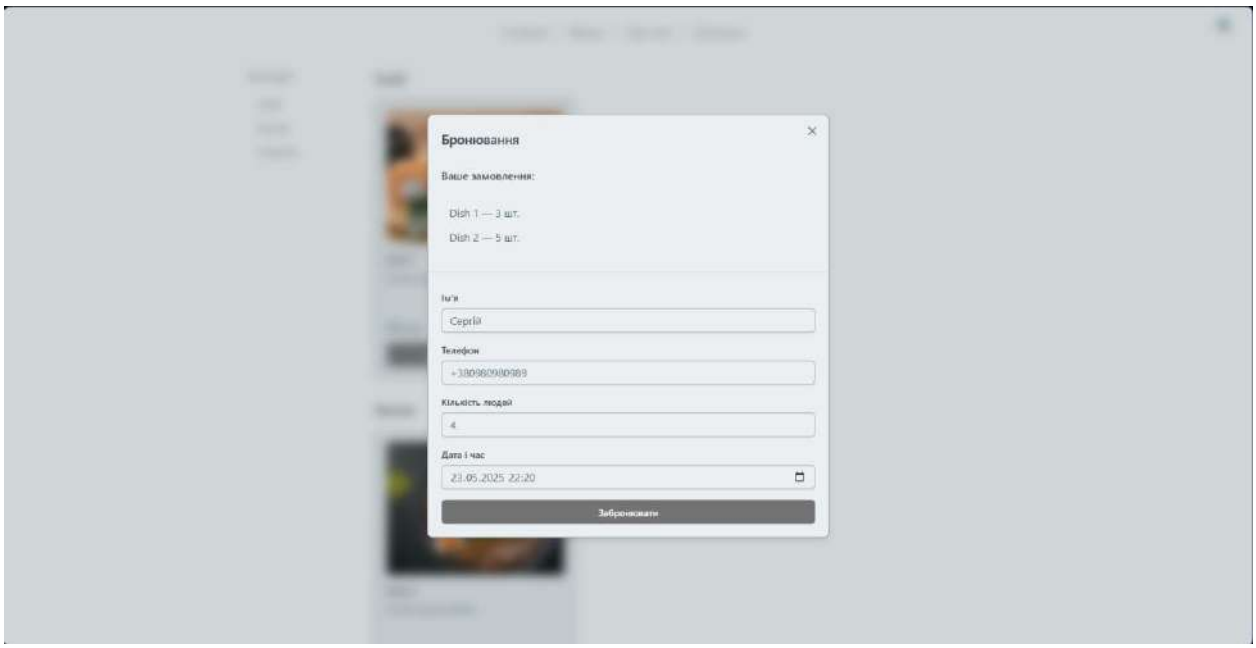

Рис. 4.19. Модальне вікно оформлення замовлення

На сторінці контактів (ContactsPage) [Додаток Ж] вбудована інтерактивна карта на основі react-leaflet, яка дозволяє переглянути місцезнаходження ресторанів. Поруч також розміщуються номери телефонів (Рис. 4.20.).

