НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

_____Ігор БОЛБОТ
(підпис)

«____» ____2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри інформаційних систем і технологій

_____ Михайло ШВИДЕНКО
(підпис)

«____» ____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему: «Система бронювання столику в ресторані»

Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми _____ **Роман РУДЕНСЬКИЙ**
к.т.н., доцент (підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Роман ЗОЛОТУХА**
ст.викладач (підпис)

Виконав _____ **Марія ДРОБЯЗКО**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій
к.т.н., доцент___Михайло ШВИДЕНКО
(підпис)

«____» ____2026 р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту
Дробязко Марії Володимирівни

Спеціальність 122 – «Комп'ютерні науки»

ОП «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Система бронювання столику в ресторані» затверджена наказом ректора НУБіП України від 06.01.2026 №46 «С»

2. Термін подання завершеної роботи на кафедру_____2026.05.22_____
(рік, місяць, число)

3. Вихідні дані до роботи:

механізм функціонування інформаційної системи бронювання столиків у ресторані, включаючи реєстрацію користувачів, створення та підтвердження бронювання, облік доступності столиків, збереження даних про замовлення, а також формування звітності й аналітики через веб-платформу.

4. Перелік питань, що розглядаються:

Аналіз взаємодії клієнтів, адміністратора та персоналу ресторану із системою.

Проєктування інформаційного забезпечення системи бронювання столиків.

Розробка програмного забезпечення системи управління бронюваннями.
Впровадження та експлуатація системи в діяльності ресторану

Дата видачі завдання «_» ____2026 р.

Керівник бакалаврської кваліфікаційної робота _____ **Роман ЗОЛОТУХА**
(підпис)

Завдання взяв до виконання _____ **Марія ДРОБЯЗКО**
(підпис)

Зміст

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 5

ВСТУП 7

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9

1.1 Опис предметної області 9

1.2 Аналіз існуючих рішень 11

1.3 Моделювання предметної області 16

1.4 Аналіз вимог розроблюваної системи 20

1.5 Постановка завдання 22

1.6 Висновки до першого розділу 24

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 25

2.1 Логічна модель даних у вигляді ER-діаграми 25

2.2 Діаграма класів і кооперації 27

2.3 Діаграма компонентів 31

2.4 Діаграма пакетів 34

2.5 Висновки до другого розділу 37

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ 40

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації 40

3.2 Представлення архітектури системи 44

3.3 Фізична модель даних 47

3.4 Висновки до третього розділу 51

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ 54

4.1 Тестування програмних модулів CRM-системи бронювання столиків у ресторані 54
4.2 Тестування системи 58
4.3 Розгортання системи, склад інсталяційного пакету 66
4.4 Висновки до четвертого розділу 69
ВИСНОВКИ 71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 74
ДОДАТОК А 77
ДОДАТОК Б 79
ДОДАТОК В 82
ДОДАТОК Г 87

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API – Application Programming Interface, програмний інтерфейс взаємодії компонентів.
2. ARIMA – AutoRegressive Integrated Moving Average, авторегресійно-інтегрована модель середнього ковзання.
3. AS – Anomaly Severity, метричний показник ступеня аномальності даних.
4. CPU – Central Processing Unit, центральний процесор.
5. CSV – Comma-Separated Values, формат табличних текстових даних.
6. DAQ – Data Acquisition, процес збирання даних із сенсорних вузлів.
7. DB – Database, база даних.
8. DQI – Data Quality Index, індекс якості даних.
9. EMQX – високопродуктивний MQTT-брокер для IoT-систем.
10. ESP32 – мікроконтролер IoT-шлюзу сенсорних вузлів.
11. FNN – Feedforward Neural Network, багаторівнева нейронна мережа прямого поширення.
12. HTML – HyperText Markup Language, мова розмітки веб-документів.
13. HTTP(S) – HyperText Transfer Protocol (Secure), протокол передавання гіпертекстових даних.
14. IoT – Internet of Things, інтернет речей.
15. JSON – JavaScript Object Notation, формат структурованих даних.
16. K-means – метод кластеризації на основі середніх значень.
17. LSTM – Long Short-Term Memory, довготривала рекурентна пам’ять для прогнозування часових рядів.
18. Lux – одиниця вимірювання інтенсивності освітленості.
19. MAE – Mean Absolute Error, середня абсолютна похибка моделі.
20. MQTT – Message Queuing Telemetry Transport, легковаговий брокерський протокол обміну телеметрією.

21. ORM – Object-Relational Mapping, технологія відображення об’єктів у реляційну БД.

22. PCA – Principal Component Analysis, метод головних компонент для зниження розмірності даних.

23. PDF – Portable Document Format, формат електронних документів.
PHI – Plant Health Index, інтегральний індекс стану рослини.

24. pH – водневий показник кислотності ґрунту.

25. RAM – Random Access Memory, оперативна пам’ять.

26. REST – Representational State Transfer, архітектурний стиль веб-взаємодії.

27. RMSE – Root Mean Square Error, середньоквадратична похибка прогнозної моделі.

ВСТУП

Актуальність теми зумовлена необхідністю підвищення ефективності управління процесами обслуговування клієнтів у закладах ресторанного господарства в умовах цифровізації сфери послуг. Сучасна діяльність ресторанів характеризується зростанням потоку відвідувачів, ускладненням організації посадки гостей та необхідністю оперативного планування використання ресурсів. Традиційні підходи до бронювання столиків, що ґрунтуються на телефонних замовленнях або ручному обліку, не забезпечують достатнього рівня оперативності, прозорості та контролю, що може призводити до перевантаження закладу, втрати клієнтів і зниження якості сервісу. Використання інформаційних систем бронювання, інтегрованих із базами даних, вебінтерфейсами та аналітичними модулями, створює нові можливості для автоматизації ресторанних процесів, оптимізації завантаженості столиків та підвищення рівня клієнтського сервісу на основі актуальних даних.

Метою дослідження є розроблення інформаційної системи бронювання столиків у ресторані, що забезпечує автоматизований облік бронювань, управління доступністю столиків, взаємодію з клієнтами та підтримку прийняття управлінських рішень у режимі реального часу.

Для досягнення поставленої мети необхідно реалізувати функціональні механізми створення, редагування та скасування бронювань, управління інформацією про столики та часові інтервали, а також забезпечити взаємодію клієнтської та серверної частин системи з використанням бази даних і вебінтерфейсу.

У рамках поставленої мети передбачено розв’язання таких **завдань**:

– проаналізувати предметну область ресторанного обслуговування та існуючі підходи до бронювання столиків;

– дослідити наявні програмні рішення для автоматизації процесів бронювання в закладах харчування;

– розробити модель даних для представлення інформації про столики, клієнтів і бронювання;

– спроєктувати архітектуру інформаційної системи з розмежуванням клієнтської та серверної частин;

– реалізувати програмний прототип системи та провести тестування її функціональних можливостей.

Об’єктом дослідження є процес організації та управління бронюванням столиків у ресторані.

Предметом дослідження є методи та програмні засоби автоматизації бронювання, обліку та управління інформацією про використання ресторанних столиків.

Для досягнення поставлених завдань використано методи системного аналізу, моделювання інформаційних систем, проєктування баз даних, об’єктно-орієнтованого програмування та веброзробки. Реалізація програмної частини здійснюється з використанням сучасних вебтехнологій, що забезпечують взаємодію користувачів із системою через зручний інтерфейс та підтримують оброблення запитів у реальному часі.

Наукова новизна роботи полягає у розробленні узагальненої архітектури інформаційної системи бронювання столиків, що поєднує автоматизований облік бронювань, управління доступністю ресурсів ресторану та можливості подальшого аналітичного розширення.

Практична цінність отриманих результатів полягає у можливості використання розробленої системи в закладах ресторанного господарства для підвищення ефективності організації обслуговування клієнтів, зменшення кількості помилок під час бронювання, оптимізації завантаженості столиків та покращення загальної якості сервісу.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сфера ресторанного господарства характеризується високою динамічністю процесів обслуговування клієнтів, що зумовлює необхідність чіткої організації використання матеріальних і часових ресурсів закладу. Одним із ключових процесів у діяльності ресторану є бронювання столиків, яке забезпечує попереднє планування відвідувань, рівномірне завантаження залу та підвищення якості сервісу. Ефективність цього процесу безпосередньо впливає на задоволеність клієнтів, швидкість обслуговування та економічні показники закладу.

Проведений аналіз предметної області показав, що в умовах використання традиційних підходів до бронювання, зокрема телефонних замовлень або паперових журналів, виникають проблеми, пов’язані з дублюванням записів, відсутністю актуальної інформації про доступність столиків та складністю контролю змін бронювань. Це зумовлює необхідність впровадження інформаційної системи, яка забезпечує централізоване управління бронюваннями та оперативний доступ до даних для всіх учасників процесу.

У межах предметної області виокремлюються основні учасники процесу бронювання: клієнт, адміністратор ресторану та офіціант. Клієнт ініціює створення, зміну або скасування бронювання та отримує інформацію про його статус. Адміністратор здійснює управління столиками, підтвердження бронювань і контроль загального стану завантаженості ресторану. Офіціант використовує дані про бронювання для підготовки залу та організації обслуговування відвідувачів.

Функціональна взаємодія між зазначеними учасниками, процесами оброблення бронювань і сховищами даних відображена на DFD-діаграмі першого рівня, наведеній на рисунку 1.1. На діаграмі представлено основні процеси системи, зокрема керування бронюваннями, перевірку доступності та

підбір столиків, керування столиками, реєстрацію та управління користувачами, а також надсилання сповіщень клієнтам.

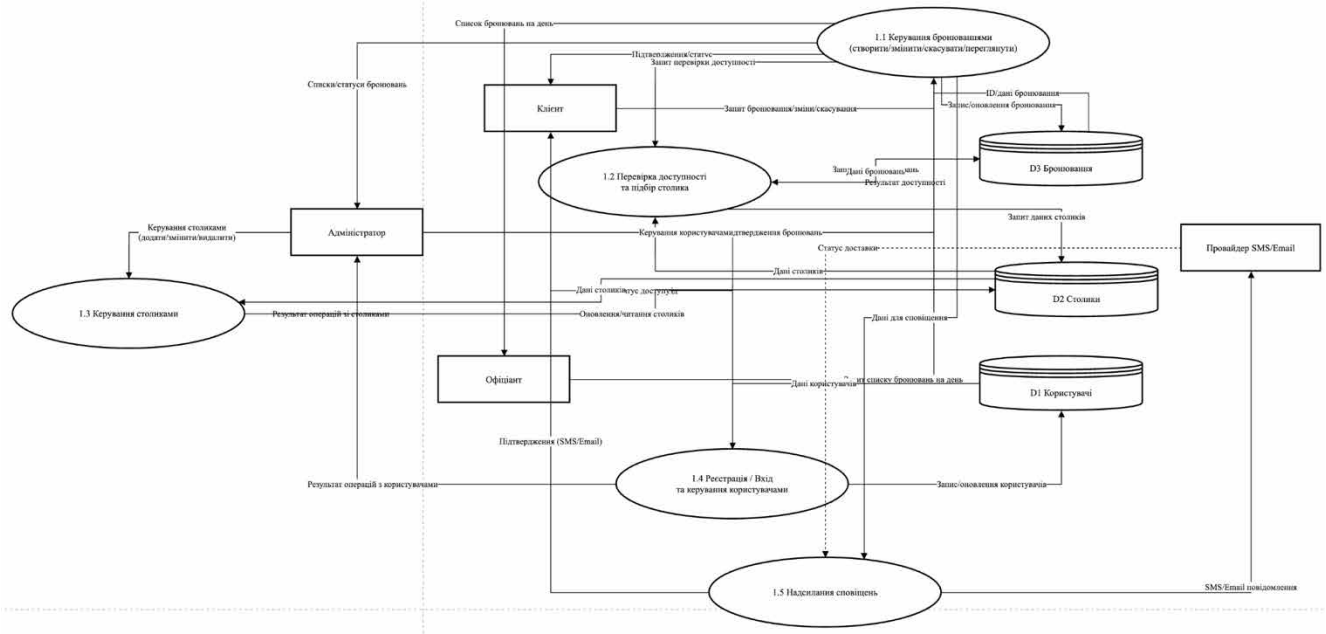

Рисунок 1.1 – DFD-діаграма першого рівня системи бронювання столиків у ресторані

Як показано на рисунку 1.1, центральним процесом є керування бронюваннями, яке взаємодіє зі сховищами даних бронювань, столиків і користувачів. Процес перевірки доступності забезпечує аналіз наявних столиків з урахуванням уже створених бронювань, що дозволяє уникнути конфліктів і дублювання. Окрему роль відіграє процес надсилання сповіщень, який забезпечує інформування клієнтів про підтвердження або зміну статусу бронювання за допомогою SMS або електронної пошти.

Для формалізації інформаційних об’єктів, що використовуються в межах предметної області, доцільно виділити основні сутності та їх характеристики. Перелік ключових сутностей предметної області системи бронювання столиків у ресторані наведено в таблиці 1.1.

Таблиця 1.1 – Основні сутності предметної області системи бронювання столиків

Сутність	Опис сутності
Користувач	Особа, яка взаємодіє із системою (клієнт, адміністратор, офіціант)

Продовження таблиці 1.1

Столик	Ресурс ресторану з визначеною кількістю місць та статусом доступності
Бронювання	Запис про резервування столика на визначену дату і час
Повідомлення	Інформаційне сповіщення про статус бронювання
Роль користувача	Набір прав доступу до функцій системи

Відповідно до таблиці 1.1, предметна область характеризується чітко визначеним набором сутностей, взаємодія між якими реалізується через інформаційну систему. Це забезпечує логічну структуризацію даних, спрощує процеси оброблення інформації та створює основу для подальшого проєктування архітектури системи.

Отже, предметна область системи бронювання столиків у ресторані охоплює процеси планування відвідувань, управління ресурсами закладу та інформаційної взаємодії між учасниками обслуговування. Формалізація цих процесів і їх реалізація у вигляді інформаційної системи створює передумови для підвищення ефективності роботи ресторану та покращення якості обслуговування клієнтів.

1.2 Аналіз існуючих рішень

Проведений аналіз ринку програмних продуктів для автоматизації процесів бронювання столиків у закладах ресторанного господарства показав, що на сьогодні існує низка комерційних інформаційних систем, які забезпечують попереднє резервування місць, управління посадкою гостей та підтримку взаємодії між клієнтами і персоналом ресторану. Найбільш поширеними серед них є OpenTable, Resy, Bookatable, Quandoo та Yelp Reservations. Зазначені рішення орієнтовані переважно на масовий ринок і використовуються у великій кількості ресторанів у різних країнах.

Одним із найвідоміших рішень є система OpenTable, яка надає клієнтам можливість здійснювати онлайн-бронювання столиків через веб-інтерфейс або

мобільний застосунок, а адміністраторам - керувати розсадкою гостей та переглядати статистику бронювань. Інтерфейс користувацької частини системи OpenTable наведено на рисунку 1.2.

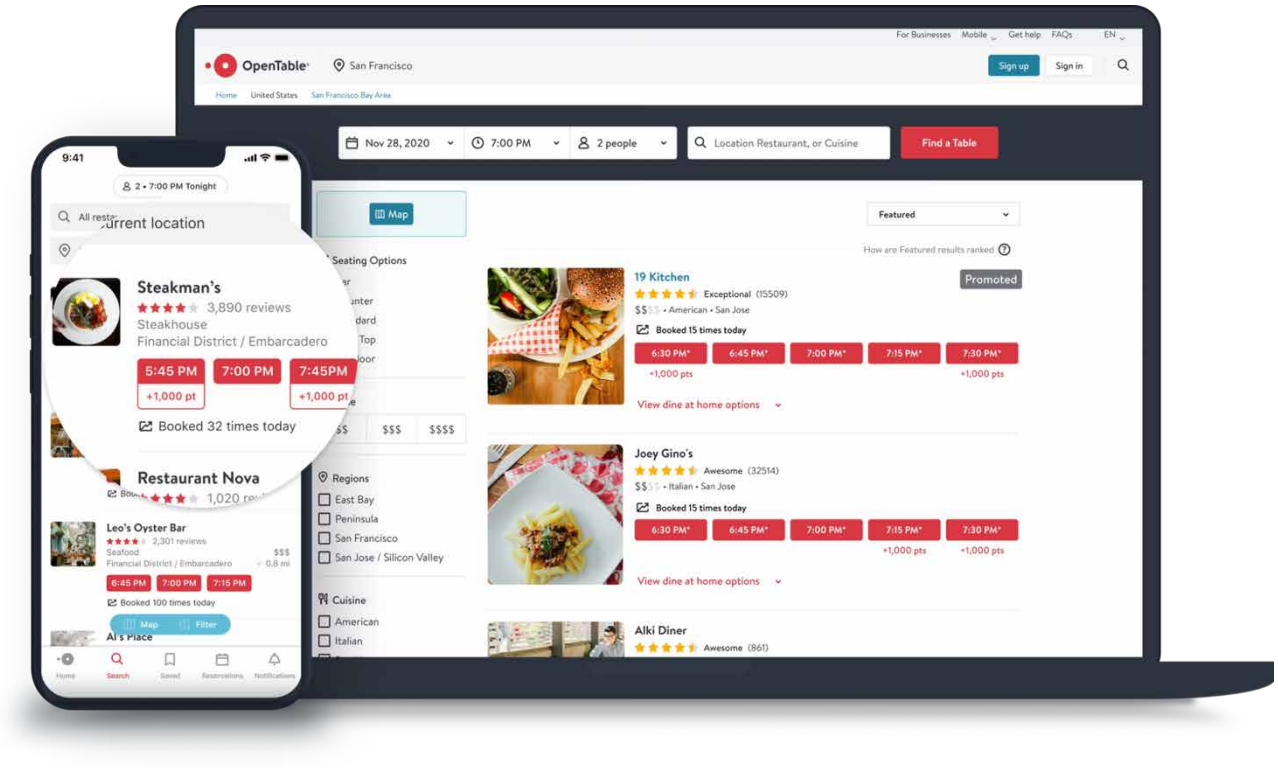

Рисунок 1.2 – Інтерфейс користувацької частини системи OpenTable

Як показано на рисунку 1.2, система забезпечує зручний пошук ресторанів за місцем розташування, часом та кількістю гостей, а також відображає доступні часові слоти для бронювання. Водночас значна частина функціональності орієнтована на кінцевого користувача, тоді як можливості гнучкого налаштування бізнес-логіки для окремого ресторану є обмеженими.

Іншим поширеним рішенням є система Resy, яка фокусується на внутрішніх процесах управління залом ресторану та посадкою гостей. Приклад інтерфейсу адміністративної частини системи Resy з візуальним відображенням столиків наведено на рисунку 1.3.

Рисунок 1.3 – Інтерфейс керування розсадкою гостей у системі Resy

Як видно з рисунка 1.3, Resy надає розширені можливості візуального контролю завантаженості залу, статусів столиків і часу обслуговування гостей. Проте використання такої системи потребує складного первинного налаштування та орієнтоване переважно на ресторани середнього й великого масштабу, що обмежує її застосування в невеликих закладах.

Система Quandoo є прикладом рішення, яке поєднує функції онлайн-бронювання для клієнтів та базові інструменти управління бронюваннями для ресторану. Інтерфейс клієнтської частини системи Quandoo наведено на рисунку 1.4.

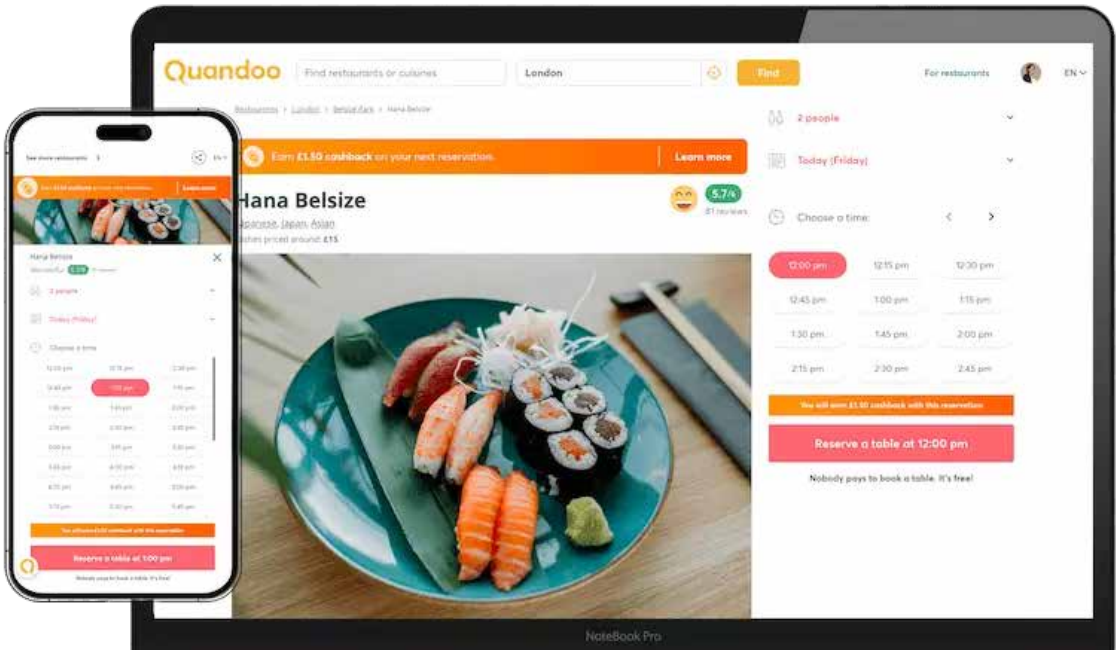

Рисунок 1.4 – Інтерфейс бронювання столиків у системі Quandoo

З рисунка 1.4 видно, що система орієнтована на швидке створення бронювань і має інтуїтивно зрозумілий інтерфейс. Однак функціональні можливості щодо аналізу даних, кастомізації ролей користувачів і розширення системи є обмеженими та залежать від умов комерційної підписки.

Окрему категорію рішень становлять системи, інтегровані з платіжними та CRM-модулями, прикладом яких є Yelp Reservations. Фрагмент інтерфейсу керування бронюваннями та історією відвідувань у подібних системах наведено на рисунку 1.5.

