**Міністерство освіти і науки України**
**Луцький національний технічний університет**
**Факультет комп'ютерних та інформаційних технологій**
**Кафедра інженерії програмного забезпечення**

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

# РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ КЕРУВАННЯ ЗАМОВЛЕННЯМИ В РЕСТОРАНІ З ВИКОРИСТАННЯМ REACT, NODE.JS ТА MYSQL

# DEVELOPMENT OF A WEB SERVICE FOR RESTAURANT ORDER MANAGEMENT USING REACT, NODE.JS AND MYSQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
**Гапонюк Олександр Сергійович**

Керівник:
**Гордєєв Олександр Олександрович**

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

**ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення*»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ**

Гапонюку Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка вебсервісу для керування замовленнями в ресторані з використанням React, Node.js та MySQL»*
Керівник роботи: *професор Гордєєв О. О.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Visual Studio Code, Node.js, React.js, MySQL, методичні вказівки до виконання кваліфікаційної роботи бакалавра*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз існуючих систем керування замовленнями, постановка завдання, визначення вимог до системи, проєктування архітектури вебзастосунку, вибір технологій і середовищ розробки, реалізація клієнтської та серверної частин, розробка бази даних, тестування і налагодження, забезпечення безпеки, створення документації*
5. Перелік графічного матеріалу: *10 рисунків, 2 таблиці*

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Аналіз предметної області | /Гордєєв О. О. | | |
| Специфікація вимог до розробленої системи | /Гордєєв О. О. | | |
| Розробка об`єкта проектування | /Гордєєв О. О. | | |
| Нормоконтроль | Вознюк А. В. | | |
| Гарант ОП | Ліщина Н. М. | | |
| Показник запозичень тексту | 1,38 % | | |
| Академічна доброчесність | /Гордєєв О. О. | | |

7. Дата видачі завдання *«20» грудня 2024 р.*

| № | Назва етапів кваліфікаційної роботи бакалавра | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Огляд літературних джерел по темі кваліфікаційної роботи бакалавра | до 04.02.2025 р. | Викон. |
| 2 | Аналіз проблеми розробки та впровадження об`єкту проектування | до 01.03.2025 р. | Викон. |
| 3 | Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання | до 15.03.2025 р. | Викон. |
| 4 | Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних | до 29.03.2025 р. | Викон. |
| 5 | Практична реалізація об'єкта проектування та розробка бази даних | до 26.04.2025 р. | Викон. |
| 6 | Тестування та налагодження об`єкта проектування | до 03.05.2025 р. | Викон. |
| 7 | Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру | до 10.06.2025 р. | Викон. |

Здобувач вищої освіти _____ Олександр ГАПОНЮК

/Керівник кваліфікаційної роботи _____ Олександр ГОРДЄЄВ

# АНОТАЦІЯ

Гапонюк О. С. Розробка вебсервісу для керування замовленнями в ресторані з використанням React, Node.js та MySQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі проаналізовано існуючі системи керування замовленнями в ресторанному бізнесі, визначено їх переваги й недоліки, а також сформульовано мету і завдання проєкту. У другому розділі описано вимоги до програмного забезпечення, обґрунтовано вибір технологій, здійснено проєктування системи з використанням UML-діаграм та розроблено інтерфейс користувача. У третьому розділі наведено практичну реалізацію вебсервісу: створено фронтенд на React.js та бекенд на Node.js з використанням бази даних MySQL, реалізовано механізм реєстрації замовлень, авторизацію офіціанта, перегляд замовлень, оновлення їх статусу та збереження відгуків користувачів. Проведено тестування, налагодження та описано механізми захисту даних і документацію до системи.

У висновках підбито підсумки розробки та окреслено перспективи подальшого вдосконалення системи.

Ключові слова: вебсервіс, ресторан, замовлення, React, Node.js, MySQL, інтерфейс користувача, база даних**.**

# ABSTRACT

Gaponyuk O. Development of a Web Service for Restaurant Order Management Using React, Node.js and MySQL. Manuscript. Bachelor's qualification thesis of the educational program "Software Engineering", specialty 121 "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions and a list of references.

The first chapter analyzes existing restaurant order management systems, identifies their strengths and weaknesses, and formulates the project goals and objectives. The second chapter describes the software requirements, justifies the choice of technologies, presents the system design using UML diagrams, and outlines the user interface. The third chapter presents the practical implementation of the web service: the frontend was developed with React.js and the backend with Node.js using a MySQL database. The system supports order creation, waiter authentication, order status updates, and user feedback. Testing and debugging were conducted, along with data protection mechanisms and user documentation.

The conclusions summarize the development results and highlight prospects for further system improvement.

Keywords: web service, restaurant, orders, React, Node.js, MySQL, user interface, database.

# ЗМІСТ

# ВСТУП

Актуальність теми. Сучасний ресторанний бізнес є однією з найдинамічніших галузей економіки, що вимагає ефективного управління процесами для забезпечення високого рівня обслуговування клієнтів. Автоматизація керування замовленнями є важливим інструментом для оптимізації роботи закладів громадського харчування, підвищення продуктивності персоналу та мінімізації помилок у процесі обслуговування.

Сьогодні існує велика кількість систем керування замовленнями, однак багато з них мають певні обмеження у функціональності, адаптивності до специфічних потреб ресторанів та інтеграції з сучасними вебтехнологіями. Однією з основних проблем є застарілі інтерфейси, недостатня масштабованість і обмежені можливості кастомізації систем під конкретний заклад. Світові тенденції вказують на перехід до веборієнтованих рішень для керування замовленнями, які забезпечують доступ до системи з будь-якого пристрою через веббраузер. Використання технологій, таких як React для фронтенду, Node.js для бекенду та MySQL для роботи з базами даних, дозволяє створювати сучасні, ефективні та надійні програмні рішення.

Відовідно, актуальність дослідження полягає в необхідності створення універсальної вебплатформи для керування замовленнями, яка дозволяє швидко обробляти запити, оптимізувати робочий процес і забезпечити зручний інтерфейс для користувачів різних рівнів доступу (адміністратори, офіціанти, кухарі).

Метою роботи є розробка вебсервісу для керування замовленнями в ресторані, що забезпечить ефективну взаємодію між персоналом закладу та клієнтами, автоматизацію процесів прийому, обробки та виконання замовлень.

Завдання на кваліфікаційну роботу:

1) провести аналіз сучасних рішень у сфері автоматизації ресторанного обслуговування;

2) сформувати вимоги до вебзастосунку для керування замовленнями;

3) спроєктувати архітектуру системи та її базу даних;

4) реалізувати вебінтерфейс для клієнтів і персоналу за допомогою React.js;

5) створити серверну частину із застосуванням Node.js і MySQL;

6) забезпечити механізми авторизації, безпеки та захисту даних;

7) протестувати функціональність системи та усунути помилки;

8) підготувати користувацьку та технічну документацію.

Об’єктом є процес автоматизації керування замовленнями в закладах ресторанного бізнесу.

Предметом є вебсервіс, що використовує сучасні технології, такі як React, Node.js та MySQL, для ефективного керування замовленнями.

Галузь застосування розробленої системи охоплює ресторани, кафе, бари та інші заклади громадського харчування, які прагнуть автоматизувати процеси обслуговування та оптимізувати свою діяльність.

Взаємозв’язок з іншими роботами проявляється у використанні найкращих практик веброзробки, баз даних та інформаційної безпеки, що вже зарекомендували себе у схожих системах автоматизації.

Отже, розробка вебсервісу для керування замовленнями в ресторані є актуальним завданням, що відповідає сучасним вимогам до технологій управління бізнес-процесами та відкриває нові можливості для розвитку ресторанної індустрії.

# РОЗДІЛ 1
# АНАЛІЗ СИСТЕМ КЕРУВАННЯ ЗАМОВЛЕННЯМИ В РЕСТОРАННОМУ БІЗНЕСІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

## 1.1 Аналіз сучасного стану проблеми

Сучасний ресторанний бізнес є однією з найбільш конкурентоспроможних галузей, що вимагає ефективного управління всіма бізнес-процесами, включно із системою керування замовленнями.

Дослідження показують, що впровадження автоматизованих систем дозволяє не лише підвищити швидкість обслуговування клієнтів, але й оптимізувати роботу персоналу, мінімізувати помилки та збільшити прибуток закладу.

У наукових працях підкреслюється важливість інтеграції сучасних технологій, таких як веборієнтовані платформи, хмарні рішення та мобільні додатки, для ефективного управління ресторанним бізнесом. Зокрема, використання JavaScript-фреймворків, як-от React для фронтенду, Node.js для бекенду, та реляційних баз даних, таких як MySQL, забезпечує високу продуктивність, масштабованість та зручність використання систем.

Стаття «Сучасні технології управління в готельно-ресторанному бізнесі: практики та інновації» авторів Кащук К. М., Мосійчук І. В., Саух І. В. [1] досліджує новітні технології, що використовуються для підвищення ефективності управління готелями та ресторанами, а також для поліпшення обслуговування клієнтів і створення конкурентних переваг.

Автори виокремлюють основні тенденції розвитку готельного-ресторанного бізнесу та надають перелік провідних компаній, що пропонують сучасні технології управління. Вони підкреслюють важливість впровадження цифрових технологій, серед яких мобільні додатки для онлайн-бронювання, інструменти для автоматизації процесів і системи на основі штучного інтелекту. Доведено, що ці інструменти дозволяють підвищити ефективність роботи

закладів, мінімізувати помилки, покращити досвід користування та забезпечити оперативне управління ресурсами.

