

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Система управління закладом харчування»

Виконав студент групи ПЗп-21-1
РЯБКО Ольга Денисівна

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **РЯБКО Ольга Денисівна**

Керівник роботи: **Вакалюк Тетяна Анатоліївна**

Тема роботи: **«Система управління закладом харчування»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи:

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т. А.	14 лютого 2025 р.	14 лютого 2025 р.
2	Вакалюк Т. А.	14 лютого 2025 р.	14 лютого 2025 р.
3	Вакалюк Т. А.	14 лютого 2025 р.	14 лютого 2025 р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку «Системи управління закладом харчування» та пояснювальної записки. Пояснювальна записка до випускної роботи містить 67 сторінок, 51 ілюстрацій та 20 таблиць. Список використаних джерел містить 27 найменувань. Загальний обсяг 101 сторінок.

Метою роботи є розроблення системи для автоматизації роботи закладів харчування, що охоплює управління меню, замовленнями, персоналом та формуванням звітності. Система спрямована на підвищення ефективності фундаментальних процесів та забезпечення узгодженої взаємодії між модулями.

В ході виконання роботи було визначено основні вимоги до системи, проведено аналіз вже існуючих сучасних рішень, а також сформовано архітектуру клієнт-серверного типу. Фронтенд частина реалізована з використанням React та Bootstrap, за допомогою яких було створено адаптивний та зручний інтерфейс. Серверна частина побудована на основі Spring Boot, що дозволило реалізувати логіку основних процесів системи, безпечну авторизацію та стабільну роботу з базою даних MySQL.

КЛЮЧОВІ СЛОВА: СИСТЕМА, ВЕБДОДАТОК, ЗАКЛАД ХАРЧУВАННЯ, АВТОМАТИЗАЦІЯ, ЗАМОВЛЕННЯ, МЕНЮ, SPRING BOOT, MYSQL, REACT, БАЗА ДАНИХ.

					ІПЗ.КР.Б-121-25-ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Рябко О.Д.			Система управління закладом харчування			Лім.	Арк.	Аркушів		
Перевір.		Вакалюк Т. А.								3	101	
Керівник								ФІКТ Гр. ІПЗп-21-2				
Н. контр.		Тичина О.П.										
Зав. каф.		Вакалюк Т. А.										

ABSTRACT

The Bachelor`s thesis consists of the web application Restaurant management system and explanatory note. The explanatory note to the thesis contains 67 pages, 51 illustrations, and 20 tables. The list of references contains 27 sources. The total volume is 101 pages.

The goal of the work is to develop a system for automating the operation of food service establishments, including menu management, order processing, staff management, and report generation. The system is aimed at improving the efficiency of fundamental processes and ensuring consistent interaction between its modules.

During the development process, the main system requirements were identified, an analysis of existing modern solutions was conducted, and a client–server architecture was designed. The frontend part was implemented using React and Bootstrap, which made it possible to create a responsive and user–friendly interface. The backend part was developed using Spring Boot, which enabled the implementation of the core system processes, secure authorization, and stable interaction with the MySQL database.

KEYWORDS: SYSTEM, WEB APPLICATION, RESTAURANT MANAGEMENT, AUTOMATION, ORDERS, MENU, SPRING BOOT, MYSQL, REACT, DATABASE.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ ..	10
1.1. Постановка задачі.....	10
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури для вебсистеми	18
1.4. Обґрунтування вибору інструментальних засобів	20
Висновки до першого розділу	22
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ.....	23
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи управління закладом харчування	23
2.2. Розробка бази даних системи.....	30
2.3. Проектування та реалізація алгоритмів роботи системи	35
2.4. Реалізація функціоналу системи управління закладом харчування	39
Висновки до другого розділу	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИТЕМОЮ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ	43
3.1. Структура інтерфейсу вебсистеми управління закладом харчування	43
3.2. Статистика продажів та загальна аналітика системи	59
3.3. Тестування вебсистеми.....	62
Висновки до третього розділу	66
ВИСНОВКИ.....	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68
ДОДАТКИ.....	71

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programing Interface

CRUD – Create, Read, Update, Delete

DTO –Data Transfer Object

HTML –Hyper Text Markup Language

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

JWT – JSON Web Token

MVC – Model-View-Controller

Spring Boot JPA – Spring Boot Java Persistence API

VS Code – Visual Studio Code

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даними

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність теми. Сфера закладів харчування перебуває в постійному розвитку: зростає конкуренція, підвищуються вимоги відвідувачів, а ринок орієнтується на швидкість та якість обслуговування. Використання застарілих фбо розрізаних інструментів ускладнює управління процесами, призводить до помилок та уповільнює роботу закладу. Зі збільшенням кількості клієнтів та обсягу інформації зростає потреба у системах, здатних оптимізувати повсякденні операції та забезпечити ефективну взаємодію між кухнею, офіціантами й адміністрацією.