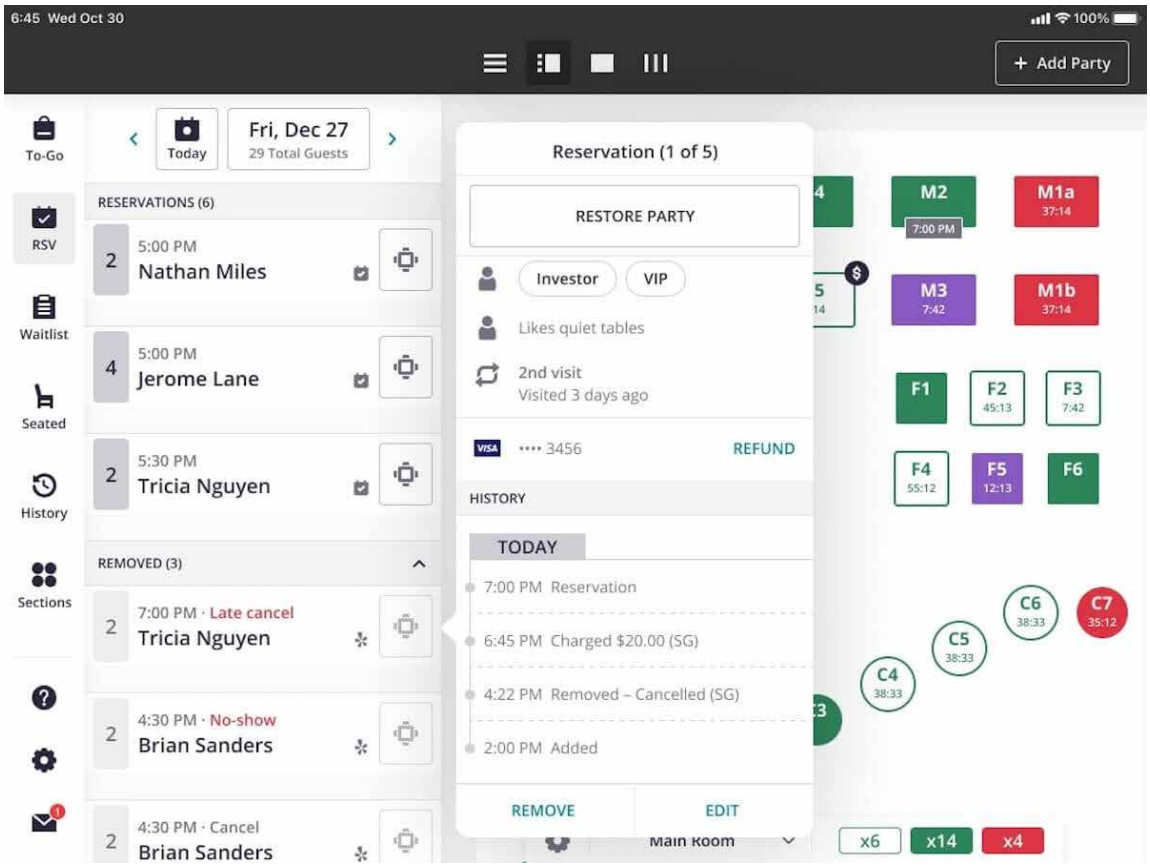

Рисунок 1.5 – Інтерфейс управління бронюваннями та історією відвідувань

Як показано на рисунку 1.5, такі системи забезпечують розширений облік клієнтів, історії відвідувань і фінансових операцій. Водночас їх впровадження супроводжується високою вартістю використання та залежністю від сторонніх сервісів, що може бути критичним для автономних або локальних рішень.

Для узагальнення результатів аналізу доцільно виконати порівняння основних функціональних характеристик існуючих рішень із розроблюваною інформаційною системою бронювання столиків. Порівняльну характеристику наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння існуючих систем бронювання з розроблюваною системою

Критерій порівняння	OpenTable	Resy	Quandoo	Yelp Reservations	Розроблювана система
Онлайн-бронювання	+	+	+	+	+
Управління столиками	+	+	±	+	+

Продовження таблиці 1.2

Гнучке керування ролями	−	±	−	±	+
Аналітика та звіти	+	+	−	+	+
Кастомізація під ресторан	−	±	−	−	+
Залежність від сторонніх сервісів	+	+	+	+	−
Можливість локального розгортання	−	−	−	−	+

Відповідно до таблиці 1.2, існуючі комерційні рішення забезпечують базову функціональність бронювання та управління посадкою гостей, однак мають обмеження щодо гнучкості налаштування, автономності та адаптації до потреб конкретного ресторану. Це підтверджує доцільність розроблення власної інформаційної системи бронювання столиків, орієнтованої на розширюваність, контроль над даними та можливість подальшого функціонального розвитку.

Отже, проведений аналіз існуючих рішень свідчить про наявність функціональних прогалин у сучасних системах бронювання столиків, що зумовлює необхідність створення розроблюваної інформаційної системи, здатної поєднати базові можливості бронювання з гнучким управлінням ресурсами ресторану та підтримкою аналітичних функцій.

1.3 Моделювання предметної області

Моделювання предметної області є необхідним етапом проєктування інформаційної системи, оскільки дозволяє формалізувати основні процеси, учасників та сценарії взаємодії в межах системи бронювання столиків у ресторані. Побудова моделей забезпечує цілісне уявлення про функціонування системи, спрощує аналіз вимог та створює основу для подальшої реалізації програмного забезпечення.

Для опису функціональних можливостей системи та взаємодії користувачів із нею використано діаграму прецедентів UML. На діаграмі відображено основних акторів системи - клієнта, адміністратора та офіціанта - а також перелік прецедентів, що реалізуються кожною роллю. Діаграма прецедентів предметної області системи бронювання столиків наведена на рисунку 1.6.

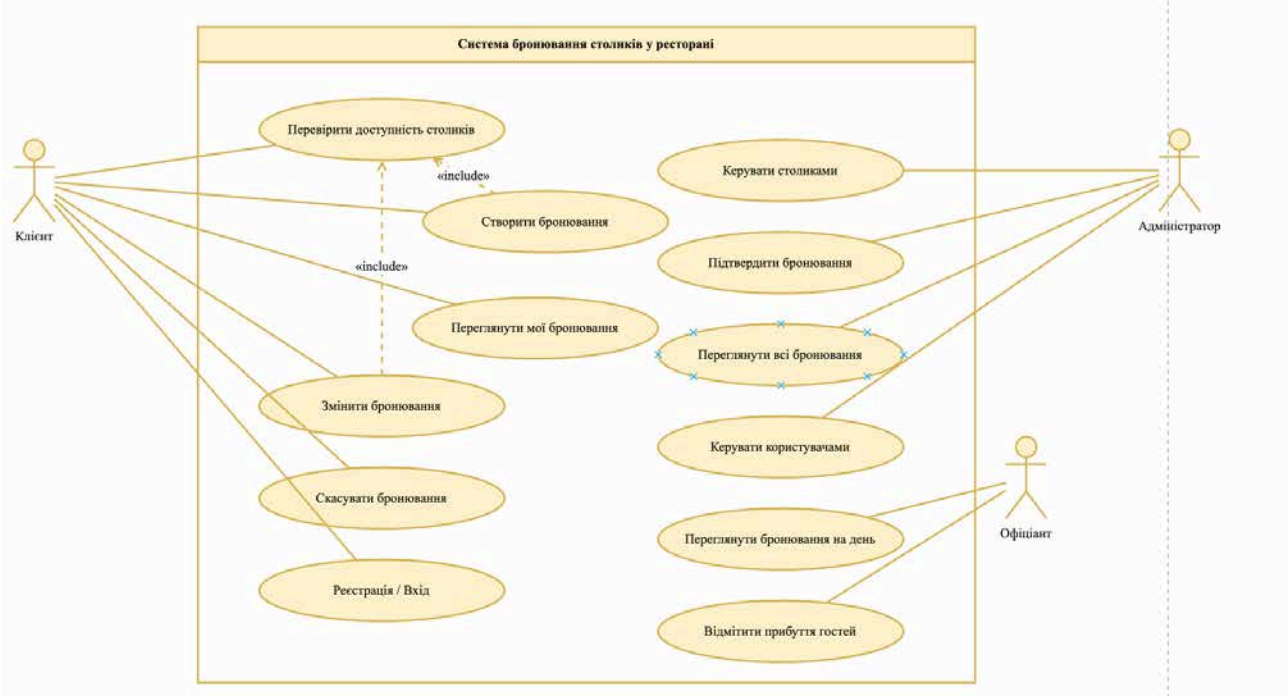

Рисунок 1.6 – Діаграма прецедентів системи бронювання столиків у ресторані

Як показано на рисунку 1.6, клієнт має можливість перевіряти доступність столиків, створювати, змінювати та скасовувати бронювання, а також переглядати власні бронювання після проходження процедури реєстрації або входу до системи. Адміністратор здійснює керування столиками, підтвердження бронювань, перегляд усіх активних бронювань і управління користувачами системи. Офіціант, у свою чергу, використовує систему для перегляду бронювань на день та відмітки факту прибуття гостей. Зв’язок типу «include» між прецедентами відображає залежності між перевіркою доступності столиків та операціями створення або зміни бронювання.

Для деталізації сценарію створення бронювання та аналізу динамічної взаємодії між компонентами системи побудовано діаграму послідовності. Вона відображає обмін повідомленнями між клієнтом, веб- або мобільним інтерфейсом, сервісом бронювання, базою даних та сервісом сповіщень. Діаграма послідовності процесу бронювання столика наведена на рисунку 1.7.

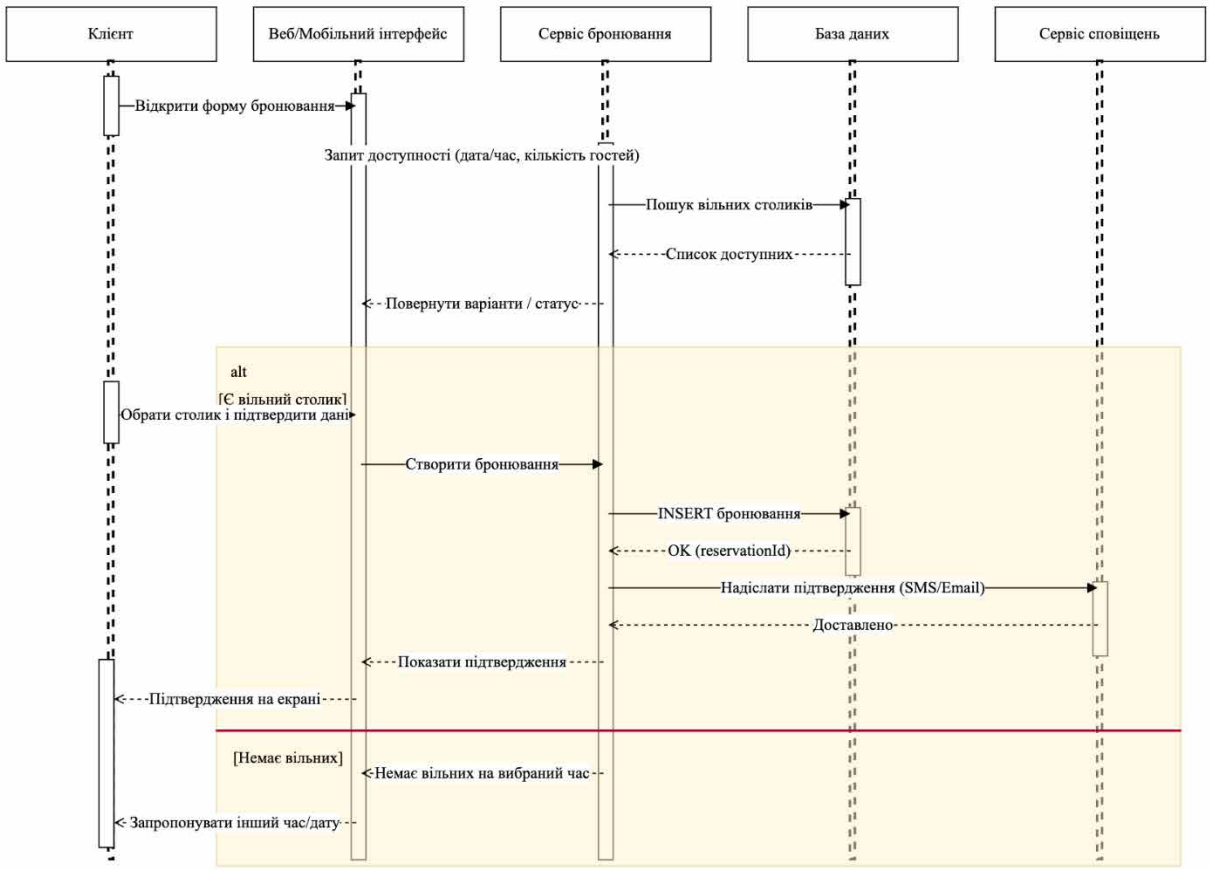

Рисунок 1.7 – Діаграма послідовності процесу створення бронювання

Відповідно до рисунка 1.7, процес починається з ініціації клієнтом запиту на бронювання, після чого система перевіряє доступність столиків на основі заданих параметрів. У разі наявності вільних столиків клієнт обирає відповідний варіант і підтверджує дані бронювання, що призводить до збереження інформації в базі даних та надсилання підтвердження за допомогою SMS або електронної пошти. У випадку відсутності вільних столиків система інформує клієнта та пропонує обрати інший час або дату, що реалізовано за допомогою альтернативного фрагмента (alt) діаграми.

Для моделювання логіки виконання дій і умовних переходів у межах процесу бронювання використано діаграму діяльності UML. Вона дозволяє

наочно представити послідовність дій клієнта та системи, а також варіанти розвитку процесу залежно від наявності або відсутності вільних столиків. Діаграма діяльності процесу бронювання столика наведена на рисунку 1.8.

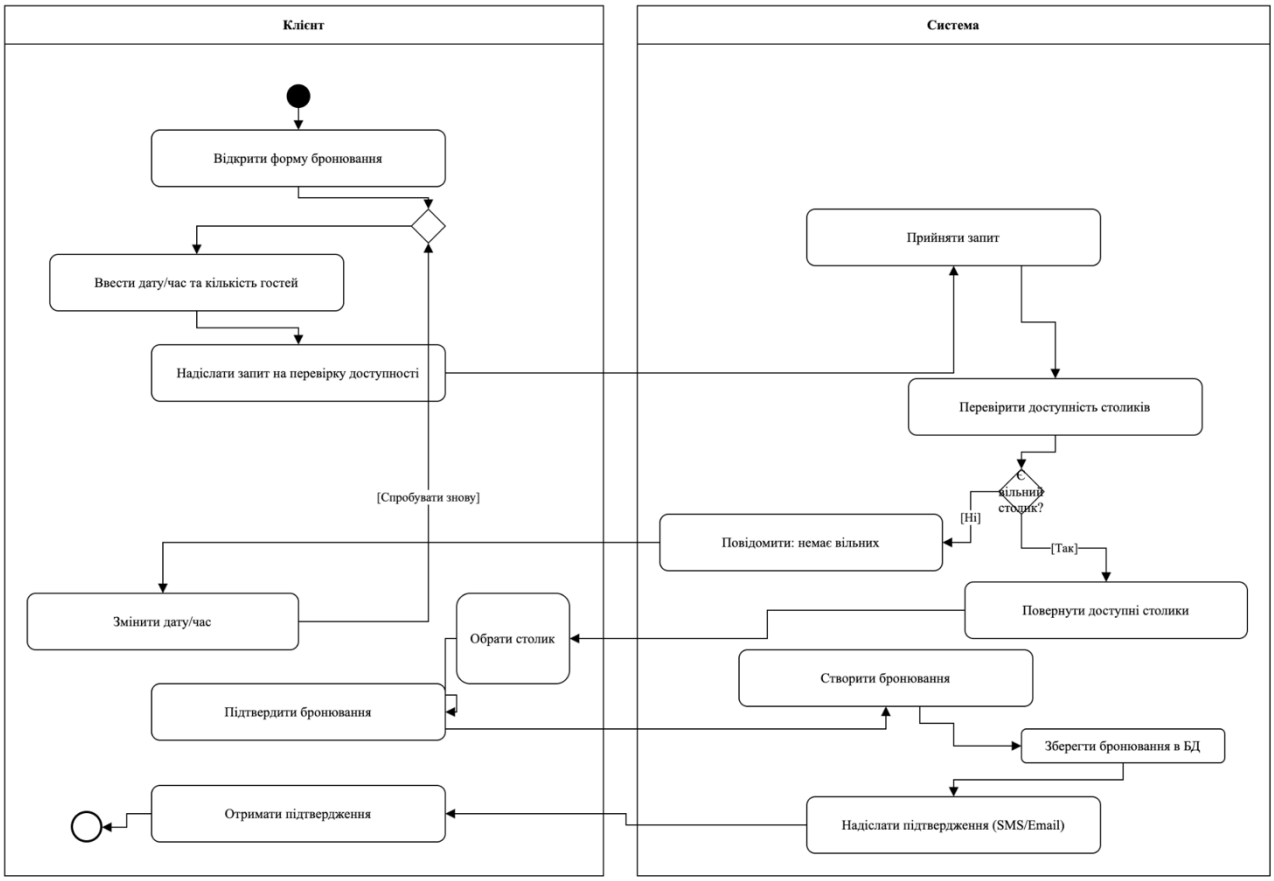

Рисунок 1.8 – Діаграма діяльності процесу бронювання столика

Як показано на рисунку 1.8, процес розпочинається з відкриття клієнтом форми бронювання та введення параметрів запиту. Система обробляє запит, перевіряє доступність столиків і, залежно від результату, або повертає доступні варіанти для підтвердження бронювання, або повідомляє про відсутність вільних столиків. Завершальним етапом процесу є отримання клієнтом підтвердження бронювання після його успішного збереження в базі даних.

Отже, побудовані UML-діаграми прецедентів, послідовності та діяльності дозволяють формалізувати предметну область системи бронювання столиків у ресторані, визначити ролі користувачів, основні сценарії взаємодії та логіку виконання процесів. Застосування зазначених моделей створює методологічно

обґрунтовану основу для подальшого проєктування архітектури системи та реалізації її програмної частини.

1.4 Аналіз вимог розроблюваної системи

Аналіз вимог до інформаційної системи бронювання столиків у ресторані є необхідним етапом проєктування, оскільки дозволяє визначити функціональні можливості системи, обмеження її реалізації та критерії якості. Формалізація вимог забезпечує узгодженість між очікуваннями користувачів і технічними рішеннями, а також створює основу для подальшого проєктування архітектури та програмної реалізації системи.

Вимоги до розроблюваної системи доцільно поділити на функціональні, нефункціональні, технічні та безпекові. Функціональні вимоги визначають перелік дій, які система повинна виконувати для підтримки процесів бронювання. Основні функціональні вимоги до системи бронювання столиків наведено в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до системи бронювання столиків

Код	Опис вимоги
F1	Забезпечення перевірки доступності столиків за датою, часом і кількістю гостей
F2	Створення бронювання клієнтом через веб-інтерфейс
F3	Можливість зміни та скасування бронювання
F4	Перегляд клієнтом власних бронювань
F5	Підтвердження бронювань адміністратором
F6	Управління столиками (додавання, редагування, видалення)
F7	Перегляд усіх бронювань адміністратором
F8	Перегляд бронювань на день офіціантом
F9	Надсилання повідомлень про статус бронювання
F10	Реєстрація та автентифікація користувачів

Відповідно до таблиці 1.3, функціональні вимоги охоплюють повний цикл роботи з бронюваннями та забезпечують взаємодію між клієнтами, персоналом ресторану та інформаційною системою.

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на зручність використання, продуктивність і надійність. Основні нефункціональні вимоги до розроблюваної системи наведено в таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до системи

Код	Опис вимоги
NF1	Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу
NF2	Підтримка одночасної роботи декількох користувачів
NF3	Час відповіді системи не більше 2 секунд
NF4	Надійна робота системи в режимі 24/7
NF5	Можливість масштабування системи
NF6	Адаптивність інтерфейсу для мобільних пристроїв

Згідно з таблицею 1.4, нефункціональні вимоги спрямовані на забезпечення стабільної роботи системи та позитивного користувацького досвіду.

Технічні вимоги визначають програмні та апаратні обмеження, у межах яких здійснюється реалізація системи. Основні технічні вимоги до розроблюваної системи наведено в таблиці 1.5.

Таблиця 1.5 – Технічні вимоги до системи

Код	Опис вимоги
T1	Реалізація серверної частини з використанням сучасних вебтехнологій
T2	Використання реляційної бази даних для зберігання інформації
T3	Підтримка REST API для взаємодії клієнтської та серверної частин
T4	Робота системи через веб-браузер без встановлення додаткового ПЗ
T5	Можливість розгортання на локальному або хмарному сервері

Відповідно до таблиці 1.5, технічні вимоги забезпечують гнучкість реалізації системи та її інтеграцію з іншими програмними компонентами.

Безпекові вимоги визначають заходи, необхідні для захисту даних користувачів і запобігання несанкціонованому доступу. Основні безпекові вимоги до розроблюваної системи наведено в таблиці 1.6.

Таблиця 1.6 – Безпекові вимоги до системи

Код	Опис вимоги
S1	Автентифікація та авторизація користувачів
S2	Розмежування прав доступу за ролями
S3	Захист персональних даних користувачів
S4	Захист від несанкціонованого доступу до бази даних
S5	Логування дій користувачів

Згідно з таблицею 1.6, безпекові вимоги спрямовані на забезпечення цілісності, конфіденційності та доступності інформації, що обробляється системою.

Отже, проведений аналіз вимог до розроблюваної системи бронювання столиків дозволив сформувати структурований перелік функціональних, нефункціональних, технічних і безпекових вимог. Дотримання зазначених вимог під час проєктування та реалізації системи забезпечить її коректну роботу, надійність, зручність використання та відповідність потребам користувачів і ресторанного бізнесу.

1.5 Постановка завдання

На основі проведеного аналізу предметної області, існуючих програмних рішень, а також сформованих вимог до розроблюваної інформаційної системи, доцільно сформулювати завдання дипломної роботи. Постановка завдання визначає напрям подальшого проєктування та реалізації системи бронювання столиків у ресторані й забезпечує узгодженість між теоретичною та практичною частинами дослідження.