Окрему увагу приділено інноваційним рішенням, таким як робототехніка, яка забезпечує швидке та точне виконання завдань, а також інтеграція автоматизованих систем управління для скорочення часу на обслуговування клієнтів. У статті розглядаються приклади успішного впровадження цих технологій у практичній діяльності.

У статті Т. Лисюк «Інноваційні рішення в готельно-ресторанному бізнесі: технології автоматизації та персоналізації послуг» [2] досліджуються сучасні технологічні тренди, які трансформують цю сферу. Основна увага приділена впровадженню систем автоматизації, зокрема CRM-систем, чат-ботів і систем управління готельним бізнесом, які допомагають оптимізувати внутрішні процеси, ефективніше керувати ресурсами та скорочувати час обслуговування клієнтів. Розглядаються технології персоналізації послуг, що базуються на використанні Big Data та штучного інтелекту, завдяки яким можливо створювати індивідуальні пропозиції для клієнтів, від підбору номерів до складання персоналізованого меню.

Авторка аналізує реальні приклади успішного впровадження цих технологій у провідних готелях і ресторанах, підкреслюючи їхній внесок у підвищення рівня задоволеності клієнтів та лояльності.

Також розглядається роль мобільних додатків, систем безконтактного обслуговування та інтеграції IoT (інтернету речей), які сприяють підвищенню комфорту і зручності для гостей.

На ринку вже існує низка готових рішень для автоматизації керування замовленнями в ресторанах, серед яких:

1) Toast POS (рис. 1.1) – система управління замовленнями з можливістю аналітики та інтеграції з іншими платформами;

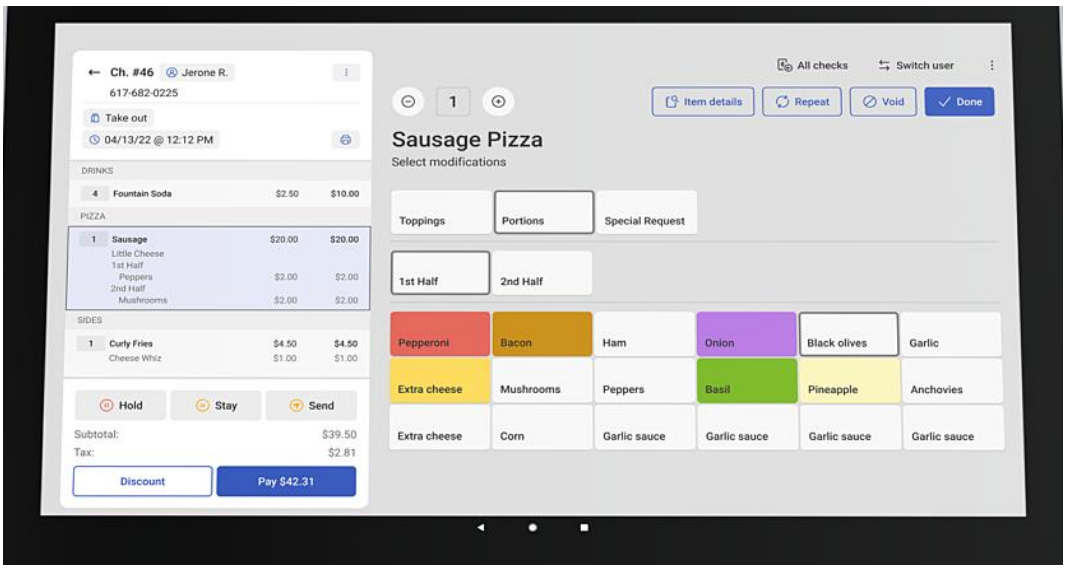


Рисунок 1.1 – Інтерфейс Toast POS [3]

2) Square for Restaurants (рис. 1.2) – мобільне рішення для керування замовленнями, розрахунками та аналітикою;

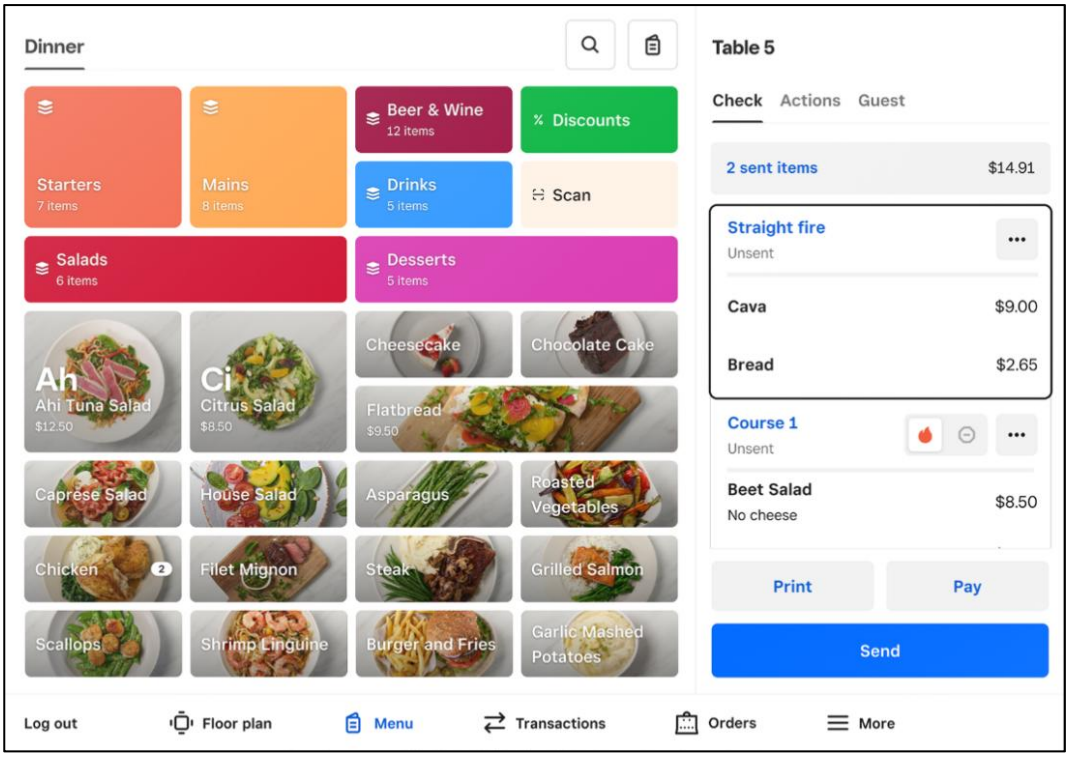


Рисунок 1.2 – Інтерфейс Square for Restaurants [4]

3) Lightspeed Restaurant – комплексна система для обслуговування клієнтів, ведення аналітики та управління персоналом.

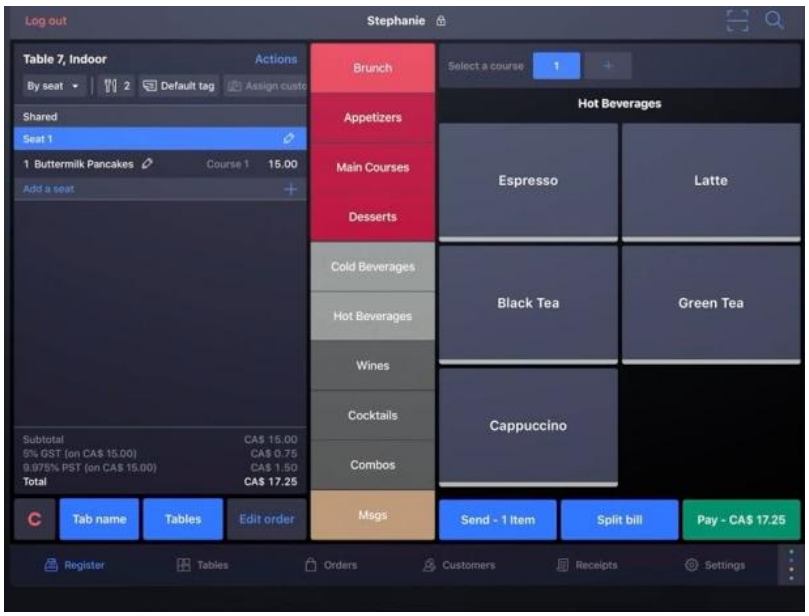


Рисунок 1.3 – Інтерфейс Lightspeed Restaurant [5]

Нижче наведена порівняльна таблиця аналогів систем керування замовленнями в ресторані (табл. 1.1).

Таблиця 1.1 – Порівняння аналогів

| Система | Toast POS | Square for Restaurants | Lightspeed Restaurant | Розроблювана система |
|---|---|---|---|---|
| Функціональність | Широкий набір функцій для управління замовленнями та аналітики | Основний функціонал для обслуговування замовлень | Розширена аналітика, підтримка інвентаризації | Гнучка система з можливістю кастомізації |
| Адаптивність | Середня | Висока | Середня | Висока |
| Вартість | Висока | Середня | Висока | Середня |
| Інтеграція | Обмежена | Висока | Середня | Висока |
| Підтримка мобільних пристроїв | Обмежена | Висока | Середня | Висока |
| Простота використання | Середня | Висока | Середня | Висока |
| Масштабованість | Обмежена | Висока | Середня | Висока |

Аналіз показує, що більшість існуючих рішень мають певні недоліки:

1) обмежені можливості налаштування під специфічні потреби ресторану;

2) висока вартість впровадження та обслуговування;

3) складність інтеграції з іншими інформаційними системами закладу;

4) недостатня підтримка мобільних пристроїв або неадаптивний інтерфейс.