Автоматизація діяльності закладів харчування дозволяє впорядковувати робочі процеси, зробити їх контрольованими та зменшити вплив людського фактору. Сучасні технології дають можливість створення зручних, швидких і доступних систем, що не потребують встановлення програмного забезпечення та можуть використовуватися на будь-якому пристрої.

Таким чином упровадження вебдодатку системи управління дає змогу значно вдосконалити організацію роботи, полегшити роботу персоналу та відстежувати виконання всіх необхідних дій у закладі. Тому створення такого програмного продукту є актуальним, адже воно сприяє оптимізації основних процесів, підвищенню продуктивності, поліпшенню рівня сервісу та бути конкурентноспроможним на сучасному ринку.

Метою кваліфікаційної роботи є розробка вебдодатку, реалізує автоматизовану підтримку ключових процесів закладу харчування. Система повинна надавати можливість працювати з меню, обробляти замовлення, координувати дії персоналу та формувати необхідну звітність.

Завданням кваліфікаційної роботи є дослідження особливостей функціонування закладів громадського харчування, аналіз існуючих систем та визначення вимог до майбутнього вебзастосунку. У межах роботи необхідно розробити структуру бази даних, спроектувати серверну частину за допомогою Spring Boot, створити інтерфейс користувача на основі React, а також впровадити інструменти автентифікації, правильне розмежування прав доступу та безпечну роботу з даними. Окремою задачею є розроблення модулів роботи з меню,

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

персоналом, замовленнями і статистикою. Однією з важливих частин є проведення тестування для перевірки коректності та стабільності функціонування системи.

Об’єктом дослідження є процеси автоматизації діяльності закладів харчування, що включає обробку замовлень, управління меню, координацію роботи персоналу та формування звітності.

Предметом дослідження є вебсистема управління закладом харчування, а також підходи до автоматизації процесів у даній сфері з використанням сучасних інструментів фронтенду, бекенду та баз даних.

Апробація. За результатами дослідження опубліковано тези у матеріалах конференцій [1-2].

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ

1.1. Постановка задачі

Сфера громадського харчування є однією з найбільш динамічних галузей на сьогодні, де якість обслуговування, швидкість реагування на запити клієнтів та ефективна організація внутрішніх процесів мають вирішальне значення. У таких умовах успіх будь-якого закладу залежить від того, наскільки злагоджено працює команда, наскільки швидко обробляються замовлення, наскільки точно ведеться облік, відслідковування фінансів та наскільки керівництво контролює усі процеси.

Водночас, значна кількість підприємств продовжує працювати за застарілими моделями управління : використовуючи ручний облік, розрізнені програми або несумісні між собою інструменти. Такий підхід створює підвищене навантаження на персонал, ускладнює обіг даних, спричиняє помилки під час оформлення замовлень та перешкоджає формуванню достовірної аналітики [3].

За відсутності комплексної системи закладу складно підтримувати належний рівень організації роботи: передача замовлень на кухню відбувається із затримками, контроль виконання ускладнюється, працівники витрачають більше часу на рутинні задачі, а адміністрація не має можливості оперативно оцінити ефективність діяльності. У результаті страждає якість сервісу, знижується швидкість обслуговування та втрачається конкурентоспроможність [4].

Враховуючи це, актуальним стає створення вебсистеми управління закладом харчування, здатної автоматизувати ключові робочі процеси та об'єднати їх у єдиному середовищі. Основна мета розробки полягає у створенні ПЗ, яке забезпечуватиме зручну взаємодію між персоналом, оптимізуватиме обробку замовлень, надаватиме засоби для управління меню, підтримуватиме аналітичні інструменти та гарантуватиме належний рівень безпеки даних.

Клієнтський інтерфейс реалізовано з використанням React, що підтримує компонентний підхід, стан керування та використання Hooks, зокрема useState та useEffect [5]. Використання Bootstrap дозволяє створити адаптивний та доступний

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейс, оптимізований для роботи на різних пристроях. Серверна частина створена засобами Spring Boot, який підтримує архітектуру REST API та забезпечує стандартизований доступ до бізнес-логіки [3]. Для роботи з даними у системі застосовується Spring Data JPA, що дозволяє використовувати ORM-підхід для роботи з сутностями, репозиторіями та запитами до бази даних [4]. Зберігання інформації реалізовано на основі реляційної СУБД MySQL [6].