Метою розроблюваної системи є автоматизація процесів бронювання столиків, управління доступністю ресурсів ресторану та інформаційної взаємодії між клієнтами і персоналом закладу. Для досягнення поставленої мети необхідно створити інформаційну систему, яка забезпечує централізоване зберігання даних про бронювання, столики та користувачів, а також підтримує виконання основних бізнес-процесів ресторанного обслуговування.

Виходячи з цього, у межах дипломної роботи необхідно розв’язати такі основні завдання. По-перше, слід розробити концептуальну та логічну модель інформаційної системи бронювання столиків із визначенням ролей користувачів, їхніх прав доступу та сценаріїв взаємодії з системою. По-друге, необхідно спроєктувати архітектуру системи з розмежуванням клієнтської та серверної частин, що забезпечить масштабованість і зручність подальшого розвитку. По-третє, потрібно розробити модель бази даних для зберігання інформації про бронювання, столики та користувачів із забезпеченням цілісності й узгодженості даних.

Крім того, важливим завданням є реалізація програмного прототипу системи, який підтримує функції перевірки доступності столиків, створення, зміни та скасування бронювань, управління столиками й користувачами, а також надсилання повідомлень про статус бронювання. Реалізація зазначених функцій повинна відповідати сформованим функціональним, нефункціональним, технічним і безпековим вимогам.

Окремим завданням є забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу, який дозволяє клієнтам швидко здійснювати бронювання, а персоналу ресторану - ефективно керувати процесами обслуговування. Також необхідно передбачити можливість тестування розробленої системи з метою перевірки коректності її роботи та відповідності поставленим вимогам.

Отже, постановка завдання дипломної роботи полягає у створенні інформаційної системи бронювання столиків у ресторані, яка забезпечує автоматизацію ключових бізнес-процесів, підвищення ефективності управління ресурсами закладу та покращення якості обслуговування клієнтів. Розв’язання поставлених завдань створює підґрунтя для подальшого проєктування, реалізації та впровадження системи, що розглядається в наступних розділах роботи.

1.6 Висновки до першого розділу

У першому розділі дипломної роботи було здійснено комплексний аналіз предметної області системи бронювання столиків у ресторані, що дозволило визначити основні особливості функціонування процесів ресторанного обслуговування та обґрунтувати доцільність їх автоматизації. Проведений аналіз показав, що використання традиційних підходів до організації бронювання не забезпечує необхідного рівня оперативності, прозорості та ефективності управління ресурсами закладу.

У межах дослідження було проаналізовано існуючі програмні рішення для бронювання столиків, визначено їх основні функціональні можливості та виявлено обмеження щодо гнучкості налаштування, автономності й адаптації до потреб конкретного ресторану. Результати порівняльного аналізу підтвердили доцільність розроблення власної інформаційної системи, орієнтованої на розширюваність і контроль над даними.

У процесі моделювання предметної області побудовано UML-діаграми прецедентів, послідовності та діяльності, які дозволили формалізувати ролі користувачів, основні сценарії взаємодії та логіку виконання процесів бронювання. Застосування зазначених моделей створило методологічно обґрунтовану основу для подальшого проєктування архітектури інформаційної системи.

Також було сформовано структурований перелік вимог до розроблюваної системи, зокрема функціональних, нефункціональних, технічних і безпекових, що визначають критерії якості та обмеження реалізації програмного продукту. На основі проведеного аналізу здійснено постановку завдання дипломної роботи, визначено мету, основні завдання та напрями подальших досліджень.

Отже, результати першого розділу створюють теоретичне та методологічне підґрунтя для проєктування архітектури та реалізації інформаційної системи бронювання столиків у ресторані, що розглядається в наступному розділі дипломної роботи.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних інформаційної системи бронювання столиків у ресторані розроблена для структурованого збереження інформації про клієнтів, столики, бронювання, статуси бронювання та працівників ресторану. Модель відображає основні об’єкти предметної області та зв’язки між ними, що забезпечує коректну роботу функцій створення, підтвердження, зміни та супроводу бронювань.

ER-діаграму логічної моделі даних наведено на рисунку 2.1.

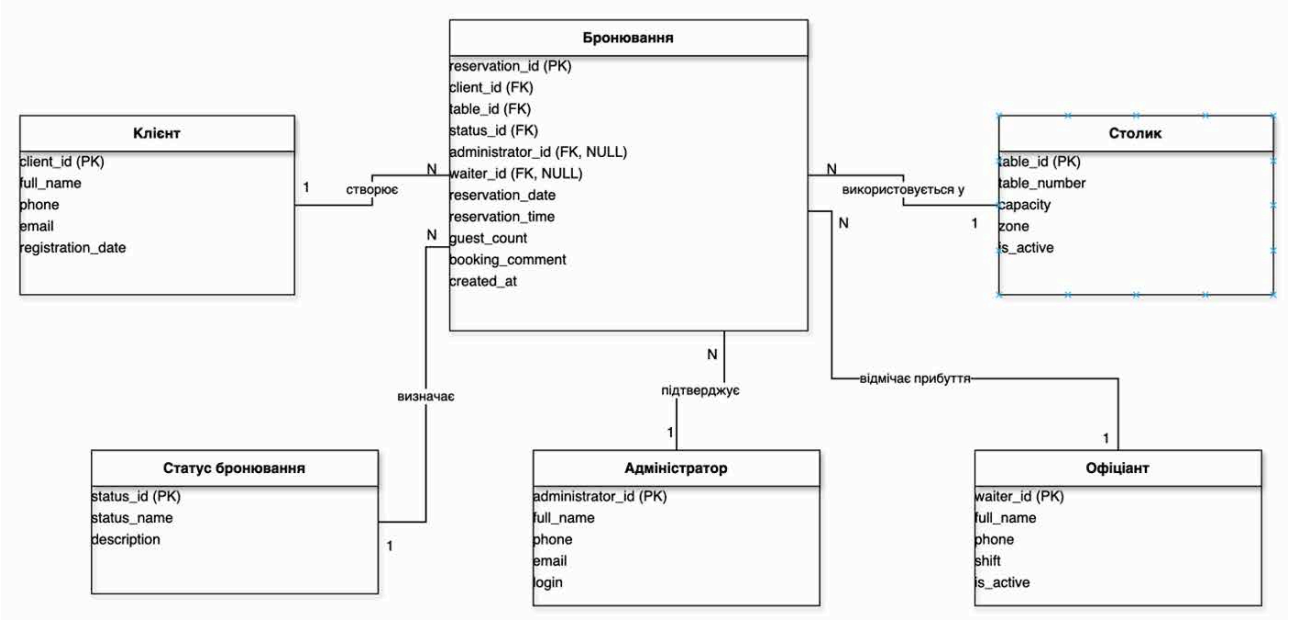

Рисунок 2.1 – Логічна модель даних інформаційної системи бронювання столиків у ресторані

Основою моделі є сутність «Бронювання», яка пов’язує клієнта, столик, поточний статус бронювання, адміністратора та офіціанта. Така структура дозволяє зберігати повну інформацію про кожне бронювання без дублювання основних даних про клієнтів, працівників і столики.

Таблиця 2.1 – Основні сутності логічної моделі даних

Сутність	Первинний ключ	Основні атрибути	Призначення
Клієнт	client_id	full_name, phone, email, registration_date	Зберігання контактних даних клієнтів, які створюють бронювання
Бронювання	reservation_id	client_id, table_id, status_id, administrator_id, waiter_id, reservation_date, reservation_time, guest_count, booking_comment, created_at	Фіксація факту резервування столика на визначену дату й час
Столик	table_id	table_number, capacity, zone, is_active	Опис ресторанних столиків, їх місткості, зони розміщення та активності
Статус бронювання	status_id	status_name, description	Зберігання можливих станів бронювання
Адміністратор	administrator_id	full_name, phone, email, login	Зберігання даних працівника, який підтверджує або контролює бронювання
Офіціант	waiter_id	full_name, phone, shift, is_active	Зберігання даних працівника, який обслуговує гостей і відмічає їх прибуття

У моделі використано зв’язки типу «один до багатьох». Один клієнт може створювати багато бронювань, один столик може використовуватися в багатьох бронюваннях у різні часові проміжки, а один статус може бути призначений багатьом записам бронювання. Також один адміністратор або офіціант може бути пов’язаний із кількома бронюваннями. Поля administrator_id та waiter_id у сутності «Бронювання» можуть бути порожніми, оскільки на початковому етапі бронювання ще може бути не підтверджене адміністратором або не закріплене за офіціантом.

Таблиця 2.2 – Зв’язки між сутностями логічної моделі

Сутність 1	Тип зв’язку	Сутність 2	Зміст зв’язку
Клієнт	1:N	Бронювання	Клієнт створює одне або декілька бронювань

Продовження таблиці 2.2

Столик	1:N	Бронювання	Столик використовується в різних бронюваннях у різний час
Статус бронювання	1:N	Бронювання	Статус визначає поточний стан бронювання
Адміністратор	1:N	Бронювання	Адміністратор підтверджує або контролює бронювання
Офіціант	1:N	Бронювання	Офіціант відмічає прибуття гостей і супроводжує обслуговування

Логічна модель даних побудована з урахуванням нормалізації до третьої нормальної форми. Усі сутності мають власні первинні ключі, а зв’язки між таблицями реалізуються через зовнішні ключі. Атрибути в межах таблиць є атомарними та описують лише відповідну сутність. Дані про клієнтів, столики, статуси та працівників не дублюються в таблиці бронювань, а зберігаються окремо. Це зменшує надлишковість інформації, спрощує оновлення записів і знижує ризик появи суперечливих даних.

Отже, розроблена логічна модель даних забезпечує впорядковане зберігання інформації про процес бронювання столиків у ресторані. Вона є основою для подальшого створення фізичної моделі бази даних, визначення SQL-структури таблиць і реалізації програмних модулів системи.

2.2 Діаграма класів і кооперації

Діаграма класів використовується для відображення статичної структури інформаційної системи бронювання столиків у ресторані. Вона визначає основні програмні класи, їх атрибути, операції та зв’язки між об’єктами, які забезпечують створення, зміну, підтвердження та супровід бронювань.

Діаграму класів інформаційної системи бронювання столиків у ресторані наведено на рисунку 2.2.

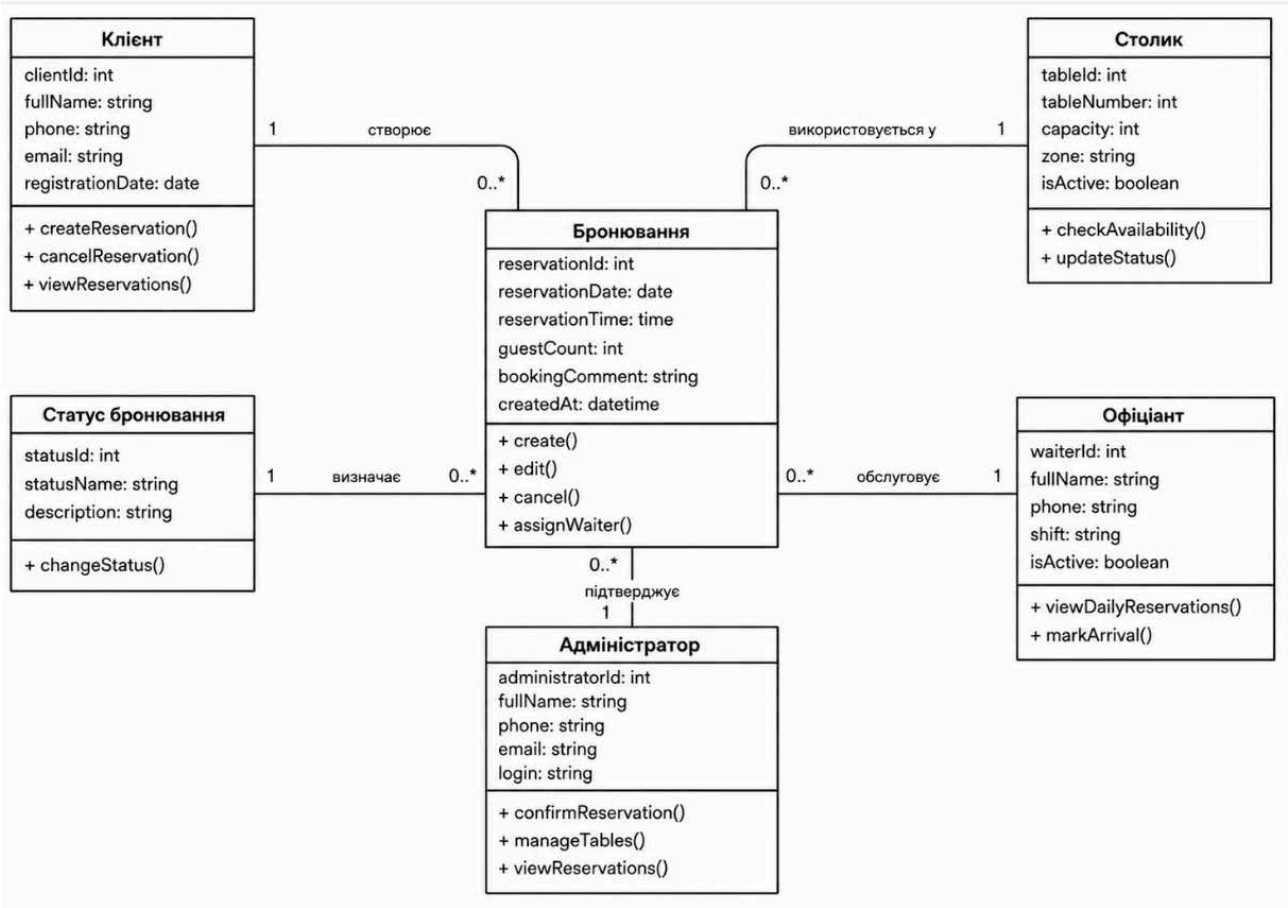

Рисунок 2.2 – Діаграма класів інформаційної системи бронювання столиків у ресторані

У розробленій діаграмі центральним класом є «Бронювання», оскільки саме він поєднує основні об’єкти системи: клієнта, столик, статус бронювання, адміністратора та офіціанта. Через цей клас реалізуються основні операції системи: створення бронювання, редагування, скасування та призначення офіціанта. Інші класи виконують допоміжні або сервісні функції, пов’язані з обліком користувачів, ресурсів ресторану та станів бронювання.

Таблиця 2.2 – Основні класи інформаційної системи бронювання столиків

Клас	Основні атрибути	Основні методи	Призначення
Клієнт	clientId, fullName, phone, email, registrationDate	createReservation(), cancelReservation(), viewReservations()	Створення, перегляд і скасування власних бронювань
Бронювання	reservationId, reservationDate, reservationTime,	create(), edit(), cancel(), assignWaiter()	Збереження даних про бронювання

	guestCount, bookingComment, createdAt		та керування його життєвим циклом

Продовження таблиці 2.2

Столик	tableId, tableNumber, capacity, zone, isActive	checkAvailability(), updateStatus()	Перевірка доступності столика та зміна його стану
Статус бронювання	statusId, statusName, description	changeStatus()	Визначення поточного стану бронювання
Адміністратор	administratorId, fullName, phone, email, login	confirmReservation(), manageTables(), viewReservations()	Підтвердження бронювань і керування столиками
Офіціант	waiterId, fullName, phone, shift, isActive	viewDailyReservations(), markArrival()	Перегляд бронювань на день і відмітка прибуття гостей

Між класами встановлено асоціативні зв’язки з кратністю. Один клієнт може створювати багато бронювань, але кожне бронювання належить одному клієнту. Один столик може використовуватися у багатьох бронюваннях у різні часові проміжки. Один статус може визначати стан багатьох бронювань. Адміністратор підтверджує бронювання, а офіціант обслуговує гостей після їх прибуття. Така структура відповідає логіці роботи ресторану та забезпечує поділ відповідальності між основними класами системи.

Для деталізації взаємодії об’єктів, визначених у діаграмі класів, побудовано три діаграми кооперації. Вони показують, як об’єкти системи обмінюються повідомленнями під час виконання основних сценаріїв роботи.

Перша кооперація описує процес створення бронювання клієнтом. У цьому сценарії беруть участь об’єкти класів «Клієнт», «Бронювання», «Столик» і «Статус бронювання». Клієнт ініціює створення бронювання, система перевіряє доступність столика, після чого формується запис бронювання зі статусом «Очікує підтвердження».

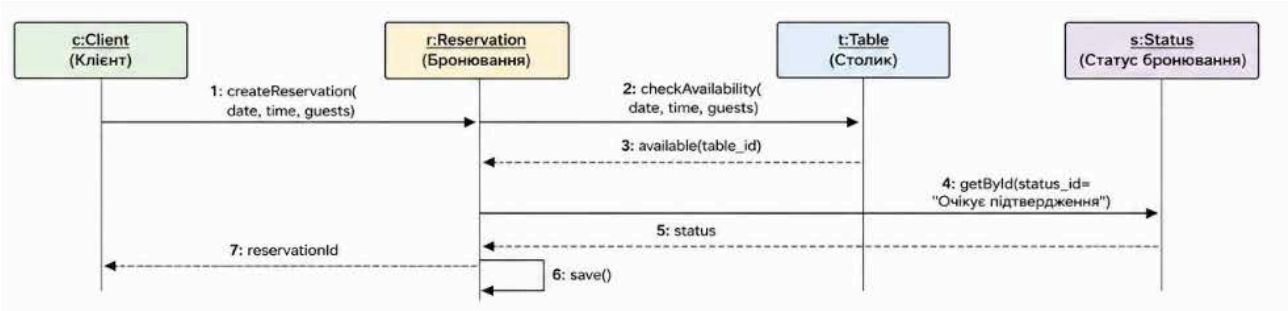

Рисунок 2.3 – Кооперація створення бронювання клієнтом

Друга кооперація відображає підтвердження бронювання адміністратором. Адміністратор переглядає список бронювань, обирає потрібний запис, підтверджує його, після чого об’єкт «Бронювання» змінює статус на «Підтверджено» та передає клієнту повідомлення про успішне підтвердження.

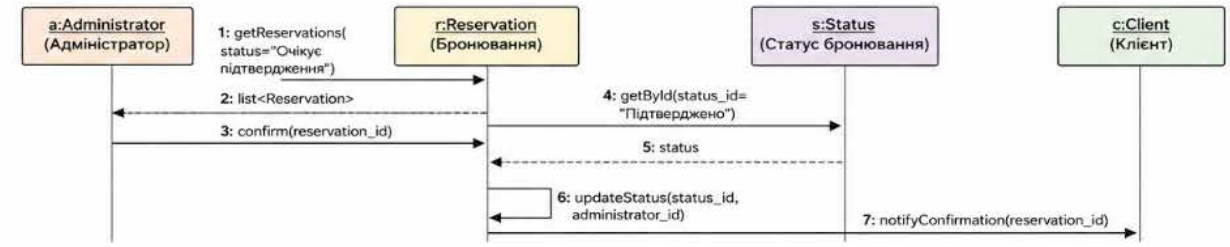

Рисунок 2.4 – Кооперація підтвердження бронювання адміністратором

Третя кооперація описує відмітку прибуття гостей офіціантом. Офіціант переглядає бронювання на поточний день, обирає відповідне бронювання та відмічає прибуття гостей. Після цього змінюється статус бронювання, а столик позначається як зайнятий.

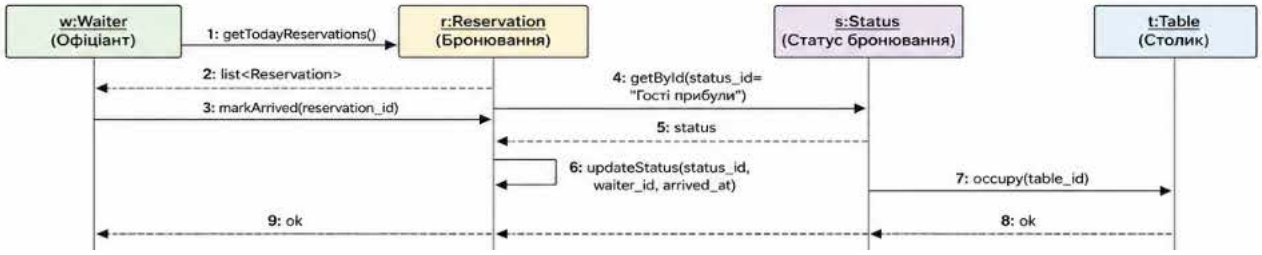

Рисунок 2.5 – Кооперація відмітки прибуття гостей офіціантом

Таблиця 2.3 – Характеристика кооперацій системи

Кооперація	Основні учасники	Результат виконання
Створення бронювання клієнтом	Клієнт, Бронювання, Столик, Статус бронювання	Створено новий запис бронювання зі статусом очікування

Продовження таблиці 2.3

Підтвердження бронювання адміністратором	Адміністратор, Бронювання, Статус бронювання, Клієнт	Бронювання підтверджено, клієнт отримує повідомлення
Відмітка прибуття гостей офіціантом	Офіціант, Бронювання, Статус бронювання, Столик	Зафіксовано прибуття гостей, столик позначено як зайнятий

Отже, діаграма класів визначає основну об’єктну структуру інформаційної системи бронювання столиків у ресторані, а діаграми кооперації деталізують взаємодію між цими об’єктами під час виконання ключових бізнес-процесів. Побудовані моделі є основою для подальшого проєктування програмних модулів системи, реалізації логіки бронювання та організації взаємодії між користувачами й системою.

2.3 Діаграма компонентів

Діаграма компонентів використовується для подання програмної структури інформаційної системи бронювання столиків у ресторані. Вона показує основні частини системи, їх призначення та взаємодію між клієнтською частиною, серверною логікою, рівнем доступу до даних, базою даних і зовнішніми сервісами.

Діаграму компонентів інформаційної системи бронювання столиків у ресторані наведено на рисунку 2.6.