Існуючі системи часто не забезпечують повноцінної автоматизації процесів керування замовленнями, що може призводити до збоїв у роботі, зниження продуктивності та втрати клієнтів. Недостатня інтеграція з сучасними вебтехнологіями обмежує можливості масштабування та розвитку бізнесу. Важливою прогалиною є також відсутність гнучкості в налаштуванні функціональності під унікальні потреби закладу.

Аналіз патентної бази показує, що існує ряд патентованих рішень для керування замовленнями у закладах громадського харчування. Проте більшість з них зосереджені на частковій автоматизації процесів або мають застарілу архітектуру. Запропонована система, побудована на базі React, Node.js та MySQL, відрізнятиметься сучасною архітектурою, високою продуктивністю, адаптивним дизайном та можливістю масштабування.

Розробка вебсервісу для керування замовленнями в ресторані є актуальною задачею, що відповідає сучасним викликам та вимогам ринку. Створення системи з використанням сучасних вебтехнологій дозволить вирішити існуючі проблеми, підвищити ефективність роботи закладів громадського харчування та забезпечити якісне обслуговування клієнтів.

Отже, проведений аналіз показав, що існуючі системи керування замовленнями мають як переваги, так і значні недоліки. Використання сучасних вебтехнологій у поєднанні з інтуїтивно зрозумілим інтерфейсом та гнучкою архітектурою дозволить створити конкурентоспроможний продукт, що відповідає вимогам сучасного ресторанного бізнесу.

## 1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою цієї кваліфікаційної роботи є створення сучасного вебсервісу для автоматизації процесів прийому, обробки та адміністрування замовлень у

ресторанному закладі з використанням вебтехнологій React, Node.js і MySQL. Для досягнення цієї мети необхідно виконати такі завдання:

– провести аналіз сучасних рішень у сфері автоматизації ресторанного обслуговування;

– сформувати вимоги до вебзастосунку для керування замовленнями;

– спроєктувати архітектуру системи та її базу даних;

– реалізувати вебінтерфейс для клієнтів і персоналу за допомогою React.js;

– створити серверну частину із застосуванням Node.js і MySQL;

– забезпечити механізми авторизації, безпеки та захисту даних;

– протестувати функціональність системи та усунути помилки;

– підготувати користувацьку та технічну документацію.

Загалом, для реалізації вебсервісу з керування замовленнями в ресторані було сформульовано низку завдань, що охоплюють як аналітичну, так і практичну частину проєкту. Визначено цілі розробки, функціональні та технічні вимоги до системи, обґрунтовано вибір інструментів та побудовано чітку логіку реалізації. Досягнення поставлених завдань дозволить створити ефективне, зручне у використанні рішення, здатне покращити обслуговування клієнтів та оптимізувати внутрішні бізнес-процеси ресторанного закладу.

## Висновки до розділу 1

У першому розділі було проведено детальний аналіз сучасного стану систем керування замовленнями в ресторанах, визначено основні проблеми та обґрунтовано доцільність створення нової системи. Було з’ясовано, що автоматизація управління замовленнями є важливим чинником для підвищення ефективності роботи ресторанів та забезпечення якісного обслуговування клієнтів. Аналіз існуючих рішень показав, що наявні системи мають обмежену адаптивність, складну інтеграцію з іншими інструментами та недостатню підтримку мобільних пристроїв.

Також було виявлено основні недоліки аналогів, серед яких відсутність можливості гнучкого налаштування під специфічні потреби закладів, висока вартість обслуговування та складність у масштабуванні. Обґрунтовано необхідність створення нового вебсервісу на основі сучасних технологій, таких як React, Node.js та MySQL, що дозволить забезпечити високу продуктивність, безпеку та масштабованість системи.

У підсумку було сформульовано завдання кваліфікаційної роботи, які охоплюють усі етапи проєктування та реалізації системи. Результати аналізу підтвердили необхідність створення нового вебсервісу для керування замовленнями в ресторанах, що відповідатиме сучасним вимогам до продуктивності, функціональності та безпеки. Отримані висновки слугують основою для подальшого проєктування та реалізації системи.

# РОЗДІЛ 2
# СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

## 2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

У сучасному ресторанному бізнесі ефективність обслуговування клієнтів значною мірою залежить від цифрових рішень для автоматизації процесу прийому, обробки й контролю замовлень. У межах кваліфікаційної роботи розробляється вебсервіс для керування замовленнями, орієнтований на оптимізацію взаємодії між офіціантами, кухнею та гостями, забезпечуючи гнучкий доступ до функцій системи через вебінтерфейс.

Для моделювання програмної системи обрано об’єктно-орієнтований підхід, що дозволяє структурувати систему у вигляді взаємодіючих сутностей – користувачів, замовлень, позицій меню тощо. Як формалізовану основу обрано нотацію UML (Unified Modeling Language), яка надає можливість наочно описати архітектуру, взаємозв’язки та функціонування системи [6]. Було створено наступні UML-діаграми:

1) діаграма варіантів використання (рис. 2.1);

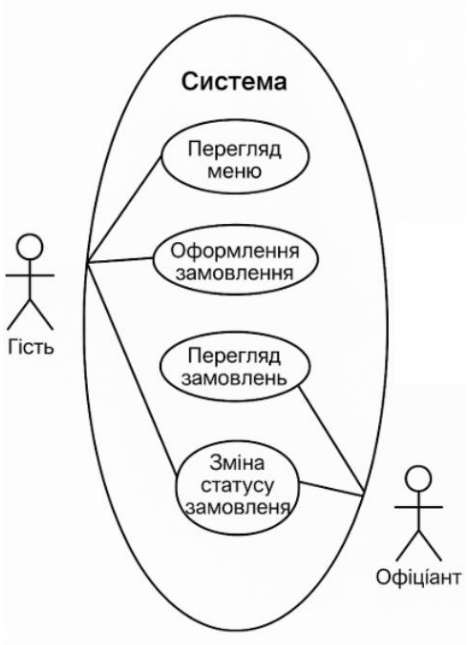


Рисунок 2.1 – Діаграма варіантів використання

2) діаграма класів (рис. 2.2);

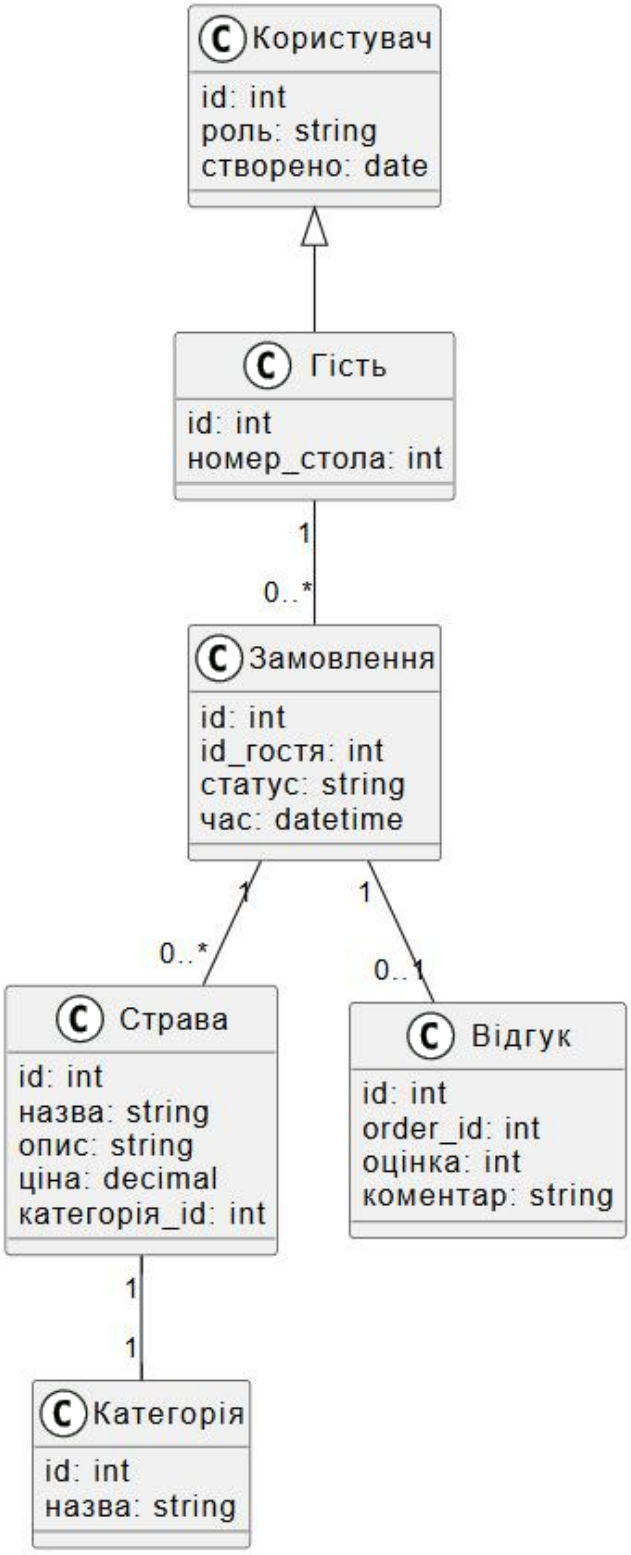


Рисунок 2.2 – Діаграма класів

3) діаграма послідовності (рис. 2.3).