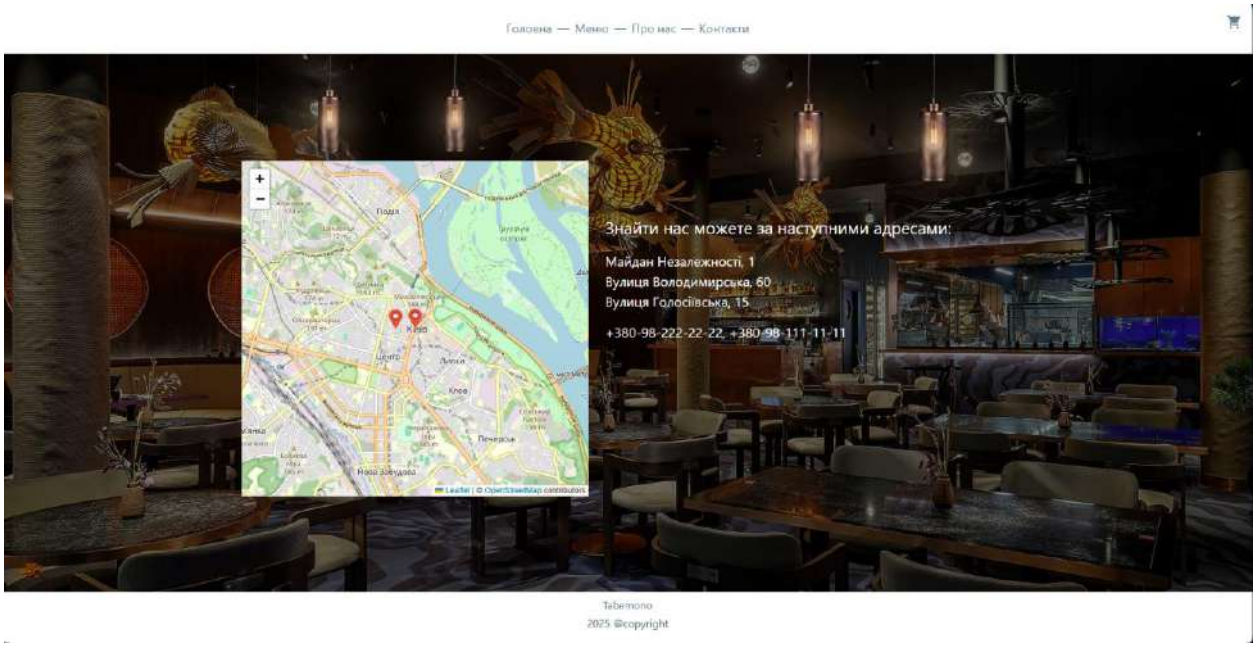

Рис. 4.20. Інтерактивна мапа із контактами ресторану

Кожна сторінка сайту використовує компоненти Header та Footer, які залишалися незмінними при переходах. Маршрутизація між сторінками здійснюється за допомогою react-router-dom.

Усі візуальні елементи були стилізовані з використанням бібліотек Joy UI, та @emotion/styled, що дозволило реалізувати зручний дизайн. Анімації додали плавності взаємодії та покращили користувацький досвід.

Висновки до розділу 4

У результаті розробки адміністративної частини було реалізовано зручну панель керування рестораном, що дозволяє працівникам у візуальному форматі взаємодіяти із залами та столами закладу. Було створено тривимірний рендеринг-модуль на базі OpenGL із використанням бібліотеки OpenTK, який забезпечує відображення структури приміщень і дозволяє персоналу наочно переглядати розміщення столів. Інтерфейс надає можливість динамічно перемальовувати сцену залежно від вибраного залу, а також було реалізовано створення нових столів із валідацією введених даних. Додатково була впроваджена система обліку бронювань і замовлень, що забезпечила зручну фіксацію нових заявок клієнтів із подальшим їх виведенням у таблицях управління. Уся реалізація підвищила ефективність внутрішніх процесів ресторану, зробивши керування простим і візуально доступним.

Клієнтська частина сайту ресторану була реалізована як повноцінний вебзастосунок на основі React із використанням бібліотек Joy UI, Framer Motion та Leaflet. Сайт забезпечив користувачам інтуїтивний доступ до основних функцій: перегляду меню, додавання страв до кошика та оформлення бронювань. Структура застосунку була розбита на окремі логічні компоненти, що відповідають за різні секції сторінки — головну, меню, замовлення, форму бронювання тощо. Для поліпшення досвіду користувача було додано анімації появи секцій, модальні вікна та інтерактивну мапу розташування ресторану.

ВИСНОВКИ

В результаті виконання даної кваліфікаційної роботи було досягнуто поставленої мети: розроблено діджитал-систему забезпечення замовлень клієнта у ресторані. Система створена за допомогою мови програмування C#, OpenGL, бібліотеки React для JavaScript та Node.js.

В результаті виконаної роботи можна зробити такі висновки:

1. У процесі планування цифрової системи для ресторану було визначено мету — створення програмного забезпечення, яке забезпечить автоматизацію обслуговування клієнтів. Сформульовані основні завдання: розробка візуальної та функціональної частини веб-сайту, інтеграція бази даних, розробка адміністративного модуля та модуля візуалізації залів ресторану.