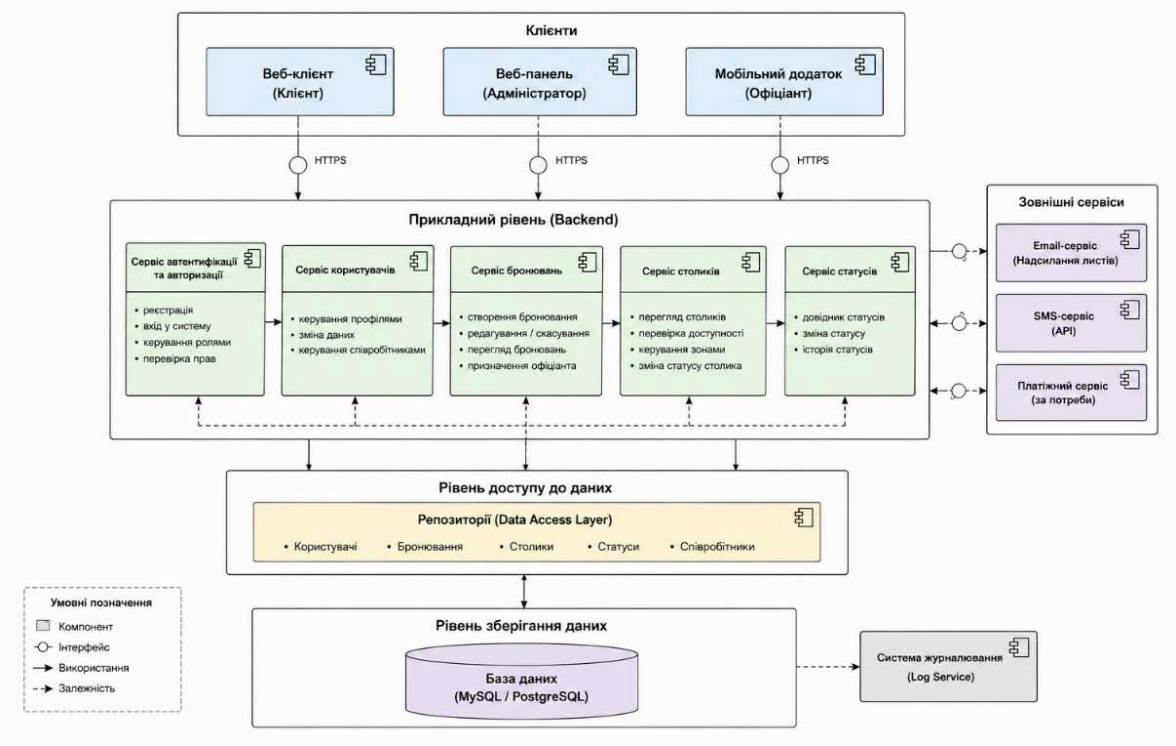

Рисунок 2.6 – Діаграма компонентів інформаційної системи бронювання столиків у ресторані

У запропонованій структурі система поділена на декілька логічних рівнів. На клієнтському рівні розміщено три основні інтерфейси: веб-клієнт для відвідувача ресторану, веб-панель адміністратора та мобільний додаток офіціанта. Усі клієнтські компоненти взаємодіють із серверною частиною через захищене HTTPS-з’єднання.

Прикладний рівень виконує основну бізнес-логіку системи. До нього входять сервіси автентифікації та авторизації, користувачів, бронювань, столиків і статусів. Ці компоненти відповідають за перевірку прав доступу, створення й редагування бронювань, керування столиками, зміну статусів і призначення працівників до процесу обслуговування.

Таблиця 2.4 – Основні компоненти системи бронювання столиків

Компонент	Рівень системи	Призначення
Веб-клієнт	Клієнтський рівень	Створення бронювання, перегляд доступних столиків, скасування або зміна бронювання
Веб-панель адміністратора	Клієнтський рівень	Перегляд бронювань, підтвердження заявок, керування столиками та співробітниками

Продовження таблиці 2.4

Мобільний додаток офіціанта	Клієнтський рівень	Перегляд бронювань на день, відмітка прибуття гостей
Сервіс автентифікації та авторизації	Прикладний рівень	Реєстрація, вхід у систему, перевірка ролей і прав доступу
Сервіс користувачів	Прикладний рівень	Керування профілями клієнтів і працівників ресторану
Сервіс бронювань	Прикладний рівень	Створення, редагування, скасування, перегляд і підтвердження бронювань
Сервіс столиків	Прикладний рівень	Перевірка доступності столиків, керування зонами та станом столиків
Сервіс статусів	Прикладний рівень	Зберігання довідника статусів і зміна стану бронювання
Репозиторій	Рівень доступу до даних	Виконання операцій читання, додавання, зміни та видалення даних
База даних	Рівень зберігання даних	Збереження інформації про користувачів, бронювання, столики, статуси та працівників
Email-сервіс	Зовнішні сервіси	Надсилання повідомлень про створення або підтвердження бронювання
SMS-сервіс	Зовнішні сервіси	Надсилання коротких повідомлень клієнтам
Система журналювання	Допоміжний компонент	Фіксація подій, помилок і дій користувачів

Рівень доступу до даних представлений репозиторієм, який ізолює прикладну логіку від безпосередньої роботи з базою даних. Через нього виконуються запити до таблиць користувачів, бронювань, столиків, статусів і співробітників. Такий підхід спрощує зміну структури збереження даних і дозволяє підтримувати єдиний механізм доступу до інформації.

На рівні зберігання даних використовується реляційна база даних MySQL або PostgreSQL. Вона забезпечує збереження основних об’єктів системи, контроль цілісності даних і підтримку зв’язків між клієнтами, бронюваннями, столиками, статусами, адміністраторами та офіціантами.

Окремо на діаграмі показано зовнішні сервіси. Email-сервіс і SMS-сервіс використовуються для надсилання клієнтам повідомлень про створення, підтвердження або зміну бронювання. Платіжний сервіс може бути підключений додатково, якщо ресторан передбачає передоплату або депозит за бронювання.

Система журналювання використовується для збереження технічних подій, помилок і важливих дій користувачів.

Таблиця 2.5 – Взаємодія компонентів системи

Джерело взаємодії	Компонент-приймач	Характер взаємодії
Веб-клієнт	Сервіс бронювань	Передавання запиту на створення або зміну бронювання
Веб-панель адміністратора	Сервіс бронювань	Підтвердження, перегляд і редагування бронювань
Мобільний додаток офіціанта	Сервіс бронювань	Отримання списку бронювань і відмітка прибуття гостей
Сервіс бронювань	Сервіс столиків	Перевірка доступності столика на задану дату й час
Сервіс бронювань	Сервіс статусів	Призначення або зміна статусу бронювання
Сервіси прикладного рівня	Репозиторій	Зчитування та оновлення даних
Репозиторій	База даних	Виконання SQL-запитів до таблиць системи
Сервіс бронювань	Email/SMS-сервіс	Надсилання повідомлення клієнту
Прикладний рівень	Система журналювання	Запис дій користувачів і системних подій

Запропонована компонентна структура є достатньо простою для реалізації, але водночас підтримує подальше розширення системи. За потреби до неї можна додати модуль аналітики завантаженості залу, модуль онлайн-оплати, інтеграцію з CRM або окремий сервіс звітності.

Отже, діаграма компонентів відображає програмну організацію інформаційної системи бронювання столиків у ресторані. Вона визначає основні клієнтські, серверні, сервісні та зовнішні компоненти, а також показує загальну логіку їх взаємодії. Побудована модель є основою для подальшого проєктування архітектури реалізації та розгортання програмного забезпечення.

2.4 Діаграма пакетів

Діаграма пакетів використовується для групування програмних модулів інформаційної системи бронювання столиків у ресторані за їх функціональним

призначенням. Така модель дозволяє показати загальну організацію програмного забезпечення, залежності між частинами системи та логіку поділу відповідальності між модулями.

Діаграму пакетів інформаційної системи бронювання столиків у ресторані наведено на рисунку 2.7.

Рисунок 2.7 – Діаграма пакетів інформаційної системи бронювання столиків у ресторані

У межах проєктування система поділена на окремі пакети, кожен із яких відповідає за певну частину функціональності. Основними пакетами є інтерфейс користувача, безпека та доступ, керування бронюваннями, керування столиками, керування користувачами, сповіщення, зберігання даних, звітність і зовнішні сервіси.

Таблиця 2.6 – Основні пакети інформаційної системи

Пакет	Склад пакета	Призначення
Інтерфейс користувача	веб-інтерфейс клієнта, панель адміністратора, інтерфейс офіціанта	Забезпечує взаємодію користувачів із системою
Безпека та доступ	автентифікація, авторизація, керування ролями	Перевіряє користувачів і визначає доступні функції відповідно до ролі
Керування бронюваннями	створення, редагування, скасування, підтвердження бронювань	Реалізує основну бізнес-логіку роботи з бронюваннями
Керування столиками	перелік столиків, зони залу, перевірка доступності, зміна стану	Забезпечує контроль ресторанних ресурсів
Керування користувачами	клієнти, адміністратори, офіціанти, профілі	Зберігає та обробляє дані користувачів і персоналу

Продовження таблиці 2.6

Сповіщення	email, SMS, системні повідомлення	Інформує клієнтів про статус бронювання
Зберігання даних	операційна база даних, довідники, журнали	Забезпечує збереження основної та службової інформації
Звітність	звіти по бронюваннях, завантаженості столиків, активності клієнтів	Формує аналітичні й управлінські звіти
Зовнішні сервіси	поштовий сервіс, SMS API, платіжний сервіс за потреби	Забезпечує інтеграцію із зовнішніми системами

Пакет «Інтерфейс користувача» залежить від пакета «Безпека та доступ», оскільки перед виконанням більшості операцій користувач має пройти автентифікацію. Після перевірки прав доступу запити передаються до функціональних пакетів, зокрема до модулів керування бронюваннями, столиками та користувачами.

Пакет «Керування бронюваннями» є центральним у структурі системи. Він використовує дані про клієнтів, столики, статуси бронювання та працівників ресторану. Через нього виконуються основні операції: створення бронювання клієнтом, підтвердження адміністратором, скасування бронювання та відмітка прибуття гостей офіціантом.

Пакет «Зберігання даних» використовується всіма функціональними модулями, оскільки забезпечує доступ до таблиць клієнтів, бронювань, столиків, статусів, адміністраторів і офіціантів. Винесення роботи з даними в окремий пакет спрощує підтримку системи та дозволяє змінювати структуру бази даних без суттєвого впливу на інші модулі.

Пакет «Сповіщення» взаємодіє із зовнішніми сервісами електронної пошти та SMS. Він використовується після створення, підтвердження, зміни або скасування бронювання. Пакет «Звітність» отримує дані з пакета зберігання та формує підсумкову інформацію для адміністратора ресторану.

Таблиця 2.7 – Основні залежності між пакетами

Пакет-джерело	Залежний пакет	Зміст залежності
Інтерфейс користувача	Безпека та доступ	Перевірка входу, ролі та прав користувача
Інтерфейс користувача	Керування бронюваннями	Передавання запитів на створення, зміну або перегляд бронювань
Керування бронюваннями	Керування столиками	Перевірка доступності столика перед створенням бронювання
Керування бронюваннями	Керування користувачами	Отримання даних клієнта, адміністратора або офіціанта
Керування бронюваннями	Сповіщення	Надсилання повідомлення про зміну статусу бронювання
Функціональні пакети	Зберігання даних	Зчитування та оновлення інформації в базі даних
Звітність	Зберігання даних	Отримання даних для формування звітів
Сповіщення	Зовнішні сервіси	Надсилання email або SMS-повідомлень

Запропонована пакетна структура забезпечує модульність програмного забезпечення та зменшує залежність між окремими частинами системи. Кожен пакет має чітко визначену зону відповідальності, що спрощує реалізацію, тестування та подальше розширення системи. За потреби до структури можна додати окремий пакет аналітики, модуль онлайн-оплати або інтеграцію з CRM-системою ресторану.

Отже, діаграма пакетів визначає логічну організацію програмного забезпечення інформаційної системи бронювання столиків у ресторані. Поділ системи на пакети забезпечує зрозумілу архітектуру, спрощує супровід програмного коду та створює основу для подальшої реалізації окремих модулів системи.

2.5 Висновки до другого розділу

У другому розділі було виконано проєктування інформаційного та програмного забезпечення інформаційної системи бронювання столиків у ресторані. Основну увагу приділено формуванню структури даних, визначенню

програмних класів, опису взаємодії об’єктів системи та побудові загальної архітектури програмних компонентів.

У пункті 2.1 розроблено логічну модель даних у вигляді ER-діаграми. У моделі визначено основні сутності системи: клієнт, бронювання, столик, статус бронювання, адміністратор та офіціант. Центральною сутністю є бронювання, оскільки воно поєднує інформацію про клієнта, обраний столик, дату й час візиту, кількість гостей, статус заявки та відповідальних працівників ресторану. Запропонована модель забезпечує структуроване зберігання даних і створює основу для подальшої побудови фізичної моделі бази даних.

У пункті 2.2 побудовано діаграму класів системи, яка відображає основні програмні об’єкти, їх атрибути, методи та зв’язки. Було визначено, що клас «Бронювання» є центральним елементом об’єктної структури, оскільки через нього реалізуються основні операції створення, редагування, скасування, підтвердження та супроводу бронювання. Також розроблено три кооперації, які деталізують ключові сценарії роботи системи: створення бронювання клієнтом, підтвердження бронювання адміністратором та відмітка прибуття гостей офіціантом.

У пункті 2.3 сформовано діаграму компонентів, яка відображає програмну організацію системи. Виділено клієнтський рівень, прикладний рівень, рівень доступу до даних, рівень зберігання даних та зовнішні сервіси. Такий поділ дозволяє чітко розмежувати відповідальність між інтерфейсами користувачів, серверною логікою, базою даних і допоміжними сервісами сповіщень та журналювання.

У пункті 2.4 побудовано діаграму пакетів, яка показує логічне групування модулів системи. До складу системи включено пакети інтерфейсу користувача, безпеки та доступу, керування бронюваннями, керування столиками, керування користувачами, сповіщень, зберігання даних, звітності та зовнішніх сервісів. Запропонована пакетна структура забезпечує модульність програмного забезпечення, спрощує його розроблення, тестування та подальше розширення.

Отже, у другому розділі було сформовано проєктну основу інформаційної системи бронювання столиків у ресторані. Розроблені ER-діаграма, діаграма класів, кооперації, діаграма компонентів і діаграма пакетів узгоджуються між собою та відображають логіку функціонування майбутнього програмного забезпечення. Отримані результати є підґрунтям для подальшого вибору технологій, реалізації програмних модулів, створення бази даних та тестування системи.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації

Розроблення CRM-системи бронювання столиків у ресторані потребує вибору таких технологічних засобів, які забезпечують швидку роботу інтерфейсу, надійне збереження даних, простоту супроводу та можливість локального використання в межах закладу. Оскільки система орієнтована на адміністратора ресторану, офіціанта та керування клієнтською базою, доцільним є створення настільного застосунку з графічним інтерфейсом, базою даних і окремими модулями для роботи з бронюваннями, столиками, клієнтами та звітами.

Для реалізації програмної системи обрано мову програмування Python та бібліотеку PyQt6. Такий вибір пояснюється тим, що Python має простий синтаксис, велику кількість бібліотек для роботи з даними, файлами, базами даних і графічними інтерфейсами. PyQt6 дає змогу створювати повноцінні настільні CRM-застосунки з формами, таблицями, кнопками, фільтрами, діалоговими вікнами та візуальними елементами керування.

Основний стек технологій, обраний для реалізації системи, наведено в таблиці 3.1.

Таблиця 3.1 – Технологічні засоби реалізації CRM-системи бронювання столиків

Засіб	Призначення в системі	Обґрунтування вибору
Python 3	Основна мова програмування	Забезпечує швидку розробку, зрозумілу структуру коду та підтримку великої кількості бібліотек
PyQt6	Розроблення графічного інтерфейсу	Дозволяє створити сучасний настільний інтерфейс із таблицями, формами, меню та діалоговими вікнами

Продовження таблиці 3.1

SQLite	Локальна база даних	Підходить для автономної CRM-системи ресторану, не потребує окремого сервера та проста в розгортанні
SQL	Створення й обробка структури БД	Забезпечує роботу з таблицями клієнтів, бронювань, столиків, статусів і працівників
Qt Designer	Проєктування форм інтерфейсу	Спрощує створення вікон, форм введення та розміщення елементів керування
PyCharm / VS Code	Середовище розроблення	Забезпечує зручне написання, запуск, тестування та налагодження програмного коду
PyInstaller	Формування інсталяційного пакета	Дозволяє зібрати застосунок у виконуваний файл для запуску без відкриття коду
QSS	Стилізація інтерфейсу	Дає змогу оформити елементи PyQt6 у єдиному мінімалістичному стилі

Архітектурно CRM-система передбачає поділ на декілька програмних частин: інтерфейс користувача, модулі бізнес-логіки, модуль роботи з базою даних і допоміжні сервіси. Інтерфейсна частина реалізується засобами PyQt6 та містить головне вікно, панель навігації, форми бронювання, таблиці перегляду даних, фільтри та діалогові вікна підтвердження дій. Бізнес-логіка відповідає за перевірку доступності столиків, створення бронювань, зміну статусів, призначення офіціанта та формування звітних даних.

Основні програмні модулі системи наведено в таблиці 3.2.

Таблиця 3.2 – Програмні модулі CRM-системи

Модуль	Призначення
Модуль авторизації	Перевірка користувача, вхід адміністратора або офіціанта до системи
Модуль клієнтів	Додавання, редагування, пошук і перегляд клієнтів ресторану
Модуль бронювань	Створення, зміна, скасування та підтвердження бронювань
Модуль столиків	Облік столиків, місткості, зон розміщення та стану доступності
Модуль статусів	Керування станами бронювання: очікує, підтверджено, скасовано, гості прибули
Модуль офіціанта	Перегляд бронювань на день і відмітка прибуття гостей
Модуль звітності	Формування статистики щодо бронювань, завантаженості столиків і активності клієнтів

Для збереження даних обрано SQLite, оскільки система розглядається як CRM-застосунок для локального використання в ресторані. Такий варіант є доцільним для невеликого або середнього закладу, де немає потреби в складній серверній інфраструктурі. SQLite забезпечує збереження інформації у локальному файлі бази даних, що спрощує встановлення програми та резервне копіювання. У разі подальшого масштабування систему можна перенести на PostgreSQL або MySQL без зміни загальної логіки роботи модулів.

Структура бази даних узгоджується з логічною моделлю, розробленою в попередньому розділі. Основними таблицями є клієнти, бронювання, столики, статуси бронювання, адміністратори та офіціанти. Така структура дозволяє зберігати повний цикл обробки бронювання: від створення заявки клієнтом до підтвердження адміністратором і відмітки прибуття гостей офіціантом.

Вибір PyQt6 обґрунтований необхідністю створення зручного інтерфейсу для персоналу ресторану. На відміну від консольних або простих веб-рішень, графічний CRM-застосунок дозволяє швидко працювати з таблицями бронювань, фільтрувати записи за датою, статусом або столиком, відкривати картку клієнта та змінювати стан бронювання без складних переходів між сторінками. Для адміністратора це забезпечує оперативний контроль завантаженості залу, а для офіціанта - швидкий доступ до списку гостей на поточний день.

Порівняння можливих засобів реалізації інтерфейсу наведено в таблиці 3.3.

Таблиця 3.3 – Порівняння засобів реалізації інтерфейсу

Засіб	Переваги	Недоліки	Доцільність використання
Tkinter	Вбудований у Python, простий для невеликих програм	Обмежені можливості дизайну та складніша стилізація	Доцільний для простих навчальних застосунків
PyQt6	Потужний інтерфейс, підтримка таблиць, форм, меню, стилів QSS	Потребує детальнішого проєктування структури вікон	Найбільш доцільний для CRM-системи
Kivy	Підтримка мобільного стилю інтерфейсу	Менш зручний для класичних настільних CRM-форм	Доцільний для мобільно орієнтованих застосунків

Продовження таблиці 3.3

Web-інтерфейс	Доступ через браузер, зручне масштабування	Потребує серверної частини та налаштування розгортання	Доцільний для мережевої версії системи

На основі порівняння обрано PyQt6, оскільки він найкраще відповідає вимогам настільної CRM-системи. Бібліотека підтримує роботу з табличними представленнями, діалоговими формами, вкладками, панелями керування, повідомленнями та стилізацією через QSS. Це дозволяє реалізувати інтерфейс, зручний для щоденного використання персоналом ресторану.

Для організації програмного коду доцільно використати модульну структуру проєкту. Окремо виділяються файли для головного вікна, форм бронювання, роботи з клієнтами, керування столиками, підключення до бази даних і службових функцій. Такий підхід полегшує підтримку програми, пошук помилок і подальше додавання нових можливостей.

Орієнтовну структуру програмного проєкту наведено в таблиці 3.4.

Таблиця 3.4 – Орієнтовна структура програмного проєкту

Елемент проєкту	Призначення
main.py	Запуск CRM-системи та ініціалізація головного вікна
database.py	Підключення до бази даних і виконання SQL-запитів
auth.py	Авторизація користувачів і перевірка ролей
clients.py	Робота з клієнтською базою
reservations.py	Логіка створення, редагування та скасування бронювань
tables.py	Облік столиків і перевірка їх доступності
reports.py	Формування звітів і статистичних показників
styles.qss	Оформлення інтерфейсу програми
restaurant_crm.db	Файл локальної бази даних SQLite

Для забезпечення зручності роботи користувача інтерфейс системи має бути мінімалістичним і структурованим. Основне вікно доцільно поділити на навігаційну панель і робочу область. У навігаційній панелі розміщуються розділи «Бронювання», «Клієнти», «Столики», «Офіціанти», «Звіти» та «Налаштування». У робочій області відображаються таблиці даних, фільтри, кнопки створення нових записів і форми редагування.

Обрані технології забезпечують виконання основних вимог до системи: швидкий запуск, локальне збереження даних, зручну роботу з бронюваннями, можливість фільтрації та пошуку, підтримку ролей користувачів і формування звітної інформації. Крім того, використання Python і PyQt6 дозволяє швидко оновлювати функціонал системи та адаптувати її до потреб конкретного ресторану.

Отже, для реалізації CRM-системи бронювання столиків у ресторані обрано стек Python, PyQt6, SQLite, SQL, QSS, Qt Designer і PyInstaller. Такий набір технологій є оптимальним для створення настільного застосунку, який поєднує зручний інтерфейс, надійне збереження даних і достатню функціональність для автоматизації основних процесів ресторанного бронювання. Обрані засоби створюють технічну основу для подальшого проєктування архітектури системи, реалізації програмних модулів і тестування готового програмного продукту.