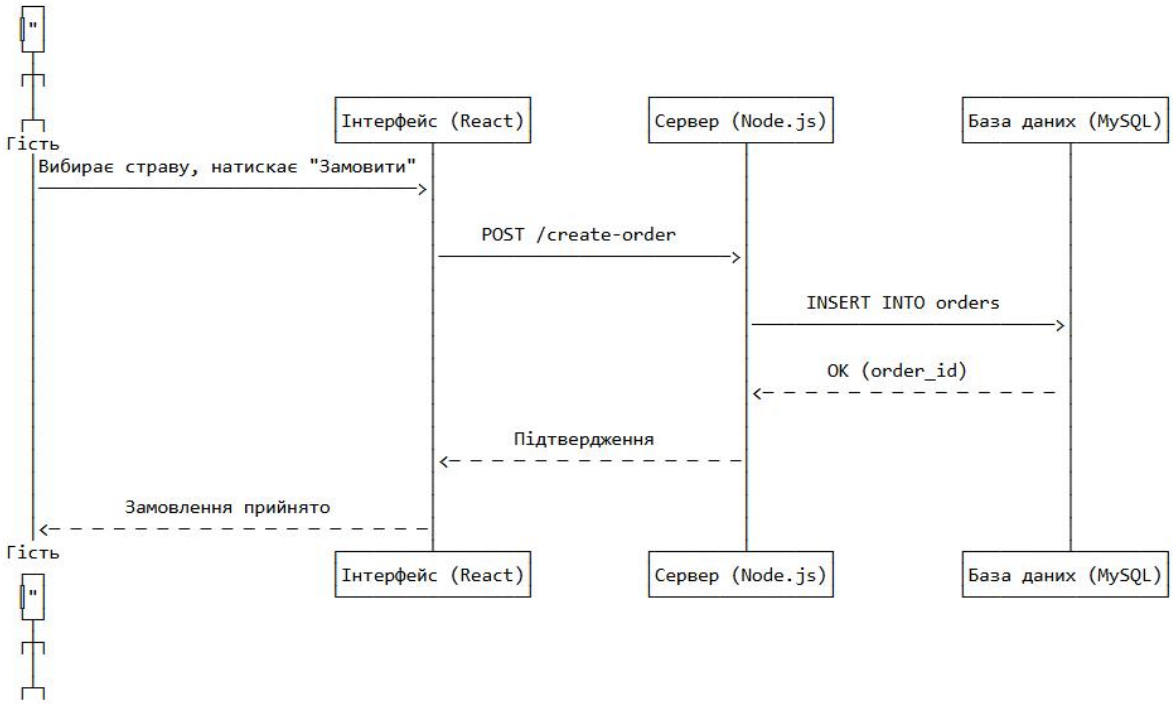


Рисунок 2.3 – Діаграма послідовності

Для аналізу вимог до системи застосовано методи:

1) опитування цільових користувачів (офіціантів і адміністратора);
2) аналіз аналогів (GloriaFood, Poster POS, Square for Restaurants);
3) аналіз типових сценаріїв роботи ресторану.

Виявлені основні ролі користувачів:

1) офіціант – створює замовлення, переглядає їхній статус;
2) гість – створює замовлення та переглядає їхній статус.

Функціональні вимоги:

1) реєстрація й авторизація офіціантів та гостей;
2) перегляд меню з фільтрацією й пошуком;
3) створення замовлень із кількістю, коментарем, часом;
4) відображення статусу замовлення («Отримано», «Виконується», «Виконано», «Оплачено»);
5) перегляд замовлень гостем у реальному часі;
6) оцінювання замовлень (1–5 зірок) після оплати;
7) розмежування прав доступу (офіціант / гість / адміністратор).

Нефункціональні вимоги:

1) кросбраузерність (Chrome, Firefox, Edge);
2) захищене зберігання даних (авторизація за JWT, HTTPS);
3) резервування замовлень у БД MySQL;
4) висока швидкодія (відповідь API не більше 300 мс при 1000 запитах).

Інтерфейс вебсервісу буде реалізовано з використанням React.js і CSS-модулів, що забезпечить гнучкість стилізації та зручність підтримки. Дизайн планується виконати у мінімалістичному стилі з адаптивною верстkою для коректного відображення на різних пристроях. Для гостей буде передбачено сторінку з меню у вигляді плиток зі стравами, де можна обирати позиції й формувати замовлення. Також буде реалізована сторінка перегляду власних замовлень із відображенням статусу виконання, часу й можливістю залишити відгук. Офіціант матиме окрему сторінку з таблицею активних замовлень та можливістю оновлення їхнього статусу. У шапці інтерфейсу будуть кнопки навігації (меню, замовлення, вихід), а футер міститиме контактну інформацію та посилання, адаптовані під мобільні пристрої.

Інформаційні потоки організовано через RESTful API, з наступними основними маршрутами:

1) POST /add-order – створення замовлення;
2) GET /orders – отримання замовлень для офіціанта/адміністратора;
3) GET /guest-orders – отримання замовлень лише для поточного гостя;
4) PUT /orders/:id/status – зміна статусу;
5) POST /add-review – додавання оцінки.

У результаті аналізу визначено основні вимоги до системи, обрано оптимальну архітектуру та засоби реалізації. Використання сучасних фреймворків React і Node.js забезпечує гнучкість розробки та масштабованість сервісу. Створена специфікація лягла в основу реалізації системи та її тестування.

## 2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для реалізації вебсервісу керування замовленнями в ресторані необхідно обрати набір технологій, які забезпечать масштабованість, швидкодію, безпеку та зручність користування. На основі аналізу функціональних потреб системи, сучасних тенденцій у веброзробці та практичних вимог ресторанного бізнесу, було здійснено порівняльний аналіз інструментів і обрано оптимальний стек технологій: React, Node.js, MySQL, а також допоміжні бібліотеки й фреймворки для підтримки фронтенду, бекенду та бази даних. Аналіз засобів розробки наведено в таблиці 2.2.

Таблиця 2.2 – Аналіз та обґрунтування вибору засобів розробки

| Засіб / Технологія | Призначення | Переваги | Недоліки |
|---|---|---|---|
| React [7] | Побудова динамічного інтерфейсу користувача | Компонентна архітектура, віртуальний DOM, висока продуктивність, велика спільнота | Потребує глибших знань JavaScript, іноді надлишковий для простих проєктів |
| Node.js [8] | Серверна частина (backend) | Асинхронна обробка, ефективність при великій кількості запитів, єдина мова розробки (JavaScript) | Вимагає ретельного контролю обробки помилок |
| MySQL [9] | Реляційна СУБД для збереження даних | Висока швидкість, підтримка транзакцій, масштабованість, велика кількість інструментів адміністрування | Не підтримує повністю об’єктно-орієнтовану структуру |
| Express.js [10] | Фреймворк для Node.js | Швидке створення REST API, middleware-підхід, простота інтеграції | Не має суворої структури – потрібно встановлювати окремо модулі безпеки |
| JWT (JSON Web Token) | Аутентифікація | Безпечна передача даних між клієнтом і сервером | Потребує правильного налаштування терміну дії токена |

У процесі реалізації вебсервісу для керування замовленнями в ресторані було обрано сучасні, перевірені часом технології, що забезпечують надійність, продуктивність та безпеку системи. React використовується для створення інтерфейсу користувача за концепцією SPA (Single Page Application), що

дозволяє динамічно оновлювати вміст сторінки без її повного перезавантаження. Це значно покращує взаємодію з користувачем та підвищує швидкодію вебдодатку. Node.js у поєднанні з Express.js реалізує серверну частину та обробляє всі RESTful-запити від клієнтської частини. Express забезпечує просту організацію маршрутизації, обробку запитів, підключення middleware та взаємодію з базою даних.

Для зберігання структурованої інформації про замовлення, користувачів і меню застосовується MySQL – реляційна система керування базами даних, яка підтримує транзакції, зовнішні ключі та забезпечує високу продуктивність при роботі з великими обсягами даних. JWT (JSON Web Token) використовується для реалізації механізму авторизації та автентифікації користувачів із розмежуванням прав доступу. Кожен авторизований користувач отримує токен, який додається до запитів і дозволяє серверу ідентифікувати користувача без повторного введення даних. BCrypt застосовується для безпечного зберігання паролів. Паролі хешуються за допомогою цього алгоритму, що унеможливлює їх розшифрування у випадку витоку бази даних.

Крім основних технологій, додатково використовуються такі бібліотеки як dotenv (для роботи зі змінними середовища), helmet (для базового захисту HTTP-заголовків) і cors (для налаштування доступу до ресурсів API з фронтенду). Їх використання сприяє покращенню захищеності, стабільності та гнучкості системи.

У системі реалізовано такі алгоритми.