2. Реалізовано адміністративну частину системи, яка дозволяє персоналу взаємодіяти з візуальним модулем, керувати бронюваннями та замовленнями. Створено форму для додавання нових столів, що забезпечує перевірку даних і динамічне оновлення сцени. Для рендерингу використано бібліотеку OpenTK на основі OpenGL, що дозволило створити тривимірне представлення залу з розміщенням столів.

3. Передбачено можливість додавання нових столів або зміни їх координат відповідно до реальних перестановок у залі. Візуалізація здійснюється за допомогою GLControl у середовищі WinForms, з підтримкою динамічної перемальовки сцени залежно від вибраного залу. Реалізовано вікно перегляду бронювань і пов’язаних із ними замовлень, що підвантажуються з бази даних у відповідні вкладки.

4. Розроблено зручний та функціональний веб-сайт, доступний у онлайн-режимі, який надає можливість клієнтам ресторану переглядати меню, обирати страви, дізнаватися про склад їжі, а також бронювати столики. Такий підхід дозволяє зробити замовлення швидким, наочним і доступним без потреби телефонного зв’язку.

Розроблена система охоплює як клієнтську, так і адміністративну частину, що дозволяє автоматизувати процес замовлення в ресторані. Завдяки використанню діджитал-технологій забезпечується швидке обслуговування клієнтів, ефективна робота персоналу та можливість подальшої аналітики діяльності закладу. Візуалізація залів і онлайн-бронювання створюють конкурентну перевагу на ринку ресторанного бізнесу. Створене програмне забезпечення повністю відповідає поставленим цілям, демонструє актуальність впровадження сучасних технологій і може використовуватись у реальних умовах роботи ресторану.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. POSist Technologies. Impact of Restaurant Technology on Customer Experience and Revenue. – [Електронний ресурс]. – Режим доступу: https://www.posist.com/restaurant-times/technology/impact-of-restaurant-technology.html
2. OpenTK Documentation. GLControl Class Overview – [Електронний ресурс]. – Режим доступу: https://opentk.net/learn/glcontrol/
3. Гамма Е., Хелм Р., Джонсон Р., Влісідес Дж. Шаблони проєктування. Елементи повторно використовуваного об'єктно-орієнтованого програмного забезпечення. – К.: Діалектика, 2020. – 366 с.
4. Стаднік В.В. Системи управління базами даних: навч. посіб. – К.: Каравела, 2021. – 296 с.
5. Романюк Є.М. Проєктування та розробка інформаційних систем: навч. посіб. – Львів: Видавництво ЛНУ імені Івана Франка, 2020. – 212 с.
6. Mozilla Developer Network. React Documentation. – [Електронний ресурс]. – Режим доступу: https://react.dev/
7. MySQL Documentation. MySQL 8.0 Reference Manual. – [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/
8. Microsoft Docs. Windows Forms Overview. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/
9. W3Schools. React Tutorial. – [Електронний ресурс]. – Режим доступу: https://w3schoolsua.github.io/react/index.html#gsc.tab=0
10. Кумар Т. Fluent React: Build Fast, Performant, and Intuitive Web Applications 1st Edition. – К.: Орейлі Медіа, 2024. – 334 с.

11. Сучасний підручник з JavaScript. – [Електронний ресурс]. – Режим доступу: https://uk.javascript.info
12. Вікіпедія. – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Головна_сторінка