3.2 Представлення архітектури системи

Архітектура розроблюваної CRM-системи бронювання столиків у ресторані побудована за модульним принципом. Такий підхід дозволяє розділити інтерфейс користувача, бізнес-логіку, роботу з базою даних, формування звітів і резервне копіювання даних. Система орієнтована на використання адміністратором ресторану або персоналом закладу через настільний застосунок, реалізований засобами Python та PyQt6.

Загальну архітектуру системи наведено на рисунку 3.1.

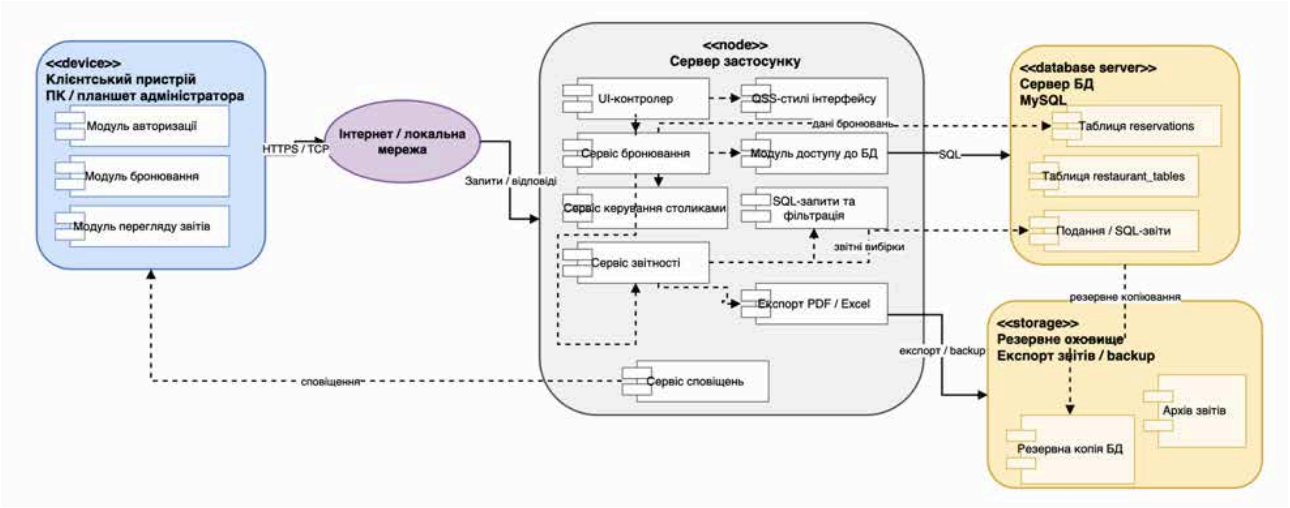

Рисунок 3.1 – Архітектура CRM-системи бронювання столиків у ресторані

У межах запропонованої архітектури користувач працює із системою через клієнтський пристрій: персональний комп'ютер або планшет адміністратора. На цьому рівні розміщуються основні модулі взаємодії з користувачем: модуль авторизації, модуль бронювання та модуль перегляду звітів. Передавання запитів між клієнтською частиною та сервером застосунку може здійснюватися через локальну мережу ресторану або через захищене з'єднання HTTPS/TCP.

Центральною частиною архітектури є сервер застосунку. У ньому зосереджено програмну логіку CRM-системи: UI-контролер, стилі інтерфейсу QSS, панель навігації, сервіс бронювання, сервіс керування столиками, сервіс звітності, модуль доступу до бази даних, модуль SQL-запитів і фільтрації, а також засоби експорту звітів у PDF або Excel. Такий поділ дозволяє не змішувати інтерфейсні елементи з логікою оброблення даних.

Таблиця 3.5 – Основні елементи архітектури системи

Елемент архітектури	Призначення
Клієнтський пристрій	Забезпечує доступ адміністратора або персоналу до CRM-системи
Модуль авторизації	Виконує вхід користувача до системи та перевірку прав доступу
Модуль бронювання	Забезпечує створення, редагування, скасування та перегляд бронювань
Модуль перегляду звітів	Відображає статистику бронювань і завантаженості столиків
UI-контролер	Керує переходами між вікнами та формами інтерфейсу
QSS-стилі інтерфейсу	Визначають зовнішній вигляд елементів програми

Продовження таблиці 3.5

Сервіс бронювання	Обробляє основні операції з бронюваннями
Сервіс керування столиками	Перевіряє доступність столиків і змінює їхній стан
Сервіс звітності	Формує підсумкові дані та звіти для адміністратора
Модуль доступу до БД	Забезпечує обмін даними між застосунком і базою даних
Сервер БД MySQL	Зберігає інформацію про бронювання, столики, клієнтів і статуси
Резервне сховище	Містить копії бази даних, скрипти експорту та архіви звітів

База даних розміщується на окремому сервері БД або локально на робочому комп'ютері залежно від умов використання системи. У роботі передбачено використання реляційної бази даних MySQL, у якій зберігаються таблиці бронювань, столиків, користувачів, статусів і працівників ресторану. Окремо можуть формуватися подання та SQL-звіти для швидкого отримання підсумкової інформації.

Основна логіка роботи системи полягає в тому, що користувач через інтерфейс формує запит, наприклад створює бронювання або відкриває звіт. Далі запит передається до відповідного сервісу на сервері застосунку. Сервіс перевіряє введені дані, звертається до модуля доступу до бази даних, виконує SQL-запит і повертає результат у графічний інтерфейс.

Таблиця 3.6 – Основні інформаційні потоки системи

Напрям обміну	Зміст інформації
Клієнтський пристрій → сервер застосунку	Дані авторизації, параметри бронювання, запити на перегляд звітів
Сервер застосунку → сервер БД	SQL-запити на додавання, зміну, пошук і фільтрацію даних
Сервер БД → сервер застосунку	Результати вибірки, інформація про столики, бронювання та статуси
Сервер застосунку → клієнтський пристрій	Відповіді системи, таблиці даних, повідомлення, сформовані звіти
Сервер застосунку → резервне сховище	Експорт звітів, резервні копії бази даних і службові файли
Сервер застосунку → сервіс сповіщень	Повідомлення про підтвердження, зміну або скасування бронювання

Окремим елементом архітектури є резервне сховище. Воно використовується для збереження резервних копій бази даних, архівів звітів і скриптів експорту. Це підвищує надійність системи та дозволяє відновити інформацію у разі пошкодження основної бази даних або помилки користувача.

Сервіс сповіщень використовується для інформування користувачів про важливі події в системі. Наприклад, після підтвердження бронювання адміністратором система може сформувати повідомлення для клієнта або службове сповіщення для персоналу ресторану.

Запропонована архітектура є достатньо простою для реалізації в межах настільної CRM-системи, але водночас підтримує подальше розширення. У майбутньому до неї можна додати веб-кабінет клієнта, онлайн-оплату, інтеграцію з SMS-сервісом або розширений модуль аналітики завантаженості ресторану.

Отже, архітектура системи бронювання столиків у ресторані передбачає поділ програмного забезпечення на клієнтську частину, сервер застосунку, базу даних і резервне сховище. Така структура забезпечує зрозумілу організацію модулів, стабільну роботу з даними, можливість формування звітів і зручне використання системи персоналом ресторану.

3.3 Фізична модель даних

Фізична модель даних визначає структуру таблиць бази даних CRM-системи бронювання столиків у ресторані, типи полів, ключі, обмеження цілісності та зв’язки між таблицями. На відміну від логічної моделі, фізична модель орієнтована на безпосередню реалізацію в середовищі реляційної СУБД MySQL.

Фізичну модель даних розроблюваної системи наведено на рисунку 3.2.

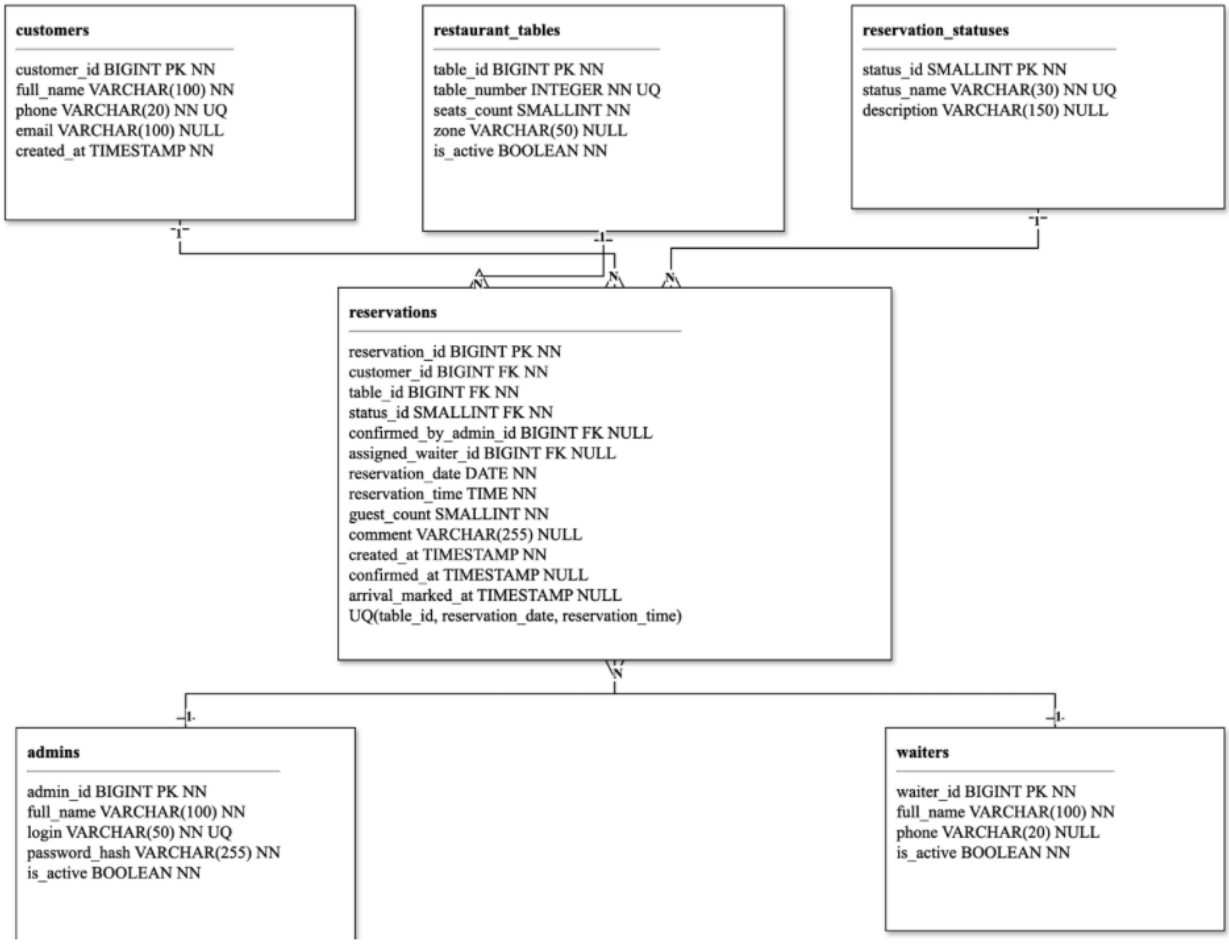

Рисунок 3.2 – Фізична модель даних CRM-системи бронювання столиків у ресторані

У фізичній моделі центральною таблицею є reservations, оскільки вона зберігає основну інформацію про бронювання та містить зовнішні ключі до таблиць customers, restaurant_tables, reservation_statuses, admins і waiters. Така структура дозволяє фіксувати клієнта, обраний столик, дату й час бронювання, кількість гостей, статус заявки, адміністратора, який підтвердив бронювання, та офіціанта, який обслуговує гостей.

Таблиця 3.7 – Основні таблиці фізичної моделі даних

Таблиця	Призначення	Ключові поля
customers	Зберігання даних клієнтів ресторану	customer_id, full_name, phone, email, created_at
restaurant_tables	Облік столиків ресторану	table_id, table_number, seats_count, zone, is_active
reservation_statuses	Довідник статусів бронювання	status_id, status_name, description

Продовження таблиці 3.7

admins	Дані адміністраторів системи	admin_id, full_name, login, password_hash, is_active
waiters	Дані офіціантів ресторану	waiter_id, full_name, phone, is_active
reservations	Основна таблиця бронювань	reservation_id, customer_id, table_id, status_id, reservation_date, reservation_time, guest_count

Зв’язки між таблицями реалізовано за допомогою зовнішніх ключів. Один клієнт може мати багато бронювань, один столик може використовуватися в різних бронюваннях у різні часові інтервали, а один статус може бути призначений багатьом бронюванням. Зв’язки з таблицями admins і waiters є необов’язковими, оскільки бронювання може бути створене до підтвердження адміністратором або до призначення офіціанта.

Таблиця 3.8 – Зв’язки між таблицями фізичної моделі

Основна таблиця	Зовнішній ключ	Пов’язана таблиця	Тип зв’язку
reservations	customer_id	customers	1:N
reservations	table_id	restaurant_tables	1:N
reservations	status_id	reservation_statuses	1:N
reservations	confirmed_by_admin_id	admins	1:N
reservations	assigned_waiter_id	waiters	1:N

Для уникнення дублювання бронювань у моделі передбачено унікальне обмеження UQ(table_id, reservation_date, reservation_time). Воно не дозволяє створити два бронювання на один і той самий столик в однакову дату та час. Також унікальними є номер телефону клієнта, номер столика, назва статусу та логін адміністратора.

Фрагмент SQL-структури фізичної моделі даних подано нижче.

```
CREATE TABLE customers (
    customer_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL UNIQUE,
    email VARCHAR(100),
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE restaurant_tables (
    table_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    table_number INTEGER NOT NULL UNIQUE,
    seats_count SMALLINT NOT NULL,
```

```
    zone VARCHAR(50),
    is_active BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE reservation_statuses (
    status_id SMALLINT PRIMARY KEY AUTO_INCREMENT,
    status_name VARCHAR(30) NOT NULL UNIQUE,
    description VARCHAR(150)
);

CREATE TABLE admins (
    admin_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100) NOT NULL,
    login VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    is_active BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE waiters (
    waiter_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    is_active BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE reservations (
    reservation_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    customer_id BIGINT NOT NULL,
    table_id BIGINT NOT NULL,
    status_id SMALLINT NOT NULL,
    confirmed_by_admin_id BIGINT NULL,
    assigned_waiter_id BIGINT NULL,
    reservation_date DATE NOT NULL,
    reservation_time TIME NOT NULL,
    guest_count SMALLINT NOT NULL,
    comment VARCHAR(255),
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    confirmed_at TIMESTAMP NULL,
    arrival_marked_at TIMESTAMP NULL,

    CONSTRAINT fk_res_customer
        FOREIGN KEY (customer_id) REFERENCES customers(customer_id),

    CONSTRAINT fk_res_table
        FOREIGN KEY (table_id) REFERENCES restaurant_tables(table_id),

    CONSTRAINT fk_res_status
        FOREIGN KEY (status_id) REFERENCES reservation_statuses(status_id),

    CONSTRAINT fk_res_admin
        FOREIGN KEY (confirmed_by_admin_id) REFERENCES admins(admin_id),

    CONSTRAINT fk_res_waiter
        FOREIGN KEY (assigned_waiter_id) REFERENCES waiters(waiter_id),

    CONSTRAINT uq_table_datetime
        UNIQUE (table_id, reservation_date, reservation_time)
);
```

Для пришвидшення пошуку бронювань за датою, клієнтом, столиком і статусом доцільно створити індекси. Вони особливо важливі для роботи

адміністративної панелі, де часто виконуються фільтрація записів за поточною датою, перегляд зайнятості столиків і пошук бронювань клієнта.

```
CREATE INDEX idx_reservations_date
ON reservations(reservation_date);

CREATE INDEX idx_reservations_customer
ON reservations(customer_id);

CREATE INDEX idx_reservations_table
ON reservations(table_id);

CREATE INDEX idx_reservations_status
ON reservations(status_id);
```

Початкове наповнення довідника статусів забезпечує коректну роботу життєвого циклу бронювання. До базових статусів належать: «Очікує підтвердження», «Підтверджено», «Скасовано», «Гості прибули» та «Завершено».

```
INSERT INTO reservation_statuses (status_name, description) VALUES
('Очікує підтвердження', 'Бронювання створено та очікує перевірки
адміністратором'),
('Підтверджено', 'Бронювання підтверджено адміністратором'),
('Скасовано', 'Бронювання скасовано'),
('Гості прибули', 'Офіціант відмітив прибуття гостей'),
('Завершено', 'Обслуговування за бронюванням завершено');
```

Отже, фізична модель даних CRM-системи бронювання столиків у ресторані забезпечує збереження основних даних про клієнтів, столики, бронювання, статуси та персонал. Використання первинних і зовнішніх ключів, унікальних обмежень та індексів підвищує цілісність даних, зменшує ризик дублювання бронювань і забезпечує швидкий доступ до інформації під час роботи системи.

3.4 Висновки до третього розділу

У третьому розділі було виконано проєктування архітектури програмної системи та розроблено основу алгоритмічного й інформаційного забезпечення CRM-системи бронювання столиків у ресторані. Основну увагу приділено вибору технологій реалізації, побудові архітектури застосунку та формуванню фізичної структури бази даних.

У пункті 3.1 обґрунтовано вибір технологічних засобів для реалізації системи. Як основну мову програмування обрано Python, оскільки вона

забезпечує швидку розробку, зручну роботу з базами даних і достатню кількість бібліотек для створення прикладного програмного забезпечення. Для розроблення графічного інтерфейсу обрано PyQt6, що дає змогу реалізувати настільну CRM-систему з формами бронювання, таблицями даних, фільтрами, діалоговими вікнами та зручним інтерфейсом для адміністратора й персоналу ресторану. Для збереження даних передбачено використання реляційної бази даних MySQL, а для оформлення інтерфейсу - QSS-стилі.

У пункті 3.2 представлено архітектуру програмної системи. Розроблена архітектура передбачає поділ системи на клієнтський пристрій, сервер застосунку, сервер бази даних і резервне сховище. На рівні застосунку виділено UI-контролер, модуль бронювання, сервіс керування столиками, сервіс звітності, модуль доступу до бази даних, SQL-запити та фільтрацію, а також експорт звітів у PDF або Excel. Така структура забезпечує зрозумілий поділ відповідальності між модулями та спрощує подальшу реалізацію системи.

У пункті 3.3 розроблено фізичну модель даних CRM-системи. Визначено основні таблиці бази даних: customers, restaurant_tables, reservation_statuses, admins, waiters і reservations. Центральною таблицею є reservations, яка зберігає інформацію про бронювання та пов’язує клієнта, столик, статус, адміністратора й офіціанта. Для забезпечення цілісності даних передбачено первинні та зовнішні ключі, унікальні обмеження й індекси для швидкого пошуку бронювань за датою, клієнтом, столиком і статусом.

Запропонована структура бази даних дозволяє уникнути дублювання записів, контролювати зайнятість столиків і забезпечувати повний цикл роботи з бронюванням: створення заявки, підтвердження адміністратором, призначення офіціанта, відмітку прибуття гостей і завершення обслуговування. Використання унікального обмеження для комбінації table_id, reservation_date і reservation_time зменшує ризик подвійного бронювання одного столика на однаковий час.

Отже, у третьому розділі сформовано технічну основу реалізації CRM-системи бронювання столиків у ресторані. Обрані технології, розроблена архітектура та фізична модель даних забезпечують можливість створення

функціонального настільного застосунку для автоматизації бронювання, керування столиками, обліку клієнтів, роботи персоналу та формування звітності. Отримані результати є базою для подальшої програмної реалізації, тестування та оцінювання ефективності системи.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Тестування програмних модулів CRM-системи бронювання столиків у ресторані

Тестування програмних модулів CRM-системи бронювання столиків у ресторані виконувалося з метою перевірки коректності роботи основних функцій застосунку, стабільності взаємодії з базою даних, правильності оброблення введених даних і відповідності реалізованої системи визначеним функціональним вимогам.

Розроблена система реалізована як настільний застосунок на Python із використанням бібліотеки PyQt6 для графічного інтерфейсу та SQLite для локального збереження даних. Тому основну увагу під час тестування приділено перевірці модулів авторизації, реєстрації, бронювання, керування клієнтами, столиками, персоналом, статусами бронювання, звітністю та резервним копіюванням бази даних.

План тестування програмних модулів наведено в таблиці 4.1.

Таблиця 4.1 – План тестування програмних модулів CRM-системи бронювання столиків

№	Модуль	Мета тестування	Очікуваний результат
1	Модуль авторизації	Перевірка входу користувача за логіном і паролем	Користувач із правильними даними входить у систему, неправильні дані відхиляються
2	Модуль реєстрації	Перевірка створення нового облікового запису	Новий користувач додається до бази даних із відповідною роллю
3	Модуль клієнтів	Перевірка додавання, редагування та видалення клієнтів	Дані клієнтів коректно зберігаються, змінюються та відображаються
4	Модуль столиків	Перевірка обліку столиків, зон і кількості місць	Столики додаються, редагуються та враховуються під час бронювання
5	Модуль бронювання	Перевірка створення, зміни, скасування та підтвердження бронювань	Бронювання проходить повний життєвий цикл без помилок

Продовження таблиці 4.1

6	Модуль перевірки доступності	Перевірка зайнятості столика на дату й час	Система не дозволяє створити дубльоване бронювання
7	Модуль офіціанта	Перевірка відмітки прибуття гостей	Статус бронювання змінюється на «Гості прибули»
8	Модуль звітності	Перевірка формування статистики та експорту	Дані звітів відповідають записам у базі даних
9	Модуль резервного копіювання	Перевірка створення копії бази даних	Резервна копія створюється у вибраному місці
10	Інтерфейс користувача	Перевірка зручності роботи з формами, таблицями та кнопками	Елементи інтерфейсу відображаються коректно й не перекриваються

Для тестування модуля авторизації перевірялися сценарії входу адміністратора та офіціанта, а також введення неправильного логіна або пароля. У разі правильного введення облікових даних користувач переходить до головного вікна CRM-системи. Якщо дані введено неправильно, система виводить повідомлення про помилку входу. Додатково перевірено, що роль користувача впливає на доступні функції: адміністратор має ширший доступ до керування системою, а офіціант працює переважно з бронюваннями на день і відміткою прибуття гостей.