1. Алгоритм створення замовлення:
   – перевірка авторизації користувача;
   – валідація введених даних;
   – запис замовлення у базу даних;
   – виведення підтвердження.
2. Алгоритм зміни статусу замовлення («офіціант/кухар» → «готове»):
   – пошук замовлення за ID;
   – зміна статусу у таблиці orders;

– генерація повідомлення frontend-клієнту.

3. Алгоритм пошуку замовлень за критеріями:

– отримання параметрів запиту;

– формування динамічного SQL-запиту;

– повернення результатів через REST API.

Блок-схема алгоритму створення замовлення наведена на рисунку 2.4.

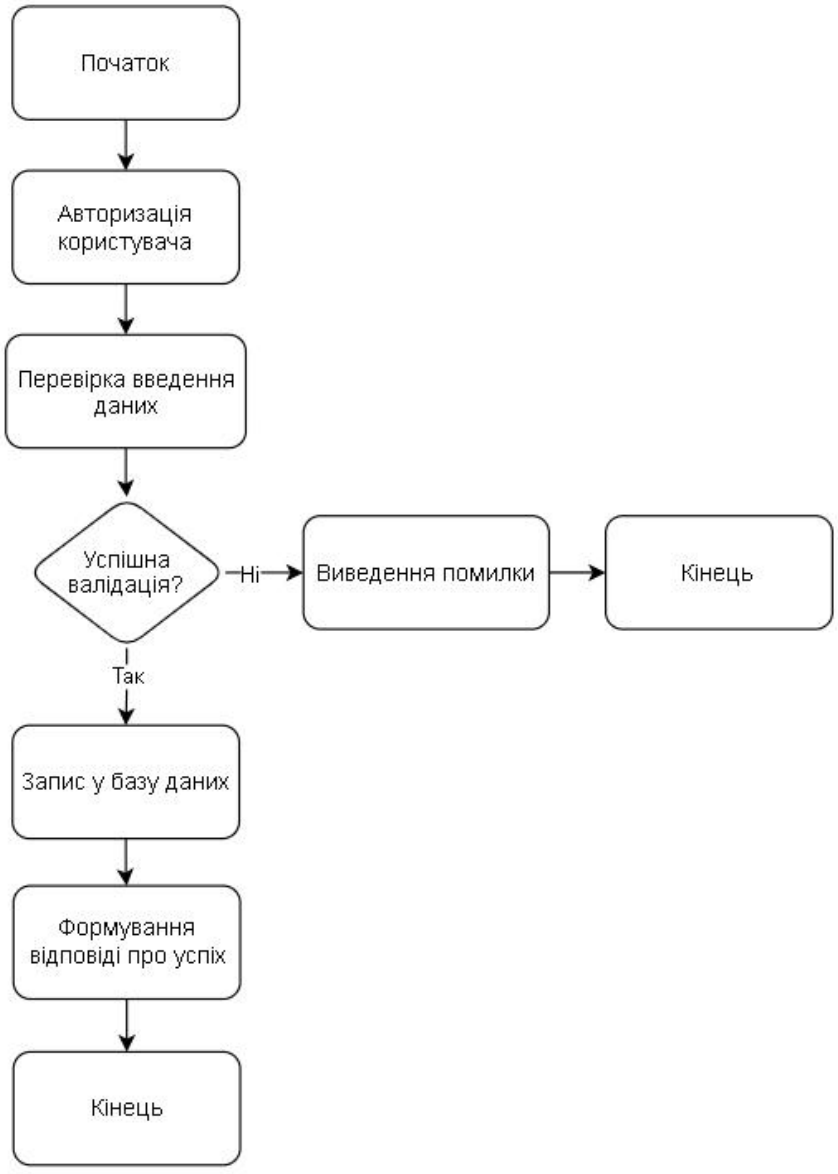


Рисунок 2.4 – Блок-схема алгоритму створення замовлення

Ця блок-схема описує послідовність дій, що виконуються під час створення замовлення у вебсервісі. Процес починається з авторизації

користувача, після чого здійснюється перевірка введених даних. Якщо валідація даних проходить успішно, інформація записується до бази даних і формується відповідь про успішне виконання операції. У разі невдалої валідації система виводить повідомлення про помилку, після чого процес завершено. Дана схема чітко демонструє логіку обробки вхідного запиту та реалізує один із основних сценаріїв роботи системи – додавання нового замовлення.

Діаграма компонентів системи наведена на рисунку 2.5.

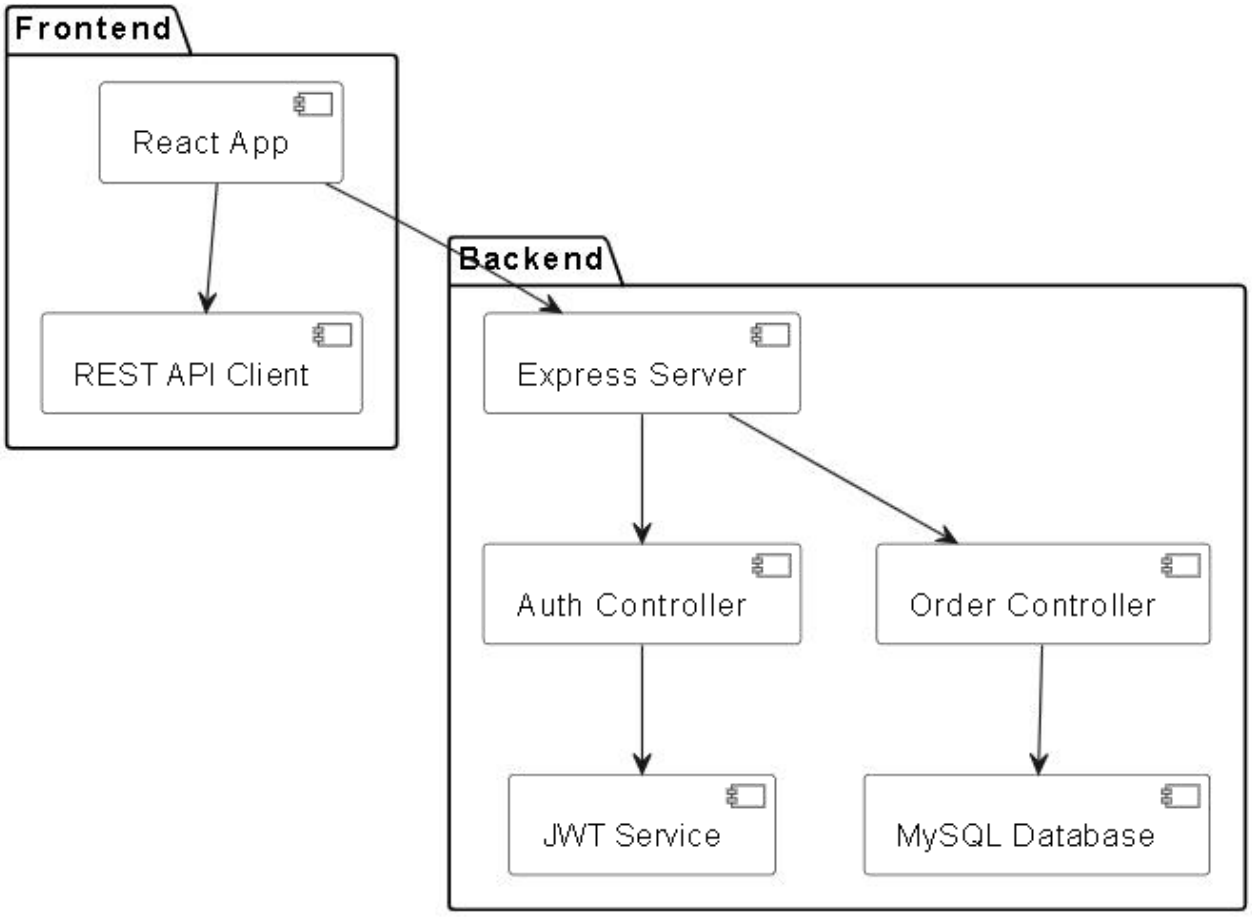


Рисунок 2.5 – Діаграма компонентів системи

Ця діаграма компонентів ілюструє архітектуру взаємодії між клієнтською та серверною частинами вебсервісу. У компоненті Frontend міститься додаток, реалізований на React, який взаємодіє з REST API-клієнтом. У Backend входить сервер Express, що має два контролери: Auth Controller для автентифікації користувачів і Order Controller для обробки замовлень. Auth Controller взаємодіє з JWT Service для генерації токенів доступу, тоді як Order Controller звертається

до бази даних MySQL. Така структура забезпечує чіткий розподіл функцій між модулями системи.

Діаграма розгортання наведена на рисунку 2.6. На діаграмі розгортання показано фізичну архітектуру програмної системи. Вебінтерфейс, реалізований у браузері за допомогою React, надсилає запити до REST API, реалізованого через Express.js на сервері Node.js. Серверна частина містить модулі Auth Module і Order Module, що обробляють відповідні запити користувачів. Вся інформація зберігається на окремому сервері бази даних з MySQL. Така структура дозволяє забезпечити масштабованість, централізоване зберігання даних та безпечну взаємодію між клієнтом і сервером.