ДОДАТКИ

Додаток А

Клас рендеру, який реалізує інтерфейс рендеру столу

```
    public abstract class BaseTableRenderer : ITableRenderer
    {
        protected float tableWidth;
        protected float tableDepth;
        protected float tableThickness;
        protected float legSize;
        protected float legHeight;
        protected ManagerDB dbManager = new ManagerDB();
        public BaseTableRenderer(float width, float depth, float
thickness, float legSize, float legHeight)
        {
            this.tableWidth = width;
            this.tableDepth = depth;
            this.tableThickness = thickness;
            this.legSize = legSize;
            this.legHeight = legHeight;
        }


        public virtual void DrawTable(float x, float y, float z,
float rotationX, float rotationY, float rotationZ)
        {
            GL.PushMatrix();
            GL.Translate(x, 0, y);
            GL.Rotate(rotationX, 1f, 0f, 0f);
            GL.Rotate(rotationY, 0f, 1f, 0f);
            GL.Rotate(rotationZ, 0f, 0f, 1f);

            GL.Color4(Color4.Brown);
            GL.Begin(PrimitiveType.Quads);

            DrawTabletop();
            GL.End();

            DrawLegs();

            GL.PopMatrix();
        }
        protected virtual void DrawTabletop()
        {
            float halfWidth = tableWidth / 2;
            float halfDepth = tableDepth / 2;
            float topY = tableThickness / 2;
            float bottomY = -tableThickness / 2;

            GL.Vertex3(-halfWidth, topY, halfDepth);
            GL.Vertex3(halfWidth, topY, halfDepth);
            GL.Vertex3(halfWidth, topY, -halfDepth);
            GL.Vertex3(-halfWidth, topY, -halfDepth);

            GL.Vertex3(-halfWidth, bottomY, halfDepth);
            GL.Vertex3(halfWidth, bottomY, halfDepth);
```

```
            GL.Vertex3(halfWidth, bottomY, -halfDepth);
            GL.Vertex3(-halfWidth, bottomY, -halfDepth);

            DrawSide(-halfWidth, bottomY, halfDepth, topY);
            DrawSide(halfWidth, bottomY, halfDepth, topY);
            DrawSide(-halfWidth, bottomY, -halfDepth, topY);
            DrawSide(halfWidth, bottomY, -halfDepth, topY);
        }

        private void DrawSide(float x, float bottomY, float z, float
topY)
        {
            GL.Vertex3(x, bottomY, z);
            GL.Vertex3(x, topY, z);
            GL.Vertex3(x, topY, -z);
            GL.Vertex3(x, bottomY, -z);
        }
        protected virtual void DrawLegs()
        {
            float topY = -tableThickness / 2;
            DrawLeg(-tableWidth / 2, topY, tableDepth / 2);
            DrawLeg(tableWidth / 2 - legSize, topY, -tableDepth / 2 +
legSize);
            DrawLeg(-tableWidth / 2, topY, -tableDepth / 2 +
legSize);
            DrawLeg(tableWidth / 2 - legSize, topY, tableDepth / 2);
        }

        protected virtual void DrawLeg(float x, float y, float z)
        {
            float bottomY = y - legHeight;
            GL.Begin(PrimitiveType.Quads);
            GL.Color4(Color4.Brown);

            GL.Vertex3(x, y, z);
            GL.Vertex3(x + legSize, y, z);
            GL.Vertex3(x + legSize, bottomY, z);
            GL.Vertex3(x, bottomY, z);

            GL.Vertex3(x, y, z - legSize);
            GL.Vertex3(x + legSize, y, z - legSize);
            GL.Vertex3(x + legSize, bottomY, z - legSize);
            GL.Vertex3(x, bottomY, z - legSize);

            GL.Vertex3(x, y, z);
            GL.Vertex3(x, y, z - legSize);
            GL.Vertex3(x, bottomY, z - legSize);
            GL.Vertex3(x, bottomY, z);

            GL.Vertex3(x + legSize, y, z);
            GL.Vertex3(x + legSize, y, z - legSize);
            GL.Vertex3(x + legSize, bottomY, z - legSize);
            GL.Vertex3(x + legSize, bottomY, z);

            GL.Vertex3(x, y, z);
            GL.Vertex3(x + legSize, y, z);
```

```
            GL.Vertex3(x + legSize, y, z - legSize);
            GL.Vertex3(x, y, z - legSize);

            GL.Vertex3(x, bottomY, z);
            GL.Vertex3(x + legSize, bottomY, z);
            GL.Vertex3(x + legSize, bottomY, z - legSize);
            GL.Vertex3(x, bottomY, z - legSize);

            GL.End();
        }
    }
```