Модуль реєстрації тестувався через створення нового користувача з роллю адміністратора або офіціанта. Перевірялися обов’язкові поля, унікальність логіна, мінімальна довжина пароля та збереження пароля у вигляді хешу. У разі спроби повторної реєстрації з уже наявним логіном система виводить повідомлення про помилку.

Основні тестові сценарії наведено в таблиці 4.2.

Таблиця 4.2 – Тестові сценарії перевірки основних модулів системи

№	Сценарій тестування	Вхідні дані	Очікуваний результат	Результат
1	Вхід адміністратора	admin / admin123	Відкриття головного вікна адміністратора	Виконано
2	Вхід із неправильним паролем	admin / 12345	Повідомлення про помилку входу	Виконано

Продовження таблиці 4.2

3	Реєстрація нового офіціанта	ПІБ, логін, пароль, роль	Створення нового облікового запису	Виконано
4	Додавання нового клієнта	ПІБ, телефон, email	Клієнт з’являється у таблиці клієнтів	Виконано
5	Додавання нового столика	Номер, кількість місць, зона	Столик додається до довідника столиків	Виконано
6	Створення бронювання	Клієнт, столик, дата, час, кількість гостей	Створюється запис зі статусом очікування	Виконано
7	Повторне бронювання того самого столика на той самий час	Той самий столик, дата і час	Система блокує створення дубля	Виконано
8	Підтвердження бронювання	Обране бронювання	Статус змінюється на «Підтверджено»	Виконано
9	Відмітка прибуття гостей	Обране бронювання	Статус змінюється на «Гості прибули»	Виконано
10	Скасування бронювання	Обране бронювання	Статус змінюється на «Скасовано»	Виконано
11	Експорт бронювань	Команда експорту CSV	Формується файл із даними бронювань	Виконано
12	Резервне копіювання БД	Команда створення backup	Створюється копія файлу бази даних	Виконано

Під час тестування модуля бронювання перевірялися операції створення, редагування, підтвердження, скасування та завершення бронювання. Особливу увагу приділено перевірці доступності столика, оскільки ця функція є однією з ключових для ресторанної CRM-системи. Якщо користувач намагається створити бронювання на столик, який уже зайнятий у вибрані дату й час, система не дозволяє зберегти запис і повідомляє про конфлікт.

Модуль клієнтів перевірявся за сценаріями додавання нового клієнта, редагування контактних даних, пошуку клієнта та використання його запису під час створення бронювання. Для унеможливлення дублювання в базі даних перевірялася унікальність номера телефону клієнта.

Модуль столиків тестувався через додавання нових столиків, зміну кількості місць, зони розміщення та активності столика. Перевірено, що неактивні столики не використовуються під час створення нового бронювання, а активні столики доступні для вибору в формі бронювання.

Для модуля звітності перевірялося відображення статистики за статусами бронювання, кількістю клієнтів, кількістю активних столиків і бронюваннями на поточну дату. Окремо перевірено експорт даних у CSV-файл, який може бути відкритий у табличному редакторі для подальшого аналізу.

Критерії оцінювання результатів тестування наведено в таблиці 4.3.

Таблиця 4.3 – Критерії оцінювання результатів тестування

Критерій	Спосіб перевірки	Очікуване значення
Коректність авторизації	Вхід із правильними та неправильними даними	Доступ надається лише зареєстрованим користувачам
Цілісність даних	Створення та зміна записів у БД	Дані зберігаються без втрат і дублювання
Перевірка доступності столика	Повторне бронювання на однаковий час	Дубльоване бронювання блокується
Швидкість відкриття основних форм	Перехід між розділами системи	Форми відкриваються без помітної затримки
Зручність інтерфейсу	Робота з таблицями, кнопками, фільтрами	Елементи зрозумілі та доступні користувачу
Стабільність роботи	Багаторазове виконання типових операцій	Програма не завершується аварійно
Експорт і резервне копіювання	Створення CSV-файлу та backup БД	Файли створюються коректно

За результатами тестування встановлено, що основні програмні модулі CRM-системи працюють коректно та забезпечують виконання ключових функцій: авторизацію, реєстрацію, керування клієнтами, столиками, бронюваннями, статусами, звітністю та резервним копіюванням. Реалізовані перевірки введених даних і обмеження бази даних дозволяють зменшити кількість помилок користувача та запобігти створенню некоректних записів.

Отже, тестування програмних модулів підтвердило працездатність розробленої CRM-системи бронювання столиків у ресторані. Система забезпечує повний цикл роботи з бронюваннями, підтримує розмежування ролей користувачів і має достатній рівень стабільності для використання в умовах невеликого або середнього ресторану.

4.2 Тестування системи

Тестування CRM-системи бронювання столиків у ресторані виконувалося після реалізації основних програмних модулів. Метою тестування було перевірити працездатність застосунку, правильність оброблення даних, зручність інтерфейсу, коректність авторизації, реєстрації користувачів, створення бронювань, перевірки доступності столиків, ведення клієнтської бази та формування звітів.

Перевірка системи проводилася у ручному режимі шляхом виконання типових дій користувача. Основними ролями під час тестування були адміністратор ресторану та офіціант. Адміністратор має доступ до керування бронюваннями, клієнтами, столиками, персоналом і звітами. Офіціант працює переважно з бронюваннями на поточний день і відміткою прибуття гостей.

Першим етапом було перевірено роботу вікна входу до системи. У формі авторизації користувач вводить логін і пароль, після чого система перевіряє наявність облікового запису в базі даних. У разі правильного введення даних відкривається головне вікно CRM-системи. Вікно входу до системи наведено на рисунку 4.1.

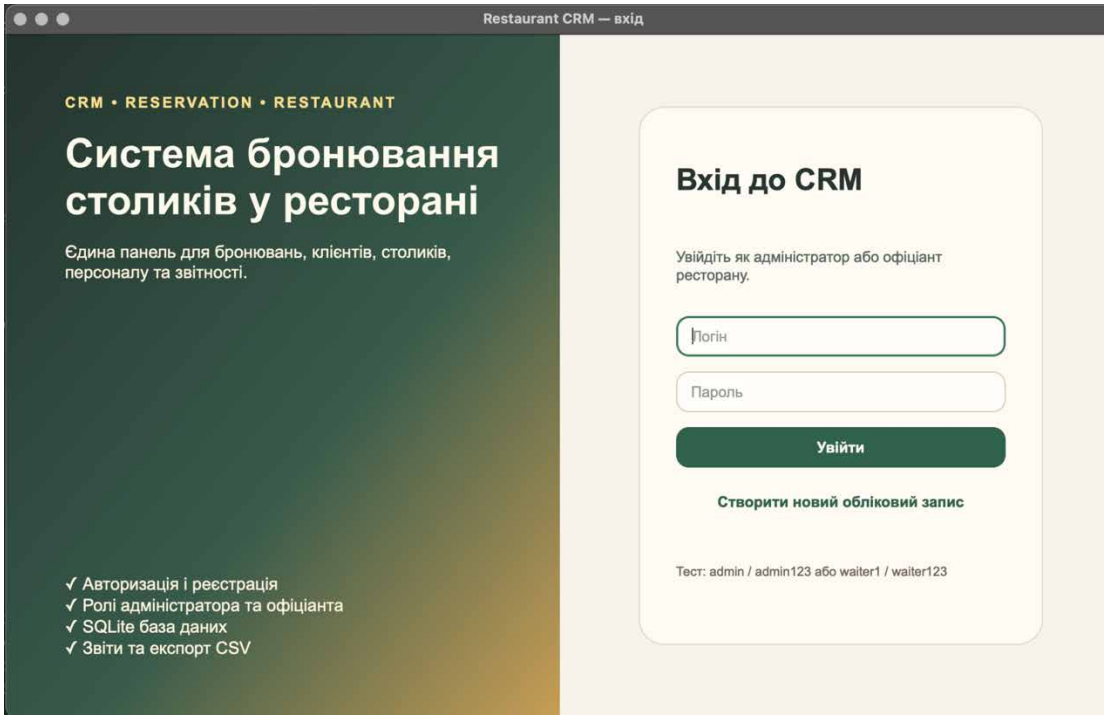

Рисунок 4.1 – Вікно авторизації користувача в CRM-системі

Окремо було протестовано модуль реєстрації. У цьому модулі користувач вводить ПІБ, телефон, email, логін, пароль і роль. Система перевіряє заповнення обов’язкових полів, унікальність логіна та коректність створення нового облікового запису. Після успішної реєстрації користувач може виконати вхід до системи. Вікно реєстрації користувача наведено на рисунку 4.2.

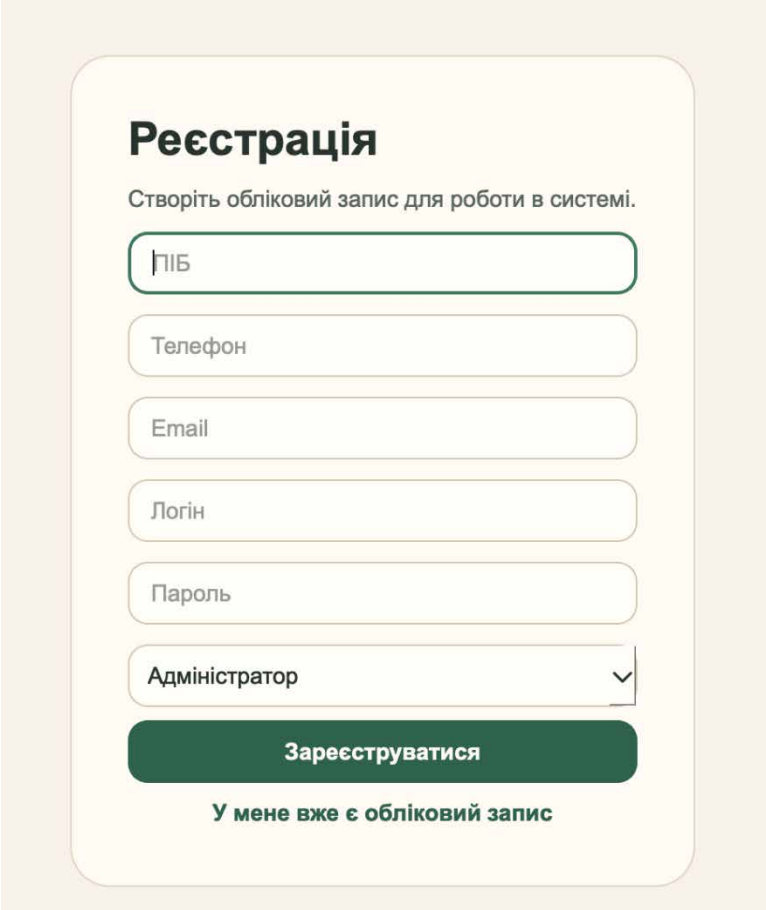

Рисунок 4.2 – Вікно реєстрації нового користувача

Після входу до системи відкривається головна панель, на якій відображаються основні показники роботи ресторану: кількість бронювань на сьогодні, кількість заявок, що очікують підтвердження, кількість прибулих гостей, кількість клієнтів у базі та активних столиків. Також відображається таблиця найближчих бронювань. Головну панель системи наведено на рисунку 4.3.

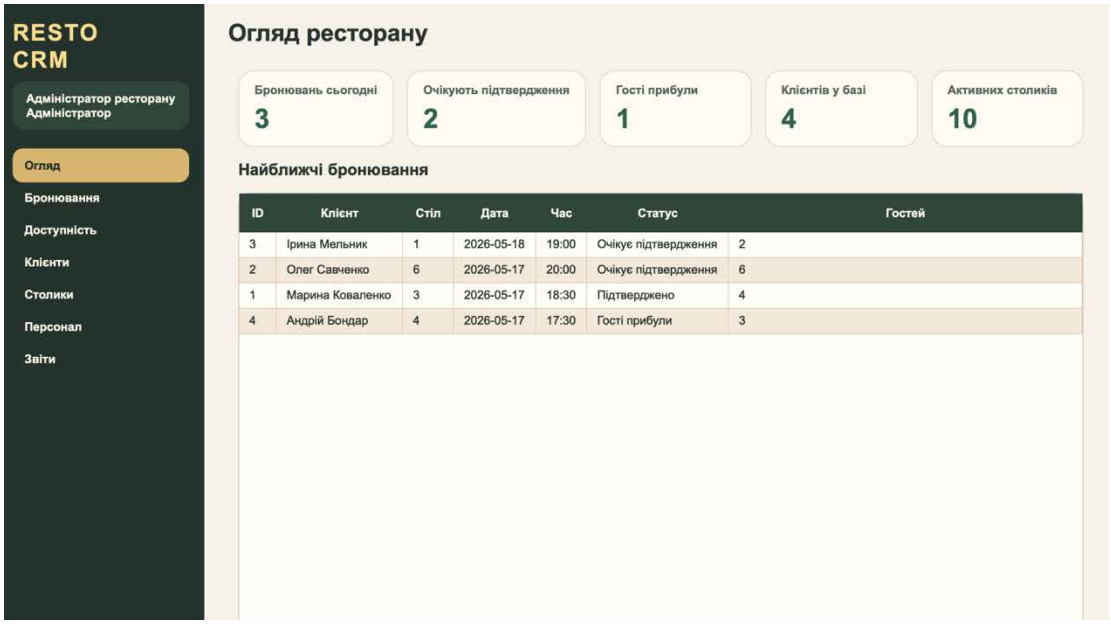

Рисунок 4.3 – Головна панель CRM-системи бронювання столиків

Наступним етапом було перевірено модуль бронювань. У цьому розділі адміністратор може переглядати список бронювань, фільтрувати записи за датою та статусом, виконувати пошук за клієнтом, телефоном або номером столика. Також доступні дії редагування, підтвердження, скасування, завершення бронювання та відмітка прибуття гостей. Приклад роботи модуля бронювань наведено на рисунку 4.4.

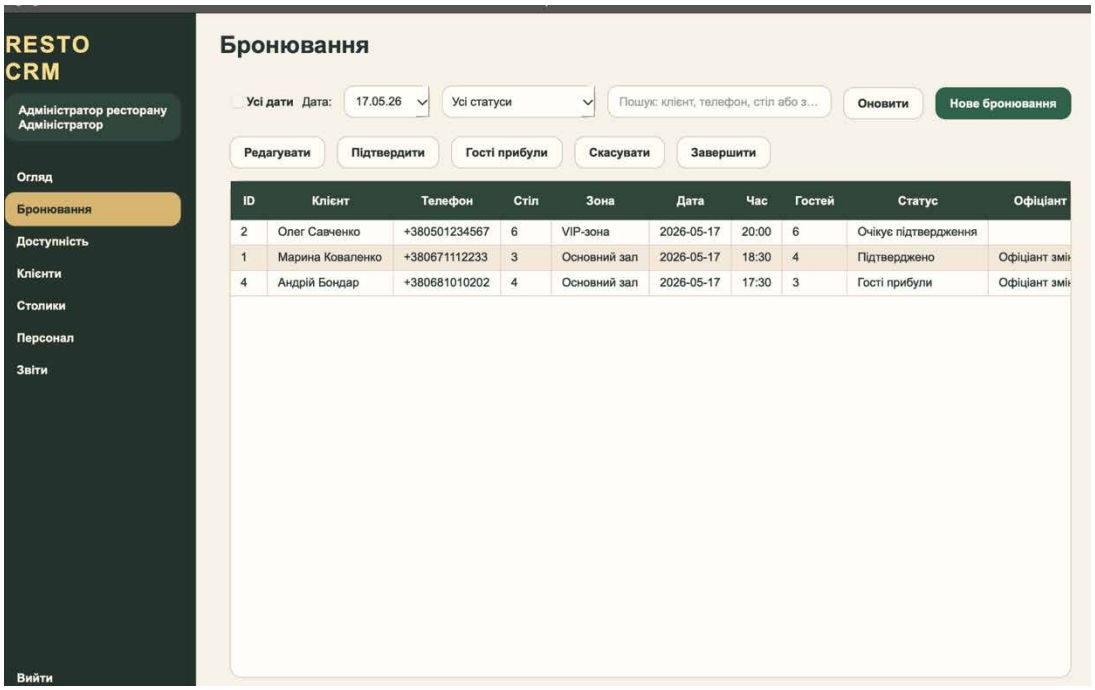

Рисунок 4.4 – Модуль перегляду та керування бронюваннями

Під час створення нового бронювання перевірялося заповнення форми, вибір клієнта, столика, офіціанта, дати, часу та кількості гостей. Також було перевірено можливість додавання нового клієнта безпосередньо під час оформлення бронювання. Це спрощує роботу адміністратора, оскільки не потрібно окремо переходити до клієнтської бази. Форму створення бронювання наведено на рисунку 4.5.

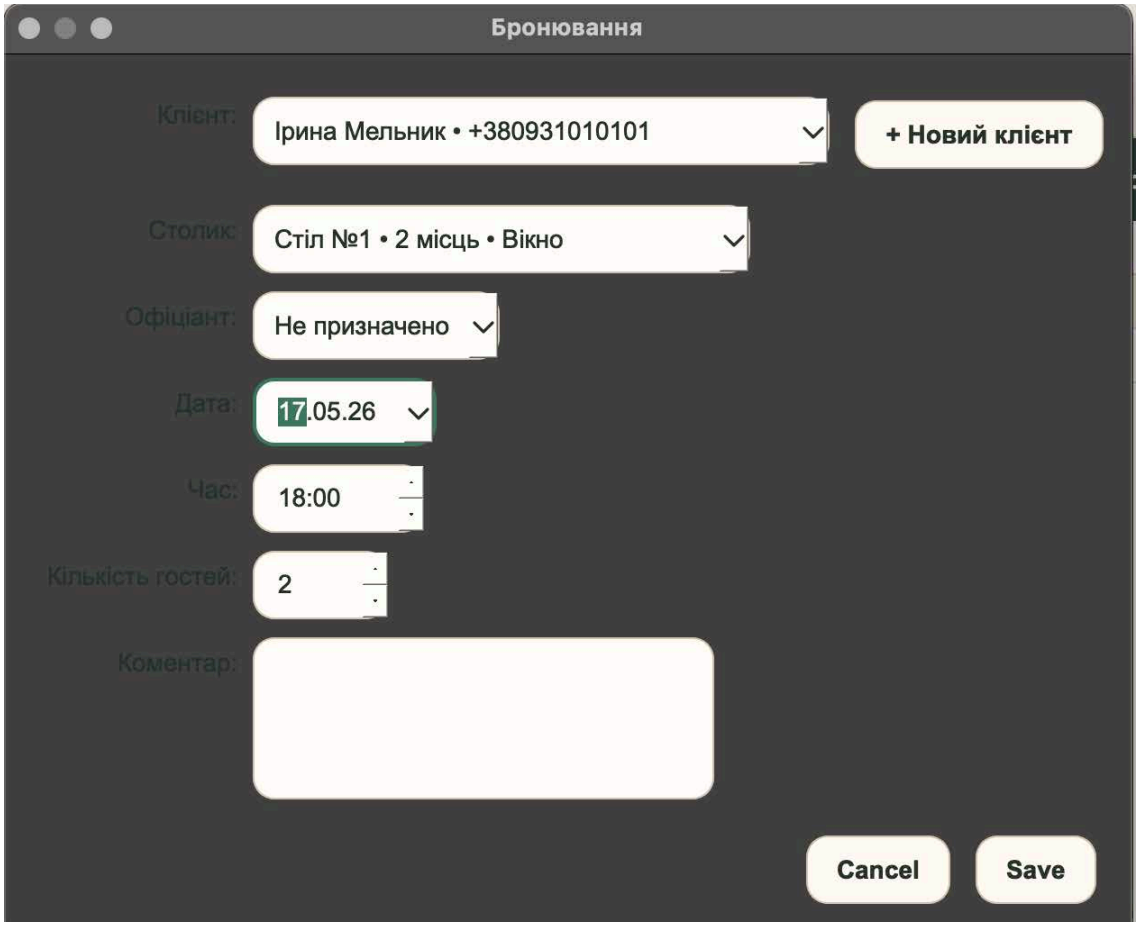

Рисунок 4.5 – Форма створення нового бронювання

Важливим етапом тестування була перевірка доступності столиків. Для цього в системі передбачено окремий модуль, у якому користувач обирає дату, час і кількість гостей. Після цього система показує список столиків, їх місткість, зону розміщення та статус доступності. Якщо столик уже зайнятий на вибраний час, система позначає його як недоступний. Модуль перевірки доступності столиків наведено на рисунку 4.6.

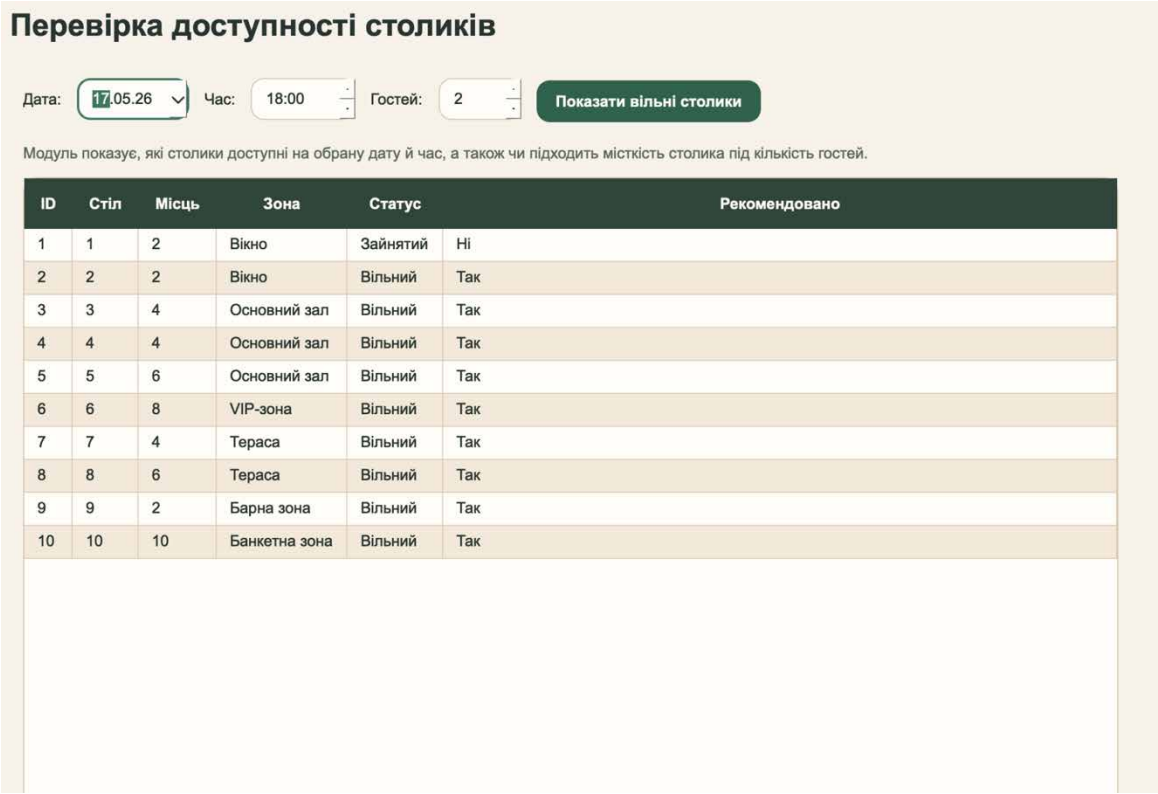

Рисунок 4.6 – Перевірка доступності столиків за датою, часом і кількістю гостей

Також було перевірено модуль клієнтської бази. У ньому адміністратор може додавати, редагувати та видаляти клієнтів. Для кожного клієнта зберігаються ПІБ, номер телефону, email і дата створення запису. Під час тестування перевірено, що новий клієнт після додавання одразу з’являється в таблиці та може бути використаний під час створення бронювання. Приклад роботи клієнтської бази наведено на рисунку 4.7.