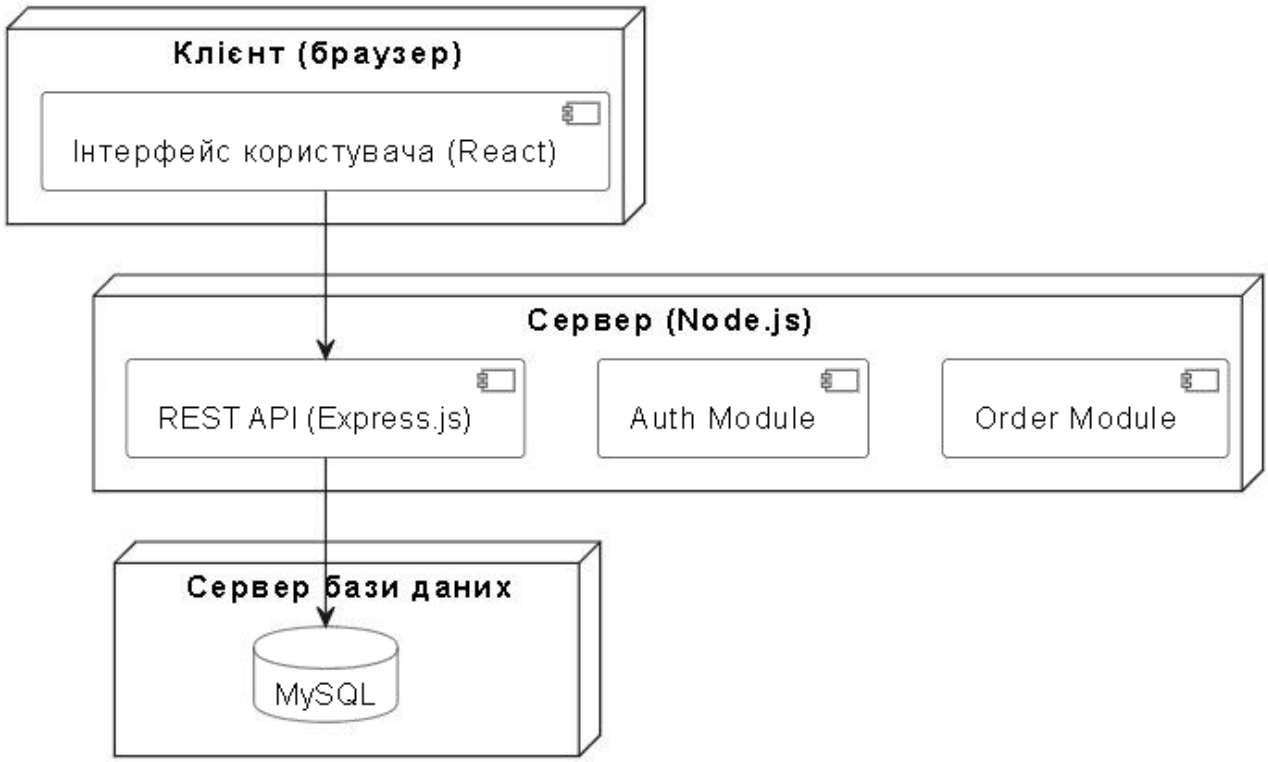


Рисунок 2.6 – Діаграма розгортання системи

В результаті аналізу обрано найбільш ефективні інструменти для реалізації вебсервісу управління замовленнями, що забезпечують швидку розробку, масштабованість, безпеку та зручний інтерфейс користувача. Запропоновані алгоритми, моделі даних і архітектурні рішення дозволяють

ефективно реалізувати прикладні задачі системи та забезпечити її працездатність у реальному середовищі.

## Висновки до розділу 2

У другому розділі було здійснено детальний аналіз вимог до розроблюваного вебсервісу для керування замовленнями в ресторані та обґрунтовано вибір технологічних засобів, моделей і методів його реалізації. Було визначено функціональні та нефункціональні вимоги до системи, сформульовано основні сценарії її використання, що враховують особливості роботи різних типів користувачів (офіціанта, кухаря, адміністратора).

Застосування UML-діаграм дозволило формалізувати логіку взаємодії користувачів із системою, структуру програмних компонентів, а також зв’язки між об’єктами даних. На основі обраної ітераційної моделі розробки визначено доцільність покрокової реалізації функціоналу з урахуванням перевірки вимог та гнучкого реагування на зміни.

Обґрунтовано вибір сучасного стеку технологій – React, Node.js, Express.js, MySQL, JWT – які забезпечують високу продуктивність, адаптивність, масштабованість та безпеку вебсервісу. Проведено порівняльний аналіз альтернативних інструментів і обґрунтовано ефективність обраного технічного рішення для реалізації задач предметної області. Описано алгоритми обробки замовлень, їх валідації, зміни статусу та пошуку, що є основними для забезпечення функціонування системи. Блок-схеми та діаграми розгортання і компонентів чітко демонструють внутрішню архітектуру та принципи побудови взаємодії між модулями.

Загалом, виконані в цьому розділі роботи створили міцне підґрунтя для практичної реалізації системи, що буде представлена у наступному розділі, та підтвердили доцільність обраного підходу до її проєктування.

# РОЗДІЛ 3
# РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ КЕРУВАННЯ ЗАМОВЛЕННЯМИ В РЕСТОРАНІ

## 3.1 Практична реалізація об’єкта проєктування

Розробка програмного забезпечення здійснювалася з використанням стеку технологій React.js для клієнтської частини, Node.js (Express.js) для серверної логіки та MySQL як системи управління базами даних. Розробку реалізовано в середовищах Visual Studio Code для фронтенду та бекенду, phpMyAdmin для управління базою даних, а також Postman для тестування API.

На етапі реалізації клієнтської частини було створено компоненти на React, які відповідають за відображення сторінки меню, авторизацію, перегляд замовлень та відправлення відгуків. Наприклад, компонент відображення меню для гостей реалізований з використанням хуків useState та useEffect (ліст. 3.1).

Ліситнг 3.1 – Хук useEffect

```
useEffect(() => {
  axios.get("http://localhost:8081/menu-items").then((res) => {
    setMenu(res.data);
  });
}, []);
```

кінець лістингу 3.1

Для зберігання стану введених даних (наприклад, кількості обраних порцій) застосовуються керовані компоненти (ліст. 3.2).

Ліситнг 3.2 – Керовані компоненти

```
const [quantities, setQuantities] = useState({});

const handleQuantityChange = (item_id, value) => {
  setQuantities((prev) => ({ ...prev, [item_id]: value }));
};
```

кінець лістингу 3.2

Користувач (гість) може створити замовлення через функцію handleOrder, яка відправляє запит до сервера (ліст. 3.3).

Ліситнг 3.3 – Створення замовлення

```
axios.post("http://localhost:8081/add-order", {
  itemId: menuItem.item_id,
  guestId,
  quantity,
  orderTime: new Date().toISOString(),
  status: "Отримано",
  comment: comment || "Коментар відсутній"
});
```

кінець лістингу 3.3

На серверній частині використано фреймворк Express.js. Вхідні запити обробляються через відповідні маршрути. Наприклад, додавання замовлення (ліст. 3.4).

Ліситнг 3.4 – Маршрут додавання замовлення

```
app.post("/add-order", verifyUser, (req, res) => {
  const { itemId, guestId, quantity, orderTime, status, comment } =
req.body;
  const sql = `
    INSERT INTO orders (item_id, guest_id, quantity, order_time, status,
comment)
    VALUES (?, ?, ?, ?, ?, ?)
  `;
  db.query(sql, [itemId, guestId, quantity, orderTime, status, comment],
(err) => {
    if (err) return res.status(500).json({ error: "Database error" });
    return res.json({ status: "Success" });
  });
});
```

кінець лістингу 3.4

Аутентифікація реалізована за допомогою JWT-токенів, які зберігаються у cookies (ліст. 3.5).

Ліситнг 3.5 – Токен аутентифікації

```
const token = jwt.sign({ id, username }, "jwt-secret-key", { expiresIn: "1d" });
res.cookie("token", token, { httpOnly: true });
```

кінець лістингу 3.5

Для збереження відгуків створено окремий маршрут /add-review, який записує оцінку та текст коментаря до таблиці reviews. Інтерфейс користувача адаптовано до мобільних пристроїв за допомогою медіазапитів і CSS-модулів. Кнопки навігації дозволяють користувачеві зручно переходити між меню та списком замовлень, які фільтруються та сортуються за статусом і часом. Усі дані передаються по протоколу HTTP через RESTful API, що дозволяє легко масштабувати або модифікувати систему.

Загалом, практична реалізація доводить ефективне використання інструментів сучасної веб-розробки для вирішення прикладної задачі автоматизації процесу обслуговування клієнтів у ресторані.

## 3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування вебсервісу для керування замовленнями в ресторані проводилося на етапах розробки клієнтської та серверної частин з метою виявлення логічних помилок, перевірки коректності обробки даних та забезпечення стабільної роботи функціоналу. Було застосовано такі методи тестування: модульне тестування, інтеграційне тестування, функціональне тестування та ручне тестування інтерфейсу.

На етапі модульного тестування на сервері (Node.js) тестуванню підлягали маршрути API: /login, /login-user, /add-order, /orders, /add-review. Наприклад, тестувалося коректне створення замовлення при вірно надісланих параметрах (ліст. 3.6).

Лістинг 3.6 – Створення замовлення при вірно надісланих параметрах

```
POST /add-order
BODY: { itemId: 1, guestId: 2, quantity: 1, orderTime: ..., status: "Отримано", comment: "без цибулі" }
```

кінець лістингу 3.6

У випадку успішної обробки запиту сервер повертає JSON-об’єкт { status: "Success" }. У разі помилки – відповідний код помилки та повідомлення.