Додаток Б

Клас рендеру кімнати

```
public class RoomRenderer
{
    private float width;
    private float height;
    private float depth;

    public RoomRenderer(float width, float height, float depth)
    {
        this.width = width;
        this.height = height;
        this.depth = depth;
    }

    public void DrawRoom()
    {
        GL.PushMatrix();
        GL.Color3(0.7f, 0.7f, 0.7f);

        GL.Begin(PrimitiveType.Quads);
        GL.Vertex3(-width, -3, -depth);
        GL.Vertex3(width, -3, -depth);
        GL.Vertex3(width, -3, depth);
        GL.Vertex3(-width, -3, depth);
        GL.End();

        GL.Color3(0.6f, 0.6f, 0.6f);
        GL.Begin(PrimitiveType.Quads);
        GL.Vertex3(-width, -3, -depth);
        GL.Vertex3(width, -3, -depth);
        GL.Vertex3(width, height, -depth);
        GL.Vertex3(-width, height, -depth);
        GL.End();

        GL.Begin(PrimitiveType.Quads);
        GL.Vertex3(-width, -3, -depth);
        GL.Vertex3(-width, -3, depth);
        GL.Vertex3(-width, height, depth);
        GL.Vertex3(-width, height, -depth);
        GL.End();


        GL.PopMatrix();
    }
}
```

Додаток В

Клас вікна з графічним модулем

```
public partial class TableViewForm : Form
{

    private Table table;
    private bool isDragging = false;
    private Point lastMousePosition;
    private ITableRenderer tableRenderer;
    private List<Table> tables = new List<Table>();
    private ManagerDB dbManager = new ManagerDB();
    private RoomRenderer roomRenderer;

    public TableViewForm()
    {
        InitializeComponent();
        LoadTablesFromDB(1);
        LoadRooms();
        this.KeyPreview = true;
        roomRenderer = new RoomRenderer(10f, 10f, 10f);

        glControl1.Load += glControl1_Load;
        glControl1.Paint += glControl1_Paint;

    }

    private void LoadTablesFromDB(int roomId)
    {
        tables = dbManager.GetTablesByRoom(roomId);
    }

    private void glControl1_Load(object sender, EventArgs e)
    {
        GL.Enable(EnableCap.DebugOutput);
        GL.Enable(EnableCap.Texture2D);

        GL.ClearDepth(1.0f);
        GL.Enable(EnableCap.DepthTest);

        GL.MatrixMode(MatrixMode.Projection);
        GL.LoadIdentity();

        float aspectRatio = (float)glControl1.ClientSize.Width /
(float)glControl1.ClientSize.Height;
        Matrix4 projection =
Matrix4.CreatePerspectiveFieldOfView(
            MathHelper.DegreesToRadians(60f),
            aspectRatio,
            0.1f, 100f
        );

        GL.MatrixMode(MatrixMode.Projection);
        GL.LoadMatrix(ref projection);
```

```
            GL.MatrixMode(MatrixMode.Modelview);

            glControl1.Invalidate();
        }

        private void glControl1_Paint(object sender, PaintEventArgs
e)
        {
            glControl1.MakeCurrent();
            GL.Viewport(0, 0, glControl1.ClientSize.Width,
glControl1.ClientSize.Height);
            GL.ClearColor(Color4.Gray);
            GL.Clear(ClearBufferMask.ColorBufferBit |
ClearBufferMask.DepthBufferBit);

            Matrix4 lookAt = Matrix4.LookAt(
                new Vector3(5, 8, 15),
                new Vector3(0, 0, 0),
                Vector3.UnitY
            );
            GL.LoadMatrix(ref lookAt);

            roomRenderer.DrawRoom();

            foreach (var table in tables)
            {

                switch (dbManager.GetTableSizeById(table.Id))
                {
                    case 2:
                        tableRenderer = new TwoPersonTableRenderer();
                        break;
                    case 4:
                        tableRenderer = new
FourPersonTableRenderer();
                        break;
                    case 6:
                        tableRenderer = new SixPersonTableRenderer();
                        break;
                    default:
                        tableRenderer = new
FourPersonTableRenderer();
                        break;
                }
                tableRenderer.DrawTable(table.X, table.Y, 0, 0, 0,
0);
            }

            glControl1.SwapBuffers();
        }

        private void GlControl1_MouseDown(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left) // ЛКМ натиснута
            {
```

```
                isDragging = true;
                lastMousePosition = e.Location;
            }
        }

        private void LoadRooms()
        {
            DataTable rooms = dbManager.GetRooms();
            roomComboBox.DataSource = rooms;
            roomComboBox.DisplayMember = "name";
            roomComboBox.ValueMember = "id";

            if (rooms.Rows.Count > 0)
                roomComboBox.SelectedIndex = 0;
        }
        private void RoomComboBox_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (roomComboBox.SelectedValue is int roomId)
            {
                LoadTablesFromDB(roomId);
                glControl1.Invalidate();
            }
        }
    }
```