Особливе значення має забезпечення безпеки даних і правильне управління доступом користувачів. Для автентифікації та авторизації використовується JWT, що гарантує безпечну передачу інформації між клієнтом і сервером та дозволяє реалізувати розподіл прав між різними категоріями користувачів : адміністратором, офіціантом, кухарем і менеджером [7, 8, 9, 10]. Вебдодаток використовує Spring Security, архітектура якого включає фільтрацію запитів, обробку токенів і перевірку прав доступу через AuthenticationManager та SecurityFilterChain [11].

З урахуванням функціональних вимог сформовано такі основні етапи розробки програмного забезпечення:

1. Проектування інтерфейсу користувача.
2. Розроблення архітектури вебсистеми, включаючи структуру REST API, модель даних, схеми взаємодії клієнта з сервером.
3. Створення та налаштування логічних компонентів системи, що відповідають за обробку даних у модулях замовлень, меню, персоналу, кухні та аналітики.
4. Проектування та створення бази даних MySQL.
5. Впровадження механізмів безпеки, включаючи JWT-автентифікацію, перевірку ролей користувачів та захист маршруту запитів.
6. Оптимізація роботи системи, поліпшення продуктивності та підготовка до розгортання.

Вебсистеми складається з окремих модулів, що реалізують ключові функції. Модуль користувачів забезпечує реєстрацію, вхід, керування профілем і рольовий доступ. Модуль замовлень відповідає за створення замовлень, зміну статусів, взаємодію з кухнею та завершення обробки. Модуль меню дозволяє керувати

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

товарами, категоріями та цінами. Модуль кухні надає можливість переглядати активні замовлення в режимі реального часу та фіксувати готовність страв. Модуль персоналу забезпечує управління працівниками та їх ролями. Модуль столів відповідає за контроль зайнятості та прив'язку замовлень. Аналітичний модуль формує звіти, графіки та статистику, необхідні для управлінського аналізу.

Отже, запропонована система створюється з метою оптимізації роботи закладів харчування шляхом автоматизації процесів, підвищення ефективності комунікації, зменшення кількості помилок та підвищення відкритості та зрозумілості процесів. Використання сучасних технологій веброзробки дає змогу реалізувати масштабовану, надійну та зручну систему.

1.2. Аналіз аналогів програмного продукту

Аналіз аналогів програмного продукту є важливою складовою процесу вибору оптимального рішення для автоматизації управління закладом харчування, оскільки дозволяє виявити сильні та слабкі сторони існуючих систем, зрозуміти тенденції розвитку галузі та визначити функціональні особливості, які доцільно реалізувати у власному програмному рішенні. Проведення такого дослідження перед створенням власної вебсистеми дає можливість оцінити доступні інструменти, порівняти їх можливості та врахувати реальні потреби користувачів для подальшого формування ефективного рішення.

Серед найпоширеніших систем, що активно використовуються в Україні та за її межами, можна виділити Poster POS, SkyService POS, Sygve Україна та SERVIO POS. Кожна з цих платформ має власну структуру, технічну реалізацію, набір інструментів та модель взаємодії з користувачами, тому вивчення цих аспектів дозволяє оцінити можливості, які сьогодні пропонують рішення для автоматизації підприємств харчової сфери.

Одним із найбільш відомих і поширених рішень в Україні є Poster POS [12]. Система орієнтована на малі та середні заклади харчування і пропонує інтерфейс, який адаптований під планшети та смартфони. Її ключовими перевагами є зручність використання, доступ до хмарної аналітики й наявність інструментів для управління меню, технологічними картами та складським обліком. Poster POS

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечує легке оновлення та редагування товарних позицій, підтримку знижок, поділ рахунків та контроль персоналу. Водночас вартість підписки є високою, а деякі модулі, наприклад функції доставки чи розширена аналітика, доступні лише за додаткову оплату. Система також має обмеження щодо кастомізації та вимагає стабільного інтернет-з'єднання. На рис. 1.1 показано інтерфейс оформлення замовлення в системі Poster POS.