Інтеграційне тестування – було перевірено взаємодію між компонентами: авторизація гостя → створення замовлення → зміна статусу офіціантом → можливість залишити відгук. Особливу увагу приділено збереженню послідовності дій і правильному розмежуванню доступу.

Функціональне тестування – проводилося на всіх основних сценаріях використання системи:

1) успішна авторизація офіціанта та користувача;
2) формування замовлення;
3) перегляд власних замовлень;
4) оновлення статусу замовлення;
5) надсилання відгуку.

Також перевірено функціональність пошуку по меню та фільтрації замовлень за статусом.

Ручне тестування інтерфейсу – UI перевірявся на відповідність очікуваному вигляду, коректність відображення компонентів, адаптивність на екранах різного розміру. Виявлені помилки в стилях і логіці (наприклад, неправильне відображення модального вікна після неуспішної авторизації) були усунені шляхом оновлення CSS і виправлення умов рендерингу компонентів.

Для роботи вебсервісу необхідні наступні параметри. Клієнт (браузер):

1) сучасний браузер (Google Chrome, Firefox, Edge);
2) підтримка JavaScript ES6+;
3) мінімальна роздільна здатність екрана – 360×640 px.

Сервер:

1) Node.js 16 або вище;
2) MySQL 5.7+;
3) ОС: Windows/Linux;
4) оперативна пам’ять: від 1 ГБ;
5) вільне місце на диску: від 50 МБ.

У процесі тестування було виявлено кілька типових помилок. Зокрема, система некоректно зберігала замовлення у випадках, коли не було вказано кількість страв. Також фіксувалося дублювання записів, що виникало при повторному натисканні на кнопку «Замовити» без відповідної перевірки. В окремих випадках спостерігалися проблеми з авторизацією, спричинені відсутністю токена. Крім того, не відображалося повідомлення про успішне надсилання або обробку дій, що створювало плутанину у взаємодії користувача з інтерфейсом.

Усі виявлені помилки були виправлені, що підтверджується результатами повторного тестування. На поточному етапі система виконує всі поставлені функціональні вимоги, демонструючи стабільну роботу як на етапі створення замовлень, так і на етапах взаємодії між користувачем і персоналом ресторану.

### 3.3 Розробка бази даних

У процесі реалізації вебсервісу для керування замовленнями в ресторані було створено реляційну базу даних restaurant, яка обслуговує як функціональні потреби користувачів (гостей та офіціантів), так і внутрішню логіку обробки замовлень та відгуків. Для створення бази даних використовувалася СКБД MySQL, а для її візуального редагування та керування – інструмент phpMyAdmin.

У базі даних реалізовано сім взаємопов’язаних таблиць:

1) tables – зберігає номери столів ресторану;

2) guests – зберігає інформацію про відвідувачів, пов’язану з конкретним столом;

3) waiters – облікові записи персоналу;

4) menu – перелік доступних страв;

5) categories – категорії меню (салати, супи тощо);

6) orders – замовлення користувачів з деталями;

7) reviews – відгуки на виконані та оплачені замовлення.

Зв’язки між таблицями реалізовано через FOREIGN KEY для забезпечення нормалізації та логічної цілісності структури. Наприклад, поле category_id у таблиці menu є зовнішнім ключем до таблиці categories. На рисунку 3.1 наведено діаграму бази даних.

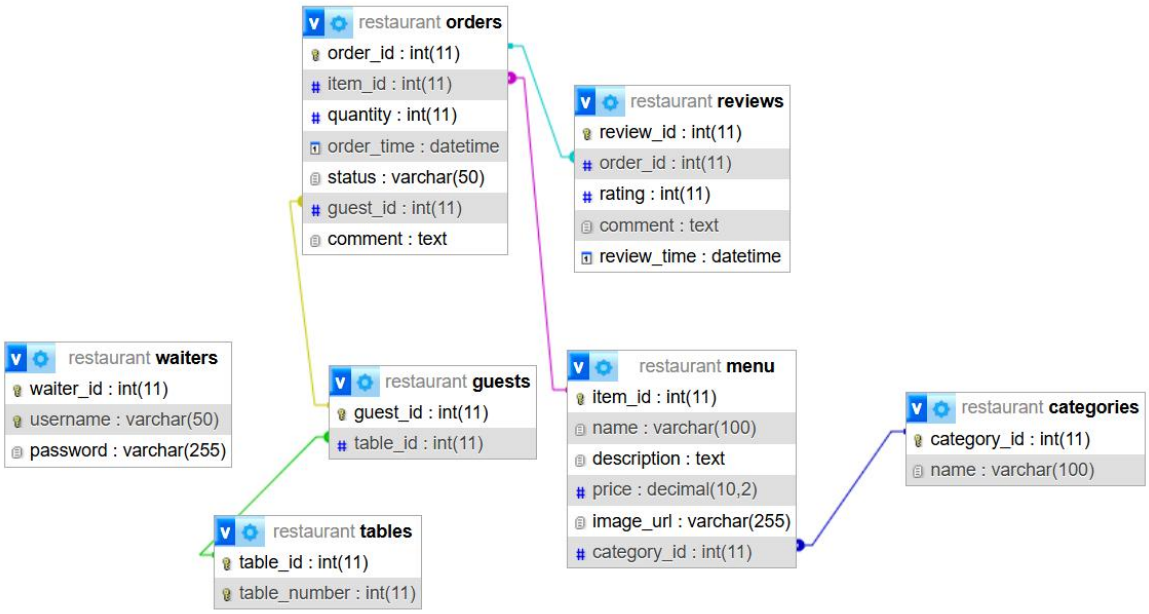


Рисунок 3.1 – Діаграма бази даних

Наведемо також приклади SQL-запитів. Створення таблиці замовлень наведено в лістингу 3.7.

Лістинг 3.7 – Створення таблиці замовлень

```
CREATE TABLE orders (
  order_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  item_id INT,
  guest_id INT,
  quantity INT,
  order_time DATETIME,
  status VARCHAR(50),
  comment TEXT,
  FOREIGN KEY (item_id) REFERENCES menu(item_id),
  FOREIGN KEY (guest_id) REFERENCES guests(guest_id)
);
```

кінець лістингу 3.7

Вибір усіх замовлень з іменем страви наведено в лістингу 3.8.

Лістинг 3.8 – Вибір замовлень з іменем страви

```
SELECT o.order_id, m.name AS item_name, o.quantity, o.status, o.order_time
FROM orders o
JOIN menu m ON o.item_id = m.item_id;
```

кінець лістингу 3.8

Отримання меню за категорією наведено в лістингу 3.9.

Лістинг 3.9 – Отримання меню за категорією

```
SELECT m.name, m.price, c.name AS category
FROM menu m
JOIN categories c ON m.category_id = c.category_id
WHERE c.name = 'Салати';
```

кінець лістингу 3.9

Додавання нового гостя наведено в лістингу 3.10, як зразок запиту на вставку даних.

Лістинг 3.10 – Додавання нового гостя

```
INSERT INTO guests (table_id) VALUES (3);
```

кінець лістингу 3.10

Отримання замовлень конкретного гостя наведено в лістингу 3.11.

Лістинг 3.11 – Отримання замовлень конкретного гостя

```
SELECT * FROM orders WHERE guest_id = 5;
```

кінець лістингу 3.11

Розроблена база даних є основою функціонування всієї інформаційної системи. Вона забезпечує збереження, оновлення, обробку та перегляд даних у реальному часі. Структура була спроектована відповідно до принципів третьої нормальної форми (3НФ), що дозволяє уникати надлишковості, забезпечити масштабованість і легке розширення системи в майбутньому.

### 3.4 Захист інформаційно-комп'ютерної системи

У процесі розробки вебсервісу для керування замовленнями в ресторані були враховані основні аспекти безпеки, що забезпечують захист даних користувачів, а також надійність та цілісність інформаційних потоків системи.

Основним механізмом автентифікації в системі є перевірка даних користувача (офіціанта) під час входу до системи. Авторизація реалізована з використанням токенів сесії, які передаються у вигляді httpOnly cookies, що значно знижує ризик викрадення даних через скриптові атаки (наприклад, XSS). При вході користувача генерується унікальний токен, який зберігається тільки на сервері та клієнт не має до нього прямого доступу через JavaScript.

Паролі користувачів у базі даних зберігаються у вигляді хешованих значень з використанням алгоритму bcrypt, що є стійким до атак типу brute-force. Bcrypt використовує соління (salt), що унеможливлює застосування словникових атак або простого порівняння хешів.

Передача даних між клієнтом і сервером захищена за допомогою протоколу HTTPS (при розгортанні на хостингу), що гарантує шифрування трафіку і запобігає його перехопленню. Це особливо важливо при передачі даних авторизації та замовлень.

Захист від CSRF-атак (Cross-Site Request Forgery) реалізовано через використання токенів сесії, а також налаштування CORS на стороні сервера. Сервер приймає лише запити з дозволених доменів та перевіряє заголовки запиту.

Також було реалізовано захист від дублювання запитів (наприклад, багаторазового натискання кнопки «Замовити») за допомогою frontend- і backend-фільтрації на рівні логіки та бази даних.