Додаток Г

Головна сторінка сайту

```
import { Box, Button, Typography } from '@mui/joy'
import React from 'react'
import { BackgroundScreen } from '../../components'
function MainPage() {
  return (
    <Box>
      <BackgroundScreen/>
      <Box
        sx={{
          pt: 30,
          ml: 20
        }}>
      <Typography
          sx={{
            color: 'white',
            fontSize:"30px",
            fontWeight: "1000"
          }}
        >
          Tabemono – справжній смак Азії у вашому місті!
        </Typography>
        <Typography
          sx={{
            color: 'white',
            fontSize:"30px",
            pt:4,
            fontWeight: "1000"
          }}
        >
          Ласкаво просимо до Tabemono – ресторану, де поєднуються
традиційні рецепти та сучасні смаки Азії
        </Typography>
        <Box>
          <Button variant='outlined' size="lg"
            sx={(theme) => ({
              backgroundColor: "",
              color: "white",
              mt:4,
              transition: ".3s",
              "&:hover": {
                backgroundColor: "grey",
              },
            })}
          >
            Зробити замовлення
          </Button>
        </Box>
      </Box>
    </Box>
  )}

export default MainPage
```

Додаток Д

Сторінка сайту «Про нас»

```
      import { Box, Button, Card, Grid, Typography } from '@mui/joy'
      import React from 'react'
      import { BackgroundScreen, PhilosophyCard } from
'../../components'
      import { img } from 'framer-motion/client'
      function AboutPage() {
        const cards = [
          {title: 'Суші та роли, приготовані за традиційною японською
технологією', img: "https://cdn-media.choiceqr.com/prod-eat-
asia/menu/JFHFIDI-FQKiHia-IDzqJIt.webp"},
          {title: 'Ароматні супи – від насиченого рамену до легкого
місо', img: "https://cdn-media.choiceqr.com/prod-eat-
asia/menu/thumbnail_HxubXJe-eNTfDIo-lvuJtrY_original.jpeg_k-M-
B.webp"},
          {title: 'Вок-страви, що поєднують яскраві смаки та корисні
інгредієнти', img: "https://cdn-media.choiceqr.com/prod-eat-
asia/menu/thumbnail_LAaAqlH-VWFrfCA-omSZtDg_original.jpeg_Q-S-
a.webp"},
          {title: 'Десерти та напої, які доповнять ваш гастрономічний
досвід', img: "https://cdn-media.choiceqr.com/prod-eat-
asia/menu/thumbnail_HVgFPEQ-RfoFHsc-JcgGLHo_i-O-L.webp"},
        ]

        return (
          <Box>
            <BackgroundScreen/>
            <Box
              sx={{
                pt: 30, }}
            >
            <Typography
                sx={{
                  color: 'white',
                  fontSize:"30px",
                  fontWeight: "1000"}}>
                Ресторан Tabemono – це місце, де кожна страва
розповідає свою історію. Ми об'єднали традиції Японії, Китаю та
Таїланду, щоб створити унікальний простір для гурманів азійської кухні
              </Typography>
              <Typography
                sx={{
                  color: 'white',
                  fontSize:"30px",
                  pt:4,
                  fontWeight: "1000"}}>
              </Typography>
              <Box mt={10}>
                <Typography fontSize={40} sx={{color: 'white'}}>
                Наша філософія
```

```
                </Typography>
                <Typography fontSize={30} mt={2} sx={{color: 'white'}}>

                У Tabemono ми віримо, що їжа – це не просто спосіб
втамувати голод, а справжнє мистецтво. Ми готуємо з душею,
використовуючи тільки свіжі інгредієнти, автентичні спеції та класичні
рецепти в сучасному виконанні
                </Typography>
              </Box>
              <Box mt={28}>
                <Typography fontSize={40}>
                  Що на вас чекає?
                </Typography>
                <Grid container width={"100%"} justifyContent={"space-
between"} mt={2}>
                  {
                    cards.map((item)=>(
                      <Grid>
                        <PhilosophyCard title={item.title}
img={item.img}/>
                      </Grid>
                    ))
                  }
                </Grid>
              </Box>
              <Box mt={5}>
              <Typography fontSize={40}>
                Атмосфера та сервіс
                </Typography>
                <Typography fontSize={20}>
                Ми подбали про затишну атмосферу, щоб кожен гість
відчув себе комфортно. Наш персонал завжди радий допомогти вам обрати
ідеальну страву та поділитися історією її створення.
                </Typography>
                <Typography fontSize={20} mt={2} mb={10}>
                Tabemono – це більше, ніж ресторан. Це місце для
зустрічей, нових відкриттів і справжнього задоволення від азійської
кухні. Завітайте до нас та відкрийте для себе смак Азії! </Typography>
              </Box>
            </Box>
          </Box>
        )
      }

      export default AboutPage
```