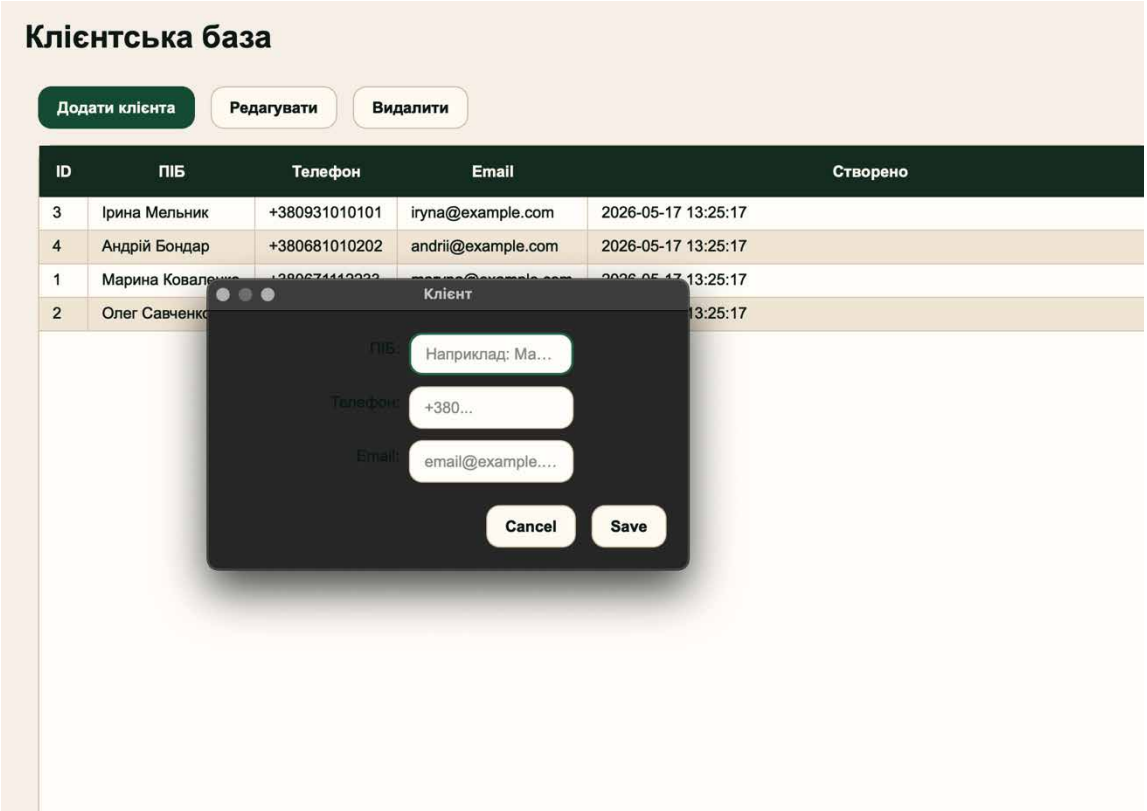

Рисунок 4.7 – Модуль клієнтської бази та форма додавання клієнта

Останнім етапом було протестовано модуль звітності та резервного копіювання. У цьому розділі система формує статистику за статусами бронювань і зонами ресторану. Також реалізовано експорт бронювань у CSV-файл і створення резервної копії бази даних. Це дозволяє адміністратору отримувати підсумкову інформацію та зберігати копії важливих даних. Модуль звітів і резервного копіювання наведено на рисунку 4.8.

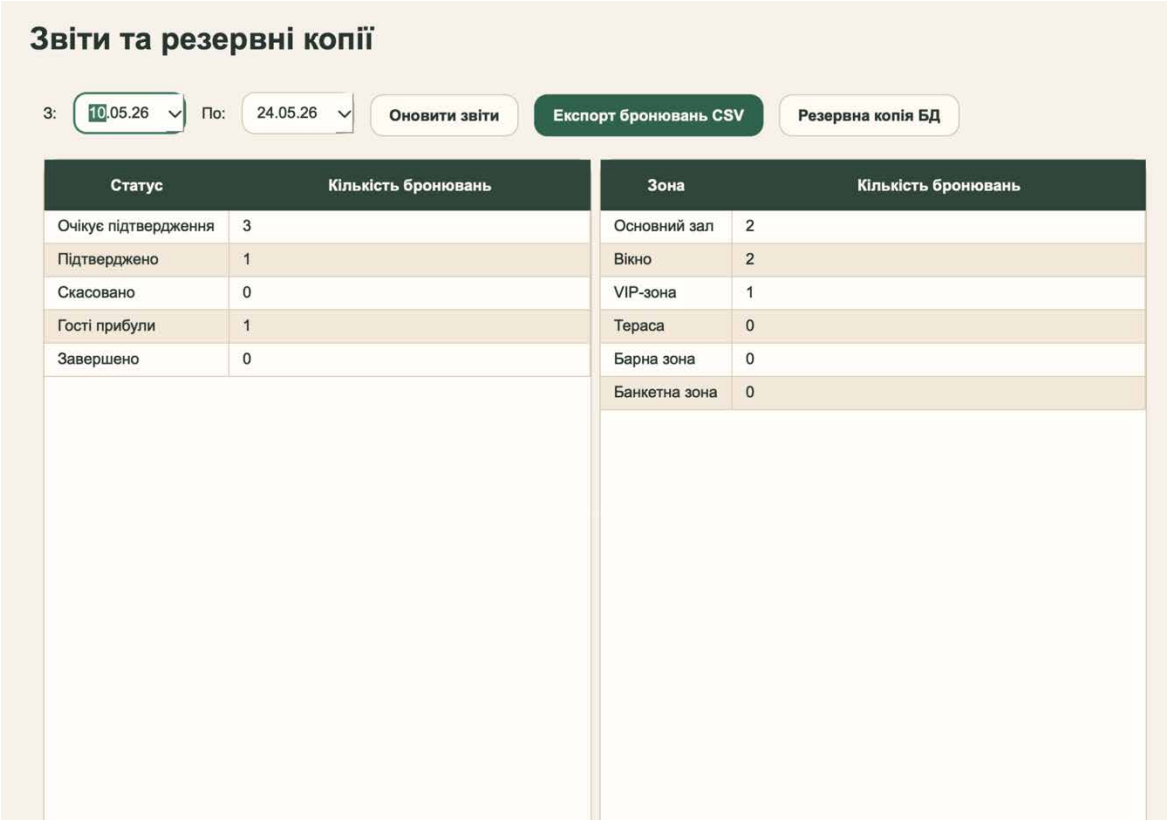

Рисунок 4.8 – Модуль звітності та резервного копіювання бази даних

Таблиця 4.4 – Результати тестування інтерфейсних модулів системи

№	Об’єкт тестування	Перевірена дія	Очікуваний результат	Результат
1	Вікно авторизації	Вхід за логіном і паролем	Користувач переходить до головного вікна	Виконано
2	Вікно реєстрації	Створення нового користувача	Обліковий запис додається до бази даних	Виконано
3	Головна панель	Відображення показників системи	Показники відповідають даним у БД	Виконано
4	Модуль бронювань	Перегляд і фільтрація бронювань	Записи коректно відображаються в таблиці	Виконано
5	Форма бронювання	Створення нового бронювання	Новий запис додається до списку бронювань	Виконано
6	Перевірка доступності	Пошук вільних столиків	Система показує зайняті та вільні столики	Виконано
7	Клієнтська база	Додавання та редагування клієнта	Дані клієнта зберігаються та оновлюються	Виконано
8	Звіти	Формування статистики	Дані групуються за статусами та зонами	Виконано
9	Експорт CSV	Збереження звіту у файл	Файл експорту створюється коректно	Виконано
10	Резервне копіювання	Створення копії бази даних	Копія БД зберігається у вибраному місці	Виконано

Під час тестування також перевірялося, чи не виникають помилки під час багаторазового переходу між розділами системи, повторного оновлення таблиць, відкриття діалогових вікон і зміни статусів бронювання. Інтерфейс працював стабільно, основні елементи керування були доступними, а дані після виконання операцій коректно оновлювалися в таблицях.

Таблиця 4.5 – Перевірка основних функціональних сценаріїв

№	Функціональний сценарій	Вхідні дані	Отриманий результат
1	Авторизація адміністратора	admin / admin123	Відкрито панель адміністратора
2	Реєстрація нового користувача	ПІБ, телефон, email, логін, пароль, роль	Користувача додано до системи
3	Створення бронювання	Клієнт, столик, дата, час, кількість гостей	Створено запис зі статусом очікування
4	Підтвердження бронювання	Обране бронювання	Статус змінено на «Підтверджено»
5	Відмітка прибуття гостей	Обране бронювання	Статус змінено на «Гості прибули»
6	Перевірка зайнятого столика	Дата, час, кількість гостей	Зайнятий столик позначено як недоступний
7	Додавання клієнта	ПІБ, телефон, email	Клієнта додано до клієнтської бази
8	Формування звітів	Період звіту	Відображено статистику за статусами і зонами
9	Експорт даних	Команда експорту CSV	Створено файл із бронюваннями
10	Резервне копіювання	Команда backup	Створено копію бази даних

За результатами тестування встановлено, що CRM-система бронювання столиків у ресторані виконує основні функції відповідно до поставлених вимог. Система забезпечує авторизацію та реєстрацію користувачів, керування бронюваннями, перевірку доступності столиків, ведення клієнтської бази, формування звітів і резервне копіювання даних. Виявлені під час попередньої перевірки недоліки інтерфейсу, пов’язані з недостатнім контрастом шрифтів, були усунені шляхом оновлення стилів оформлення.

4.3 Розгортання системи, склад інсталяційного пакету

Розгортання CRM-системи бронювання столиків у ресторані передбачає підготовку програмного середовища, бази даних, службових файлів застосунку та перевірку працездатності основних модулів після встановлення. Система може використовуватися як локальний настільний застосунок на базі PyQt6, а також може бути розширена до клієнт-серверної архітектури для роботи декількох користувачів у локальній мережі ресторану.

Загальну схему розгортання системи наведено на рисунку 4.9.

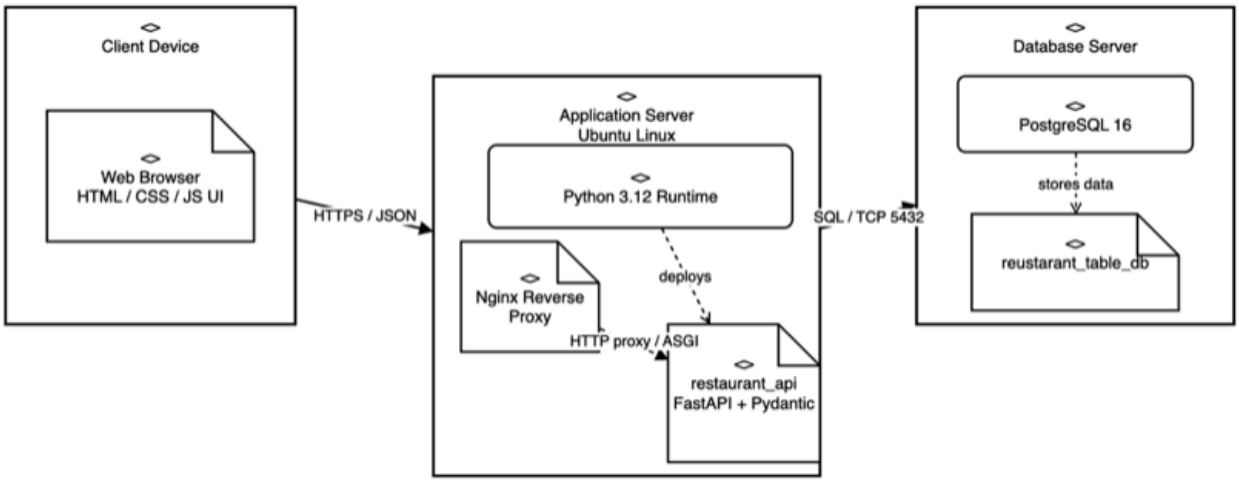

Рисунок 4.9 – Діаграма розгортання CRM-системи бронювання столиків у ресторані

На діаграмі показано три основні вузли розгортання: клієнтський пристрій, сервер застосунку та сервер бази даних. Клієнтський пристрій використовується адміністратором або персоналом ресторану для роботи з інтерфейсом системи. Сервер застосунку відповідає за виконання програмної логіки, оброблення запитів, керування бронюваннями, перевірку доступності столиків і формування звітів. Сервер бази даних забезпечує збереження інформації про клієнтів, столики, бронювання, статуси та працівників ресторану.

У клієнт-серверному варіанті розгортання обмін даними між клієнтом і сервером здійснюється через HTTPS у форматі JSON. Сервер застосунку працює в середовищі Python 3.12 та містить основні програмні компоненти системи. Для оброблення HTTP-запитів може використовуватися Nginx як зворотний проксі-

сервер, а прикладна логіка реалізується засобами Python. З'єднання із сервером бази даних здійснюється через SQL-запити.

Таблиця 4.6 – Основні вузли розгортання системи

Вузол розгортання	Призначення	Основні компоненти
Клієнтський пристрій	Робота користувача із системою	Інтерфейс користувача, модуль авторизації, модуль бронювання, перегляд звітів
Сервер застосунку	Виконання програмної логіки системи	Python runtime, UI-контролер, сервіс бронювання, сервіс столиків, сервіс звітності
Сервер бази даних	Збереження основних даних системи	Таблиці клієнтів, бронювань, столиків, статусів, персоналу
Резервне сховище	Збереження копій і службових файлів	Резервна копія БД, архіви звітів, файли експорту
Сервіс сповіщень	Інформування користувачів	Повідомлення про створення, підтвердження або зміну бронювання

Для локального варіанта використання CRM-система може запускатися без окремого сервера. У цьому випадку застосунок PyQt6 встановлюється на комп'ютер адміністратора ресторану, а база даних SQLite зберігається у файлі разом із програмою. Такий варіант є простим для впровадження в невеликому ресторані, оскільки не потребує складного адміністрування серверної інфраструктури.

Склад інсталяційного пакету наведено в таблиці 4.7.

Таблиця 4.7 – Склад інсталяційного пакету CRM-системи

Елемент пакету	Призначення
main.py	Основний файл запуску програмної системи
app/	Каталог із програмними модулями застосунку
database.py	Модуль створення, підключення та оброблення бази даних
auth_window.py	Модуль авторизації та реєстрації користувачів
main_window.py	Головне вікно CRM-системи
dialogs.py	Діалогові вікна для клієнтів, столиків і бронювань
security.py	Хешування паролів і перевірка облікових даних
styles.qss	Файл оформлення графічного інтерфейсу
restaurant_crm.db	Файл локальної бази даних SQLite
requirements.txt	Перелік необхідних бібліотек Python
run_windows.bat	Файл запуску системи в середовищі Windows
run_macos.command	Файл запуску системи в середовищі macOS

Перед запуском системи необхідно встановити Python 3 та бібліотеки, зазначені у файлі requirements.txt. Основною залежністю є PyQt6, яка забезпечує роботу графічного інтерфейсу. Після встановлення залежностей застосунок запускається через файл main.py або через підготовлений файл запуску для відповідної операційної системи.

Послідовність встановлення системи наведено в таблиці 4.8.

Таблиця 4.8 – Послідовність розгортання CRM-системи

№	Етап	Опис дії
1	Підготовка середовища	Встановлення Python 3 та перевірка його доступності
2	Розпакування проєкту	Розміщення файлів системи в окремому каталозі
3	Встановлення залежностей	Виконання команди pip install -r requirements.txt
4	Ініціалізація бази даних	Автоматичне створення таблиць під час першого запуску
5	Запуск системи	Виконання файлу main.py або запуск через підготовлений скрипт
6	Перевірка входу	Авторизація адміністратора або офіціанта
7	Контроль працездатності	Перевірка бронювань, клієнтів, столиків, звітів і backup

Після першого запуску система автоматично створює необхідні таблиці бази даних, додає базові статуси бронювання, тестові облікові записи та початкові дані для перевірки роботи. Це спрощує встановлення системи та дозволяє одразу перейти до тестування основних функцій.

До мінімальних вимог для локального запуску належать: операційна система Windows, macOS або Linux, встановлений Python 3.10 або новіший, бібліотека PyQt6, не менше 4 ГБ оперативної пам’яті та наявність вільного місця для файлу бази даних і резервних копій. Для мережевого варіанта додатково потрібен сервер застосунку, сервер бази даних та налаштування доступу через локальну мережу або HTTPS.

Отже, розгортання CRM-системи бронювання столиків у ресторані передбачає підготовку програмного середовища, встановлення залежностей, запуск застосунку та автоматичну ініціалізацію бази даних. Склад інсталяційного пакету є достатнім для локального використання системи, а

запропонована архітектура дозволяє в подальшому масштабувати її до клієнт-серверного рішення для роботи декількох користувачів у межах ресторану.

4.4 Висновки до четвертого розділу

У четвертому розділі було виконано тестування та оцінювання ефективності CRM-системи бронювання столиків у ресторані. Основну увагу приділено перевірці працездатності програмних модулів, коректності роботи інтерфейсу, взаємодії з базою даних, авторизації користувачів, створенню бронювань, керуванню клієнтами, столиками, персоналом, звітністю та резервним копіюванням.

У пункті 4.1 було сформовано план тестування програмних модулів системи. Визначено основні об’єкти перевірки: модуль авторизації, реєстрації, клієнтів, столиків, бронювань, перевірки доступності, офіціанта, звітності та резервного копіювання. Для кожного модуля встановлено очікувані результати, що дозволило системно перевірити відповідність реалізованого програмного забезпечення функціональним вимогам.

У пункті 4.2 проведено практичне тестування системи на основі основних сценаріїв роботи користувача. Перевірено вхід адміністратора й офіціанта, створення нового облікового запису, перегляд головної панелі, роботу з бронюваннями, додавання клієнтів, перевірку доступності столиків, зміну статусів бронювання, формування звітів і створення резервної копії бази даних. За результатами тестування встановлено, що основні функції системи виконуються коректно, а дані після операцій оновлюються в базі даних і відображаються в інтерфейсі.

У пункті 4.3 описано розгортання системи та склад інсталяційного пакету. Визначено, що система може використовуватися як локальний настільний застосунок на базі Python, PyQt6 і SQLite, а також має можливість подальшого масштабування до клієнт-серверної архітектури. Описано основні файли

проєкту, програмні модулі, залежності, файл бази даних, стилі інтерфейсу, скрипти запуску та порядок встановлення системи.

Під час тестування підтверджено, що CRM-система забезпечує повний цикл роботи з бронюваннями: створення заявки, перевірку доступності столика, підтвердження адміністратором, відмітку прибуття гостей офіціантом, скасування або завершення бронювання. Також система підтримує ведення клієнтської бази, облік столиків, розмежування ролей користувачів, формування звітної інформації та резервне копіювання даних.

Отже, результати четвертого розділу підтверджують працездатність і практичну придатність розробленої CRM-системи бронювання столиків у ресторані. Проведене тестування показало, що система відповідає поставленим вимогам, має зрозумілий інтерфейс, забезпечує коректну роботу з даними та може бути використана для автоматизації основних процесів бронювання в закладах ресторанного господарства.

ВИСНОВКИ

У кваліфікаційній роботі було вирішено актуальне практичне завдання розроблення CRM-системи бронювання столиків у ресторані. Актуальність роботи зумовлена потребою закладів ресторанного господарства в автоматизації процесів приймання бронювань, контролю доступності столиків, ведення клієнтської бази, організації роботи персоналу та формування звітної інформації.

У першому розділі проведено системний аналіз предметної області. Було розглянуто особливості процесу бронювання столиків у ресторані, визначено основних учасників системи: клієнта, адміністратора та офіціанта. Проаналізовано існуючі рішення, зокрема OpenTable, Resy, Quandoo та Yelp Reservations. У результаті встановлено, що наявні системи мають достатню функціональність, однак часто залежать від сторонніх сервісів, потребують комерційної підписки та не завжди є зручними для локального використання в невеликих або середніх ресторанах. Це обґрунтувало доцільність створення власної CRM-системи.

У межах моделювання предметної області було побудовано DFD-діаграму, діаграму прецедентів, діаграму послідовності та діаграму діяльності. Вони дозволили формалізувати основні процеси системи: створення бронювання, перевірку доступності столиків, підтвердження бронювання адміністратором, відмітку прибуття гостей офіціантом і зміну статусів бронювання. Також було сформовано функціональні, нефункціональні, технічні та безпекові вимоги до розроблюваної системи.

У другому розділі виконано проєктування інформаційного та програмного забезпечення. Було розроблено логічну модель даних у вигляді ER-діаграми, у якій визначено основні сутності: клієнт, бронювання, столик, статус бронювання, адміністратор та офіціант. Побудовано діаграму класів, яка відображає об’єктну структуру системи, а також три кооперації для основних сценаріїв: створення бронювання клієнтом, підтвердження бронювання

адміністратором і відмітка прибуття гостей офіціантом. Додатково розроблено діаграму компонентів і діаграму пакетів, що визначають модульну структуру програмного забезпечення.