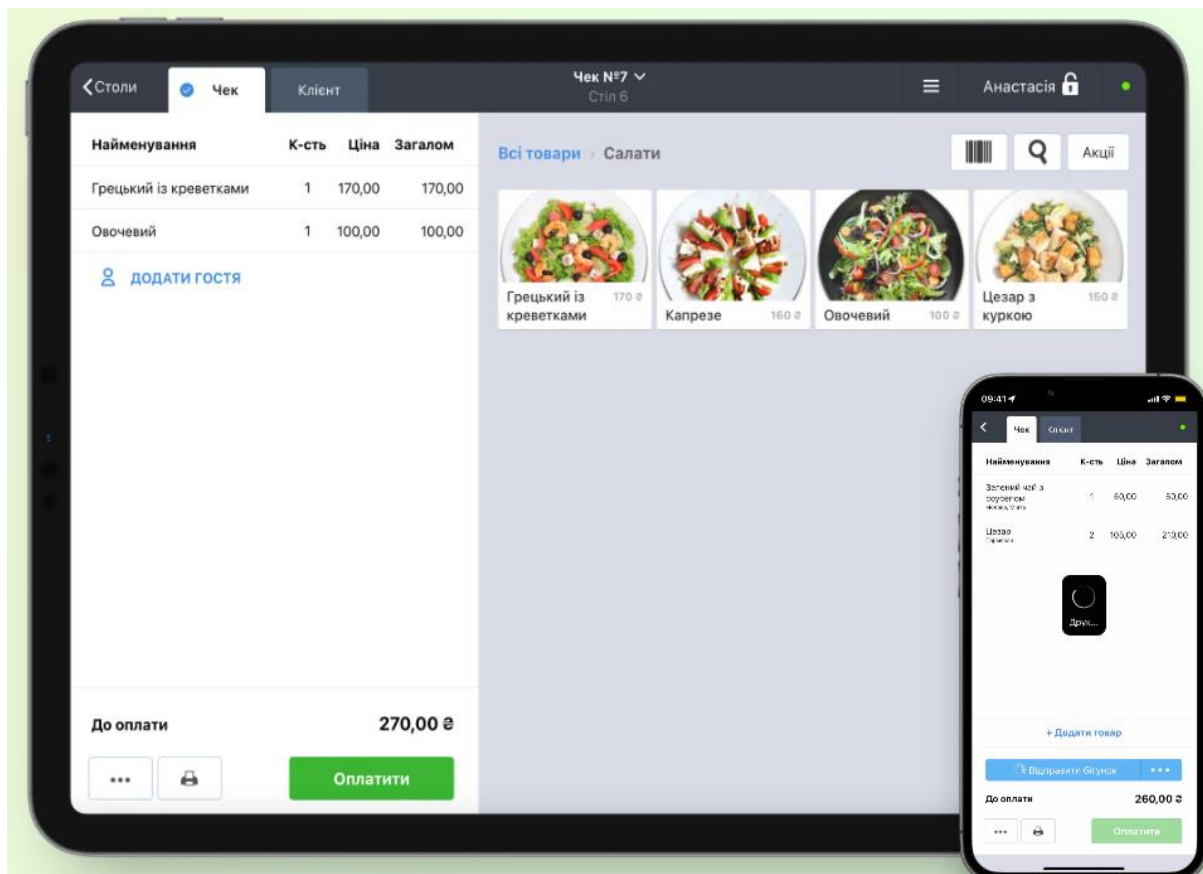


Рис. 1.1 – Інтерфейс модуля оформлення замовлень у системі Poster POS

Іншими особливостями Poster POS є потужні можливості у сфері складського обліку. Система автоматично списує інгредієнти на основі технологічних карт страв, дозволяє контролювати залишки та формувати закупівельні плани. Це робить її особливо привабливою для ресторанів із великим меню. Проте така деталізація може бути зайвою для невеликих кафе або фастфудів, де акцент робиться не на складському обліку, а на швидкості обслуговування. На рис. 1.2 наведено приклад складського модуля та на рис. 1.3 показано мобільний модуль аналітики в Poster POS.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