Додаток Е

Сторінка сайту з меню

```
import React, { useEffect, useRef, useState } from 'react';
import {
  Grid,
  Typography,
  List,
  ListItem,
  ListItemButton,
  CircularProgress,
} from '@mui/joy';
import MenuCategoryItem from '../../components/MenuCategoryItem';

const MenuPage = () => {
  const [dishes, setDishes] = useState([]);
  const [categories, setCategories] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const categoryRefs = useRef({});

  useEffect(() => {
    const fetchMenu = async () => {
      try {
        const res = await
fetch('http://localhost:5000/api/menu');
        if (!res.ok) throw new Error('Failed to fetch menu');
        const data = await res.json();
        setCategories(data.categories);
        setDishes(data.dishes);
      } catch (err) {
        setError(err.message || 'Error');
      } finally {
        setLoading(false);
      } };

    fetchMenu();
  }, []);

  const handleScrollToCategory = (categoryId) => {
    const el = categoryRefs.current[categoryId];
    if (el) {
      el.scrollIntoView({ behavior: 'smooth', block: 'start' });
    }};
  if (loading) return <CircularProgress />;
  if (error) return <Typography
color="danger">{error}</Typography>;

  return (
    <Grid container spacing={2} sx={{ p: 2 }}>
      {/* Сайдбар */}
      <Grid
        xs={12}
        sm={2}
        sx={{
```

Продовження додатку Е

```
                position: 'sticky',
                top: 16,
                alignSelf: 'flex-start',
                height: 'fit-content',
              }}
            >
              <Typography level="h5" gutterBottom>
                Категорії
              </Typography>
              <List>
                {categories.map(cat => (
                  <ListItem key={cat.id}>
                    <ListItemButton onClick={() =>
handleScrollToCategory(cat.id)}>
                      {cat.name}
                    </ListItemButton>
                  </ListItem>
                ))}
              </List>
            </Grid>

            {/* Меню по категоріях */}
            <Grid xs={12} sm={10}>
              {categories.map(cat => {
                const filteredDishes = dishes.filter(d => d.category
=== cat.id);
                return (
                  <MenuCategoryItem
                    key={cat.id}
                    categoryName={cat.name}
                    dishes={filteredDishes}
                    categoryRef={el => (categoryRefs.current[cat.id] =
el)}
                  />
                );
              })}
            </Grid>
          </Grid>
        );
      };

      export default MenuPage;
```

Додаток Ж

Сторінка сайту з контактами

```
import { Box, Grid, Typography } from '@mui/joy'
import React from 'react'
import { BackgroundScreen } from '../../components'
import {Map} from '../../components'

function ContactsPage() {

  const adresses = ["Майдан Незалежності, 1", "Вулиця
Володимирська, 60", "Вулиця Голосіївська, 15"]

  return (
    <Box>
      <BackgroundScreen/>
      <Grid container gap={3} sx={{height: "500px"}} mt={20} >
        <Grid width={"45%"}>
            <Map/>
        </Grid>
        <Grid mt={10} width={"45%"}>
          <Typography sx={{color: "white", fontSize: "25px"}}
mb={2}>
            Знайти нас можете за наступними адресами:
          </Typography>
          {
            adresses.map((adress)=>(
              <Typography sx={{color: "white"}} fontSize={20}>
                {adress}
              </Typography>
            ))
          }
            <Typography sx={{color: "white"}} fontSize={20}
mt={2}>
              +380-98-222-22-22, +380-98-111-11-11
            </Typography>
        </Grid>
      </Box>
  )
}

export default ContactsPage
```