Щодо захисту від DDoS-атак, застосовано обмеження частоти запитів на сервер за допомогою middleware express-rate-limit, що дозволяє уникнути перевантаження сервера шляхом блокування надмірної кількості запитів із одного IP.

Усі перераховані засоби створюють багаторівневу систему безпеки, яка забезпечує стійкість інформаційно-комп'ютерної системи до найпоширеніших типів атак та несанкціонованого доступу.

### 3.5 Розробка документації для програмного продукту

Для повноцінного функціонування розробленого вебсервісу необхідні такі мінімальні системні вимоги: операційна система Windows 10 або Linux Ubuntu 20.04+, процесор із частотою від 2.0 ГГц, 4 ГБ оперативної пам'яті, 1 ГБ вільного простору на диску, встановлені Node.js версії 18+ і MySQL Server 8.0+. Для фронтенд-розробки додатково потрібен менеджер пакетів npm.

Процедура розгортання вебдодатку включає кілька етапів. Спочатку необхідно встановити всі залежності для фронтенду та бекенду за допомогою команд npm install у відповідних директоріях. Далі слід налаштувати середовище для серверної частини, вказавши у .env файл параметри з'єднання з базою даних: ім'я хоста, користувача, пароль, порт і назву БД. Після цього виконується запуск серверної частини за допомогою npm start або nodemon.

Базу даних необхідно створити за заздалегідь підготовленим SQL-дампом, який включає таблиці guests, orders, menu_items, reviews тощо. Після

налаштування з’єднання між сервером і БД забезпечується коректна обробка запитів до API.

Фронтенд частина запускається окремо командою npm start у React-директорії. Для налаштування проксі-сервера до backend API використовується конфігурація у package.json.

Документація користувача охоплює такі аспекти:

1) авторизація офіціанта через спеціальну форму з перевіркою ролі;
2) оформлення замовлення гостем із вибором страв, кількості й підтвердженням;
3) перегляд замовлень та їх статусів у розділі «Мої замовлення»;
4) залишення відгуку після виконання та оплати замовлення;
5) оновлення статусів офіціантом у зручному інтерфейсі.

Довідкова інформація подається через інтерфейсні підказки (placeholder’и, повідомлення, кнопки підтвердження) та адаптований UX-дизайн, що полегшує взаємодію з системою. У майбутньому доцільно інтегрувати повноцінний Help-розділ або FAQ з найпоширенішими ситуаціями.

Загалом, розроблена документація дає змогу як швидко розгорнути систему, так і легко користуватись її функціоналом як працівникам, так і гостям закладу.

## Висновки до розділу 3

У третьому розділі було безпосередньо реалізовано програмне забезпечення відповідно до визначених вимог та проєктних рішень. Було створено повнофункціональний вебсервіс для керування замовленнями в ресторані, який дозволяє гостям переглядати меню, формувати замовлення, відстежувати їх статус та залишати відгуки, а офіціантам – адмініструвати замовлення через спеціальний інтерфейс.

Серверна частина вебсервісу реалізована на платформі Node.js з використанням Express.js, що забезпечує швидку обробку HTTP-запитів та

підключення до бази даних MySQL. База даних спроєктована відповідно до принципів нормалізації та включає всі необхідні сутності для зберігання інформації про гостей, страви, замовлення та відгуки.

Фронтенд-інтерфейс реалізовано з використанням React.js із підтримкою адаптивної верстки, що дозволяє забезпечити комфортну роботу на пристроях із різними розмірами екранів. Застосовано сучасні принципи UX/UI-дизайну: інтуїтивна навігація, модульна структура, відображення статусів у реальному часі.

У ході тестування системи виявлено та усунуто ряд типових помилок, що дозволило значно підвищити стабільність та зручність роботи з додатком. Також впроваджено базові механізми захисту – зокрема, контроль сесій, обмеження прав доступу до ресурсів, валідація вхідних даних.

Окрім програмної реалізації, підготовлено супровідну документацію з описом процесу встановлення, налаштування та використання розробленого сервісу. У результаті виконаної роботи створено ефективний інструмент для цифровізації ресторанного обслуговування, який може бути масштабований та доповнений у майбутньому відповідно до потреб бізнесу.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнуто всі поставлені цілі та повністю реалізовано функціональні можливості розроблюваного вебсервісу для керування замовленнями в ресторані.

Було проведено всебічний аналіз існуючих систем автоматизації ресторанного обслуговування, таких як Toast POS, Square for Restaurants та Lightspeed. Виявлено їхні ключові функції, а також обмеження – високу вартість, складність інтеграції та низьку адаптивність. Це дозволило чітко сформувати бачення майбутнього продукту, який має бути гнучким, масштабованим та зручним для користувача.

У ході дослідження було сформульовано як функціональні (створення замовлень, перегляд меню, авторизація, оновлення статусів), так і нефункціональні вимоги (висока продуктивність, безпека, кросбраузерність). Було визначено ролі користувачів (гість, офіціант, адміністратор) та сценарії їх взаємодії з системою, що дало змогу забезпечити чітке технічне планування розробки.

На основі аналізу вимог було спроєктовано архітектуру вебзастосунку з чітким поділом на клієнтську та серверну частини. Для формалізації архітектури створено UML-діаграми: варіантів використання, класів, послідовності та компонентів. Також розроблено реляційну базу даних MySQL із сімома взаємозв’язаними таблицями, яка повністю забезпечує логіку роботи сервісу.

Було створено адаптивний інтерфейс користувача на базі React.js, який забезпечує зручну взаємодію з системою як для гостей, так і для офіціантів. Реалізовано компоненти для перегляду меню, створення замовлень, перегляду їх статусів та відправлення відгуків. Особливу увагу приділено UX/UI-дизайну та адаптації для мобільних пристроїв.

Серверну логіку реалізовано з використанням Node.js та Express.js, що забезпечило обробку всіх REST API-запитів клієнтської частини. Сервер

взаємодіє з базою даних MySQL для збереження інформації про замовлення, користувачів і меню. Система підтримує створення, оновлення, пошук замовлень та управління ними з боку персоналу.

У реалізованій системі передбачено багаторівневий захист: автентифікація через JWT, зберігання токенів у httpOnly cookies, хешування паролів за допомогою bcrypt, CORS-контроль, фільтрація запитів, обмеження частоти запитів через middleware. Це дозволило захистити дані користувачів від несанкціонованого доступу та найпоширеніших типів атак.

Було проведено модульне, інтеграційне, функціональне та ручне тестування всіх ключових сценаріїв роботи системи. Виявлені помилки, такі як дублювання замовлень чи некоректна авторизація, були усунені. У результаті тестування підтверджено стабільну та надійну роботу системи відповідно до всіх заявлених функціональних вимог.

Створено повноцінну технічну та користувацьку документацію, яка описує процес встановлення, налаштування та використання системи. Документація включає інструкції з розгортання бази даних, запуску серверної та клієнтської частин, а також пояснення функціоналу для кінцевих користувачів, що забезпечує легкість у впровадженні системи в реальне середовище.

Отже, кваліфікаційна робота виконана у повному обсязі відповідно до поставлених завдань і може бути рекомендована для практичного використання, подальших досліджень та впровадження.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кащук К. М., Мосійчук І. В., Саух І. В. Сучасні технології управління в готельно-ресторанному бізнесі: практики та інновації. Бізнес інформ. 2023. №6. С. 93-99.

2. Лисюк Т. В. Інноваційні рішення в готельно-ресторанному бізнесі: технології автоматизації та персоналізації послуг. Економіка та суспільство. 2024. 67 с.

3. Toast. Built for restaurants. Built for you. URL: https://pos.toasttab.com/ (date of access: 02.02.2025).

4. Restaurant POS System and Software. Square for Restaurants. URL: https://squareup.com/us/en/point-of-sale/restaurants (date of access: 02.01.2025).

5. Restaurant POS system and payments – Lightspeed. URL: https://www.lightspeedhq.com/pos/restaurant/ (date of access: 05.03.2025).

6. Unified Modeling Language (UML). Bernhard Rumpe. URL: https://rumpe.github.io/research/Unified-Modeling-Language/ (date of access: 07.03.2025).

7. React. URL: https://react.dev/ (date of access: 10.03.2025).

8. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/en (date of access: 10.03.2025).

9. MySQL. URL: https://www.mysql.com/ (date of access: 07.04.2025).

10. Express – Node.js web application framework. URL: https://expressjs.com/ (date of access: 07.04.2025).

11. Grippa V. M., Kuzmichev S. Learning MySQL. O'Reilly Media, Inc. 2021. 632 p.

12. Lazuardy M. F. S., Anggraini D. Modern front end web architectures with React. js and Next. js. Research Journal of Advanced Engineering and Science. 2022. Vol. 1. №7 P. 132-141.

13. Anh V. Real-time backend architecture using Node. js, Express and Google Cloud Platform. 2021. 51 p.

14. Gascón U. Node. js for Beginners. A comprehensive guide to building efficient, full-featured web applications with Node. js. Packt Publishing Ltd. 2024. 382 p.

15. Wieruch R. The road to React. Robin Wieruch. 2020. 285 p.

16. Grippa V. M., Kuzmichev S. Learning MySQL. O'Reilly Media, Inc. 2021. 632 p.

17. Nguyen K. Building a tourism application with React and Nodes. Js. 2021. 71 p.

18. Banks A., Porcello E. Learning React. Modern patterns for developing React apps. O'Reilly Media. 2020. 310 p.