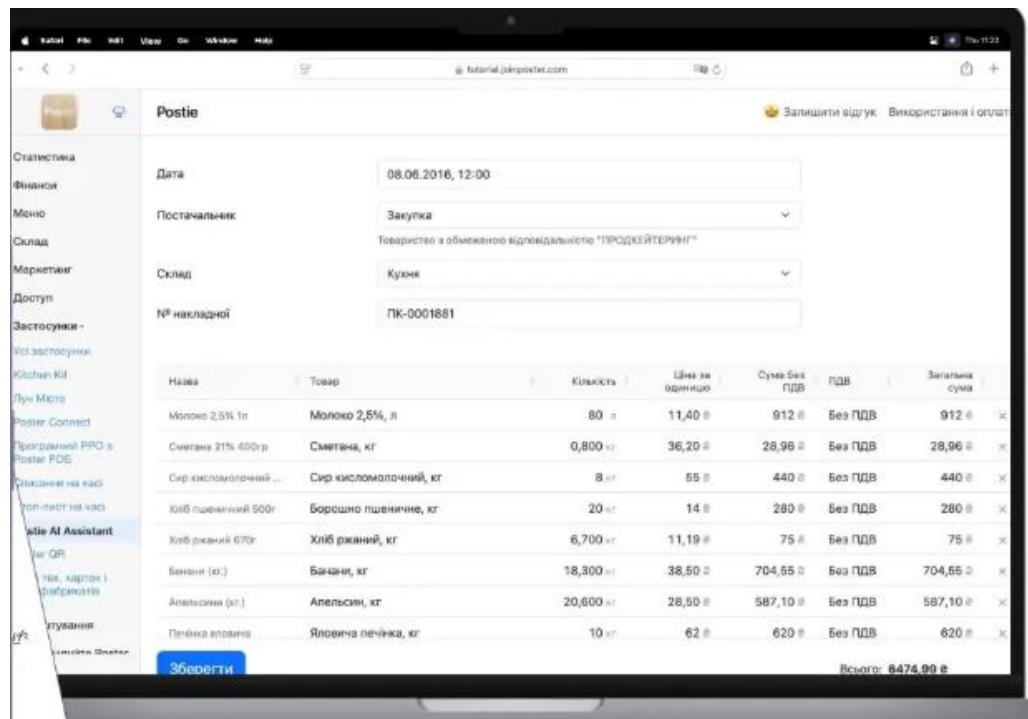


Рис. 1.2 – Модуль складського обліку в системі Poster POS

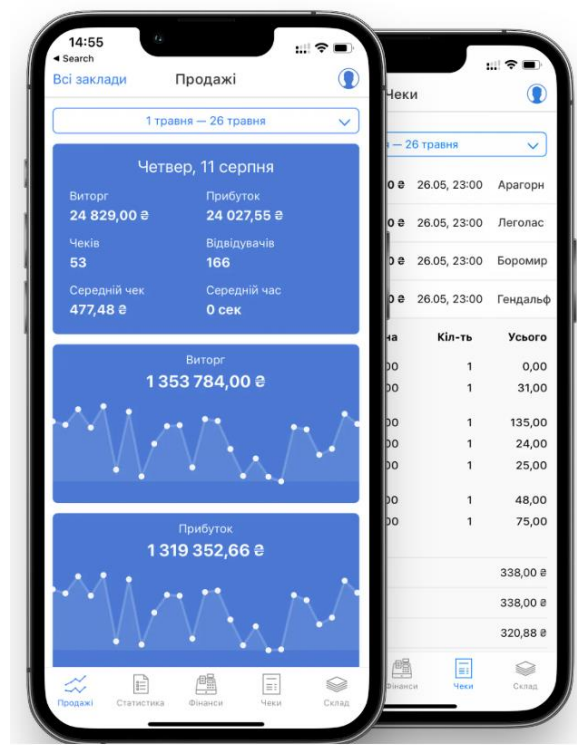


Рис. 1.3 – Мобільна аналітика й управління показниками в Poster POS

SkyService POS вирізняється тим, що пропонує широкую функціональність за порівняно низькою ціною, що робить систему доступною для малого та середнього бізнесу [13]. Вона підтримує роботу з меню, складами, персоналом, має мобільний застосунок та може працювати на будь-якому пристрої. Її додатковою перевагою є

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

наявність API, що дає змогу інтегрувати додаткові модулі. Водночас важливою складовою функціональності SkyService POS є система звітності, яка дозволяє закладам отримувати детальну інформацію про ключові показники діяльності. Платформа підтримує відображення статистики продажів у реальному часі, формування звітів за різні проміжки часу. Ще однією особливістю SkyService є можливість роботи в умовах нестабільного інтернет-з'єднання. Платформа підтримує режим офлайн-роботи, за якого каса може продовжувати обслуговування клієнтів навіть у разі тимчасової відсутності мережі. У такому режимі система зберігає всі створені замовлення та касові операції локально, а після відновлення інтернету автоматично синхронізує дані з хмарним сервером. На рис. 1.4 показано інтерфейс оформлення замовлень та панель аналітики даного програмного продукту.