У третьому розділі обґрунтовано вибір технологічних засобів реалізації. Для створення системи обрано мову Python, бібліотеку PyQt6 для розроблення графічного інтерфейсу, SQLite для локального збереження даних, QSS для стилізації інтерфейсу та PyInstaller для можливого формування інсталяційного пакету. Такий стек технологій є доцільним для настільної CRM-системи, оскільки забезпечує швидкий запуск, автономність, зручну роботу з базою даних і можливість подальшого розширення.

Було спроєктовано архітектуру системи, яка передбачає поділ на інтерфейс користувача, модулі бізнес-логіки, рівень доступу до даних, базу даних і допоміжні функції звітності та резервного копіювання. Також розроблено фізичну модель бази даних із таблицями customers, restaurant_tables, reservation_statuses, admins, waiters і reservations. У моделі передбачено первинні та зовнішні ключі, індекси й унікальні обмеження, що дозволяють уникнути дублювання бронювань одного столика на однакову дату й час.

Практичним результатом роботи стала реалізована CRM-система бронювання столиків у ресторані. У програмі реалізовано авторизацію та реєстрацію користувачів, розмежування ролей адміністратора й офіціанта, керування бронюваннями, клієнтами, столиками та персоналом, перевірку доступності столиків, зміну статусів бронювання, формування звітів, експорт даних у CSV і створення резервної копії бази даних. Інтерфейс системи виконано у мінімалістичному стилі, орієнтованому на зручність щоденної роботи персоналу ресторану.

У четвертому розділі проведено тестування системи. Було перевірено основні програмні модулі: авторизацію, реєстрацію, роботу з клієнтами, столиками, бронюваннями, доступністю столиків, звітами та резервним копіюванням. Результати тестування показали, що система коректно виконує

основні функції, оновлює дані в базі, запобігає дублюванню бронювань і забезпечує стабільну роботу інтерфейсу.

Отже, мету кваліфікаційної роботи досягнуто. Розроблена CRM-система бронювання столиків у ресторані забезпечує автоматизацію ключових процесів ресторанного обслуговування, підвищує зручність керування бронюваннями, спрощує роботу адміністратора та офіціанта, зменшує ризик помилок під час ручного обліку й створює основу для подальшого розвитку системи. У перспективі програму можна розширити за рахунок веб-кабінету клієнта, онлайн-оплати, SMS-сповіщень, інтеграції з CRM або POS-системою ресторану та аналітики завантаженості залу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 16 с.

2. ДСТУ ISO/IEC/IEEE 12207:2018. Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів. Київ : ДП «УкрНДНЦ», 2018.

3. ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes. Geneva : International Organization for Standardization, 2017. URL: https://www.iso.org/standard/63712.html

4. ISO/IEC 25010:2023. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model. Geneva : International Organization for Standardization, 2023. URL: https://www.iso.org/standard/78176.html

5. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2020. 671 p.

6. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016. 816 p.

7. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill Education, 2020. 1376 p. URL: https://www.db-book.com/

8. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Boston : Pearson, 2015. 1440 p.

9. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2003. 208 p.

10. Unified Modeling Language. Object Management Group : офіційний вебсайт. URL: https://www.omg.org/spec/UML/

11. Python 3 Documentation. Python Software Foundation : офіційний вебсайт. URL: https://docs.python.org/3/

12. sqlite3 — DB-API 2.0 interface for SQLite databases. Python Software Foundation : офіційний вебсайт. URL: https://docs.python.org/3/library/sqlite3.html

13. SQLite Documentation. SQLite : офіційний вебсайт. URL: https://sqlite.org/docs.html

14. About SQLite. SQLite : офіційний вебсайт. URL: https://sqlite.org/about.html

15. PyQt6. Python Package Index : вебсайт. URL: https://pypi.org/project/PyQt6/

16. PyQt Components. Riverbank Computing : офіційний вебсайт. URL: https://riverbankcomputing.com/software/pyqt/intro

17. Qt 6 Documentation. The Qt Company : офіційний вебсайт. URL: https://doc.qt.io/qt-6/

18. Qt Style Sheets Reference. The Qt Company : офіційний вебсайт. URL: https://doc.qt.io/qt-6/stylesheet-reference.html

19. PyInstaller Manual. PyInstaller : офіційний вебсайт. URL: https://pyinstaller.org/en/stable/

20. OWASP Top 10:2021. Open Worldwide Application Security Project : офіційний вебсайт. URL: https://owasp.org/Top10/

21. Nielsen J. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group : вебсайт. URL: https://www.nngroup.com/articles/ten-usability-heuristics/

22. OpenTable: Restaurants and Restaurant Bookings. OpenTable : офіційний вебсайт. URL: https://www.opentable.com/

23. Restaurant Reservation Software & Operations Systems. OpenTable : офіційний вебсайт. URL: https://www.opentable.com/restaurant-solutions/

24. Restaurant Management Solutions. OpenTable : офіційний вебсайт. URL: https://www.opentable.com/restaurant-solutions/our-solutions/

25. Resy. Restaurant Reservation and Table Management Platform : офіційний вебсайт. URL: https://resy.com/

26. Quandoo. Restaurant Reservation Platform : офіційний вебсайт. URL: https://www.quandoo.com/

27. Yelp for Restaurants. Yelp : офіційний вебсайт. URL: https://restaurants.yelp.com/

28. PostgreSQL Documentation. PostgreSQL Global Development Group : офіційний вебсайт. URL: https://www.postgresql.org/docs/

29. MySQL 8.4 Reference Manual. Oracle : офіційний вебсайт. URL: https://dev.mysql.com/doc/refman/8.4/en/

30. Microsoft. UML diagramming and database modeling guide. Microsoft 365 : вебсайт. URL: https://www.microsoft.com/en-us/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling

ДОДАТОК А

Програмний код створення бази даних CRM-системи бронювання столиків у ресторані

```
PRAGMA foreign_keys = ON;

CREATE TABLE IF NOT EXISTS customers (
    customer_id INTEGER PRIMARY KEY AUTOINCREMENT,
    full_name TEXT NOT NULL,
    phone TEXT NOT NULL UNIQUE,
    email TEXT,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS restaurant_tables (
    table_id INTEGER PRIMARY KEY AUTOINCREMENT,
    table_number INTEGER NOT NULL UNIQUE,
    seats_count INTEGER NOT NULL CHECK (seats_count > 0),
    zone TEXT NOT NULL,
    is_active INTEGER NOT NULL DEFAULT 1 CHECK (is_active IN (0, 1))
);

CREATE TABLE IF NOT EXISTS reservation_statuses (
    status_id INTEGER PRIMARY KEY AUTOINCREMENT,
    status_name TEXT NOT NULL UNIQUE,
    description TEXT
);

CREATE TABLE IF NOT EXISTS admins (
    admin_id INTEGER PRIMARY KEY AUTOINCREMENT,
    full_name TEXT NOT NULL,
    phone TEXT,
    email TEXT UNIQUE,
    login TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    is_active INTEGER NOT NULL DEFAULT 1 CHECK (is_active IN (0, 1))
);

CREATE TABLE IF NOT EXISTS waiters (
    waiter_id INTEGER PRIMARY KEY AUTOINCREMENT,
    full_name TEXT NOT NULL,
    phone TEXT,
    shift TEXT,
    is_active INTEGER NOT NULL DEFAULT 1 CHECK (is_active IN (0, 1))
);

CREATE TABLE IF NOT EXISTS reservations (
    reservation_id INTEGER PRIMARY KEY AUTOINCREMENT,
    customer_id INTEGER NOT NULL,
    table_id INTEGER NOT NULL,
    status_id INTEGER NOT NULL DEFAULT 1,
    admin_id INTEGER,
    waiter_id INTEGER,
    reservation_date TEXT NOT NULL,
    reservation_time TEXT NOT NULL,
    guest_count INTEGER NOT NULL CHECK (guest_count > 0),
    booking_comment TEXT,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
```

```
        ON UPDATE CASCADE ON DELETE RESTRICT,
    FOREIGN KEY (table_id) REFERENCES restaurant_tables(table_id)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    FOREIGN KEY (status_id) REFERENCES reservation_statuses(status_id)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    FOREIGN KEY (admin_id) REFERENCES admins(admin_id)
        ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (waiter_id) REFERENCES waiters(waiter_id)
        ON UPDATE CASCADE ON DELETE SET NULL
);

CREATE UNIQUE INDEX IF NOT EXISTS ux_active_table_slot
ON reservations(table_id, reservation_date, reservation_time)
WHERE status_id IN (1, 2, 3);

CREATE INDEX IF NOT EXISTS idx_reservation_date
ON reservations(reservation_date);

CREATE INDEX IF NOT EXISTS idx_reservation_customer
ON reservations(customer_id);

CREATE INDEX IF NOT EXISTS idx_reservation_status
ON reservations(status_id);

INSERT OR IGNORE INTO reservation_statuses(status_id, status_name, description)
VALUES
(1, 'Очікує підтвердження', 'Бронювання створене клієнтом і очікує перевірки
адміністратором'),
(2, 'Підтверджено', 'Бронювання підтверджене адміністратором ресторану'),
(3, 'Гості прибули', 'Офіціант відмітив прибуття гостей'),
(4, 'Скасовано', 'Бронювання скасоване клієнтом або адміністратором');

INSERT OR IGNORE INTO restaurant_tables(table_id, table_number, seats_count,
zone, is_active) VALUES
(1, 1, 2, 'Основний зал', 1),
(2, 2, 4, 'Основний зал', 1),
(3, 3, 4, 'Основний зал', 1),
(4, 4, 6, 'VIP-зона', 1),
(5, 5, 8, 'Банкетна зона', 1);

INSERT OR IGNORE INTO admins(admin_id, full_name, phone, email, login,
password_hash, is_active) VALUES
(1, 'Адміністратор ресторану', '+380501112233', 'admin@restaurant.local',
'admin', 'admin123', 1);

INSERT OR IGNORE INTO waiters(waiter_id, full_name, phone, shift, is_active)
VALUES
(1, 'Офіціант зміни 1', '+380671112233', 'Денна', 1),
(2, 'Офіціант зміни 2', '+380931112233', 'Вечірня', 1);
```

ДОДАТОК Б

Програмний код модуля роботи з базою даних

```
import sqlite3
from pathlib import Path


class Database:
    def __init__(self, db_path: str = "restaurant_crm.db"):
        self.db_path = db_path

    def connect(self):
        connection = sqlite3.connect(self.db_path)
        connection.row_factory = sqlite3.Row
        connection.execute("PRAGMA foreign_keys = ON")
        return connection

    def init_schema(self, schema_path: str = "schema.sql"):
        path = Path(schema_path)
        if not path.exists():
            raise FileNotFoundError("Файл schema.sql не знайдено")

        with self.connect() as connection:
            connection.executescript(path.read_text(encoding="utf-8"))
            connection.commit()


class ReservationRepository:
    def __init__(self, database: Database):
        self.database = database

    def create_customer(self, full_name: str, phone: str, email: str | None =
None) -> int:
        with self.database.connect() as connection:
            existing = connection.execute(
                "SELECT customer_id FROM customers WHERE phone = ?",
                (phone,)
            ).fetchone()

            if existing:
                return existing["customer_id"]

            cursor = connection.execute(
                """
                INSERT INTO customers(full_name, phone, email)
                VALUES (?, ?, ?)
                """,
                (full_name, phone, email)
            )
            connection.commit()
            return cursor.lastrowid

    def get_available_tables(self, reservation_date: str, reservation_time: str,
guest_count: int):
        with self.database.connect() as connection:
            return connection.execute(
                """
                SELECT table_id, table_number, seats_count, zone
                FROM restaurant_tables
```

```
                WHERE is_active = 1
                  AND seats_count >= ?
                  AND table_id NOT IN (
                      SELECT table_id
                      FROM reservations
                      WHERE reservation_date = ?
                        AND reservation_time = ?
                        AND status_id IN (1, 2, 3)
                  )
                ORDER BY seats_count, table_number
                """,
                (guest_count, reservation_date, reservation_time)
            ).fetchall()

    def create_reservation(
        self,
        full_name: str,
        phone: str,
        email: str,
        table_id: int,
        reservation_date: str,
        reservation_time: str,
        guest_count: int,
        comment: str = ""
    ) -> int:
        customer_id = self.create_customer(full_name, phone, email)

        with self.database.connect() as connection:
            try:
                cursor = connection.execute(
                    """
                    INSERT INTO reservations(
                        customer_id, table_id, status_id,
                        reservation_date, reservation_time,
                        guest_count, booking_comment
                    )
                    VALUES (?, ?, 1, ?, ?, ?, ?)
                    """,
                    (
                        customer_id,
                        table_id,
                        reservation_date,
                        reservation_time,
                        guest_count,
                        comment
                    )
                )
                connection.commit()
                return cursor.lastrowid
            except sqlite3.IntegrityError as error:
                raise ValueError("Обраний столик уже заброньований на цей час")
from error

    def get_reservations(self):
        with self.database.connect() as connection:
            return connection.execute(
                """
                SELECT
                    r.reservation_id,
                    c.full_name AS customer_name,
                    c.phone,
                    rt.table_number,
                    rt.zone,
```

```
                r.reservation_date,
                r.reservation_time,
                r.guest_count,
                rs.status_name,
                r.booking_comment
            FROM reservations r
            JOIN customers c ON c.customer_id = r.customer_id
            JOIN restaurant_tables rt ON rt.table_id = r.table_id
            JOIN reservation_statuses rs ON rs.status_id = r.status_id
            ORDER BY r.reservation_date DESC, r.reservation_time DESC
            """
        ).fetchall()

def confirm_reservation(self, reservation_id: int, admin_id: int = 1):
    with self.database.connect() as connection:
        connection.execute(
            """
            UPDATE reservations
            SET status_id = 2, admin_id = ?
            WHERE reservation_id = ?
            """,
            (admin_id, reservation_id)
        )
        connection.commit()

def mark_arrival(self, reservation_id: int, waiter_id: int = 1):
    with self.database.connect() as connection:
        connection.execute(
            """
            UPDATE reservations
            SET status_id = 3, waiter_id = ?
            WHERE reservation_id = ?
            """,
            (waiter_id, reservation_id)
        )
        connection.commit()

def cancel_reservation(self, reservation_id: int):
    with self.database.connect() as connection:
        connection.execute(
            """
            UPDATE reservations
            SET status_id = 4
            WHERE reservation_id = ?
            """,
            (reservation_id,)
        )
        connection.commit()
```

ДОДАТОК В

Програмний код головного вікна CRM-системи бронювання столиків

```
import sys
from PyQt6.QtCore import QDate, QTime
from PyQt6.QtWidgets import (
    QApplication,
    QComboBox,
    QDateEdit,
    QFormLayout,
    QHBoxLayout,
    QHeaderView,
    QLabel,
    QLineEdit,
    QMainWindow,
    QMessageBox,
    QPushButton,
    QSpinBox,
    QTableWidget,
    QTableWidgetItem,
    QTextEdit,
    QTimeEdit,
    QVBoxLayout,
    QWidget
)

from database import Database, ReservationRepository


class ReservationWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.database = Database()
        self.database.init_schema()
        self.repository = ReservationRepository(self.database)

        self.setWindowTitle("CRM-система бронювання столиків у ресторані")
        self.resize(1050, 650)

        self.full_name_input = QLineEdit()
        self.phone_input = QLineEdit()
        self.email_input = QLineEdit()

        self.date_input = QDateEdit()
        self.date_input.setCalendarPopup(True)
        self.date_input.setDate(QDate.currentDate())

        self.time_input = QTimeEdit()
        self.time_input.setTime(QTime.currentTime())

        self.guest_count_input = QSpinBox()
        self.guest_count_input.setMinimum(1)
        self.guest_count_input.setMaximum(20)
        self.guest_count_input.setValue(2)

        self.table_select = QComboBox()
        self.comment_input = QTextEdit()
```

```
        self.comment_input.setMaximumHeight(80)

        self.reservations_table = QTableWidget()
        self.reservations_table.setColumnCount(9)
        self.reservations_table.setHorizontalHeaderLabels([
            "ID",
            "Клієнт",
            "Телефон",
            "Столик",
            "Зона",
            "Дата",
            "Час",
            "Гостей",
            "Статус"
        ])

self.reservations_table.horizontalHeader().setSectionResizeMode(QHeaderView.Resi
zeMode.Stretch)

        self.build_ui()
        self.load_available_tables()
        self.load_reservations()

    def build_ui(self):
        form = QFormLayout()
        form.addRow("ПІБ клієнта:", self.full_name_input)
        form.addRow("Телефон:", self.phone_input)
        form.addRow("Email:", self.email_input)
        form.addRow("Дата бронювання:", self.date_input)
        form.addRow("Час бронювання:", self.time_input)
        form.addRow("Кількість гостей:", self.guest_count_input)
        form.addRow("Доступний столик:", self.table_select)
        form.addRow("Коментар:", self.comment_input)

        check_button = QPushButton("Перевірити доступність")
        check_button.clicked.connect(self.load_available_tables)

        create_button = QPushButton("Створити бронювання")
        create_button.clicked.connect(self.create_reservation)

        confirm_button = QPushButton("Підтвердити")
        confirm_button.clicked.connect(self.confirm_selected_reservation)

        arrival_button = QPushButton("Відмітити прибуття")
        arrival_button.clicked.connect(self.mark_selected_arrival)

        cancel_button = QPushButton("Скасувати")
        cancel_button.clicked.connect(self.cancel_selected_reservation)

        buttons = QHBoxLayout()
        buttons.addWidget(check_button)
        buttons.addWidget(create_button)
        buttons.addWidget(confirm_button)
        buttons.addWidget(arrival_button)
        buttons.addWidget(cancel_button)

        layout = QVBoxLayout()
        layout.addWidget(QLabel("Форма створення бронювання"))
        layout.addLayout(form)
        layout.addLayout(buttons)
        layout.addWidget(QLabel("Журнал бронювань"))
        layout.addWidget(self.reservations_table)
```

```
        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

    def get_date(self) -> str:
        return self.date_input.date().toString("yyyy-MM-dd")

    def get_time(self) -> str:
        return self.time_input.time().toString("HH:mm")

    def load_available_tables(self):
        self.table_select.clear()

        tables = self.repository.get_available_tables(
            self.get_date(),
            self.get_time(),
            self.guest_count_input.value()
        )

        for table in tables:
            label = f"Стіл {table['table_number']} - {table['seats_count']}
місць - {table['zone']}"
            self.table_select.addItem(label, table["table_id"])

        if not tables:
            self.table_select.addItem("Немає доступних столиків", None)

    def create_reservation(self):
        if not self.full_name_input.text().strip():
            QMessageBox.warning(self, "Помилка", "Введіть ПІБ клієнта")
            return

        if not self.phone_input.text().strip():
            QMessageBox.warning(self, "Помилка", "Введіть номер телефону")
            return

        table_id = self.table_select.currentData()
        if table_id is None:
            QMessageBox.warning(self, "Помилка", "Немає доступного столика для
бронювання")
            return

        try:
            reservation_id = self.repository.create_reservation(
                full_name=self.full_name_input.text().strip(),
                phone=self.phone_input.text().strip(),
                email=self.email_input.text().strip(),
                table_id=table_id,
                reservation_date=self.get_date(),
                reservation_time=self.get_time(),
                guest_count=self.guest_count_input.value(),
                comment=self.comment_input.toPlainText().strip()
            )

            QMessageBox.information(
                self,
                "Успішно",
                f"Бронювання №{reservation_id} створено"
            )

            self.clear_form()
            self.load_available_tables()
            self.load_reservations()
```

```
        except ValueError as error:
            QMessageBox.warning(self, "Помилка бронювання", str(error))

    def load_reservations(self):
        reservations = self.repository.get_reservations()
        self.reservations_table.setRowCount(len(reservations))

        for row_index, reservation in enumerate(reservations):
            values = [
                reservation["reservation_id"],
                reservation["customer_name"],
                reservation["phone"],
                reservation["table_number"],
                reservation["zone"],
                reservation["reservation_date"],
                reservation["reservation_time"],
                reservation["guest_count"],
                reservation["status_name"]
            ]

            for column_index, value in enumerate(values):
                self.reservations_table.setItem(
                    row_index,
                    column_index,
                    QTableWidgetItem(str(value))
                )

    def get_selected_reservation_id(self):
        row = self.reservations_table.currentRow()
        if row < 0:
            QMessageBox.warning(self, "Помилка", "Оберіть бронювання в таблиці")
            return None

        item = self.reservations_table.item(row, 0)
        return int(item.text())

    def confirm_selected_reservation(self):
        reservation_id = self.get_selected_reservation_id()
        if reservation_id is None:
            return

        self.repository.confirm_reservation(reservation_id)
        self.load_reservations()
        QMessageBox.information(self, "Готово", "Бронювання підтверджено")

    def mark_selected_arrival(self):
        reservation_id = self.get_selected_reservation_id()
        if reservation_id is None:
            return

        self.repository.mark_arrival(reservation_id)
        self.load_reservations()
        QMessageBox.information(self, "Готово", "Прибуття гостей зафіксовано")

    def cancel_selected_reservation(self):
        reservation_id = self.get_selected_reservation_id()
        if reservation_id is None:
            return

        self.repository.cancel_reservation(reservation_id)
        self.load_available_tables()
        self.load_reservations()
```

```
            QMessageBox.information(self, "Готово", "Бронювання скасовано")

        def clear_form(self):
            self.full_name_input.clear()
            self.phone_input.clear()
            self.email_input.clear()
            self.comment_input.clear()
            self.guest_count_input.setValue(2)


    def main():
        app = QApplication(sys.argv)
        window = ReservationWindow()
        window.show()
        sys.exit(app.exec())


    if __name__ ==
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Дробязко Марія Володимирівна, здобувачка НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Система бронювання столику в ресторані» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - ○ [ ] інструменти генеративного ШІ не використовувалися.
    - ○ [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - ■ [ ] перекладу іншомовних джерел;
        - ■ [ ] стилістичного редагування та перевірки граматики;
        - ■ [+] допомоги у написанні/відлагодженні програмного коду;
        - ■ [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

« » 2026 р. _____ **Дробязко Марія**
(підпис)