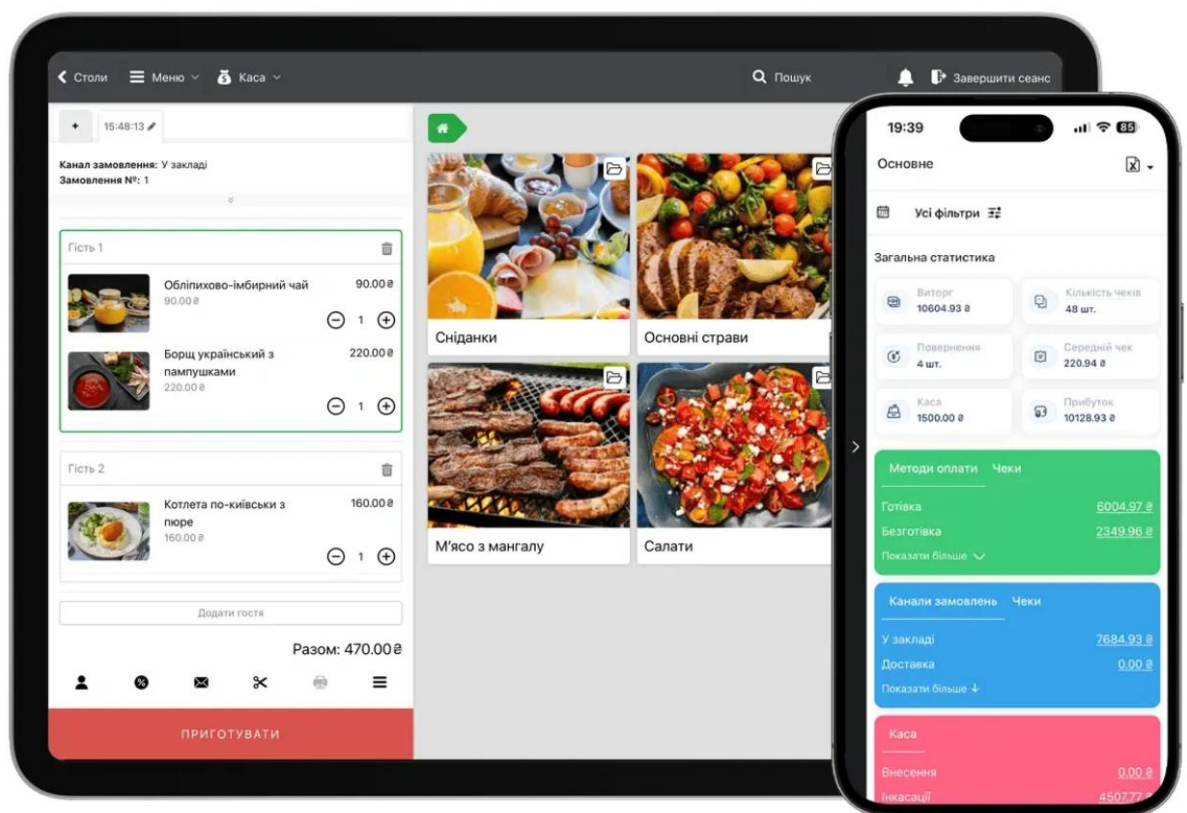


Рис. 1.4 – Інтерфейс системи SkyService

Ще одним поширеним рішенням на українському ринку є Sygve Україна, що представлений у форматі платформи, яка може інтегрувати інструменти для автоматизації закладів громадського харчування [14]. На відміну від традиційних POS-систем, Sygve Україна забезпечує гнучке управління робочими процесами,

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

підтримку ресторанних мереж і можливістю адаптації до різних моделей закладів. Платформа пропонує модулі для управління залом, роботою офіціантів, кухонними процесами, доставкою та фінансовою звітністю. Окрім того, вона включає можливість формування замовлень у режимі реального часу, передача замовлень на кухню та контроль за виконанням страв, що значно спрощує роботу на кухні й зменшує кількість помилок персоналу. Платформа також підтримує створення програм лояльності. Syrve Україна має досить складні налаштування, тому її впровадження може займати більше часу. Система більше підходить для середніх і великих ресторанів, а для маленьких закладів система може бути занадто складною та дорогою. На рис. 1.5 можна переглянути інтерфейс даної платформи.

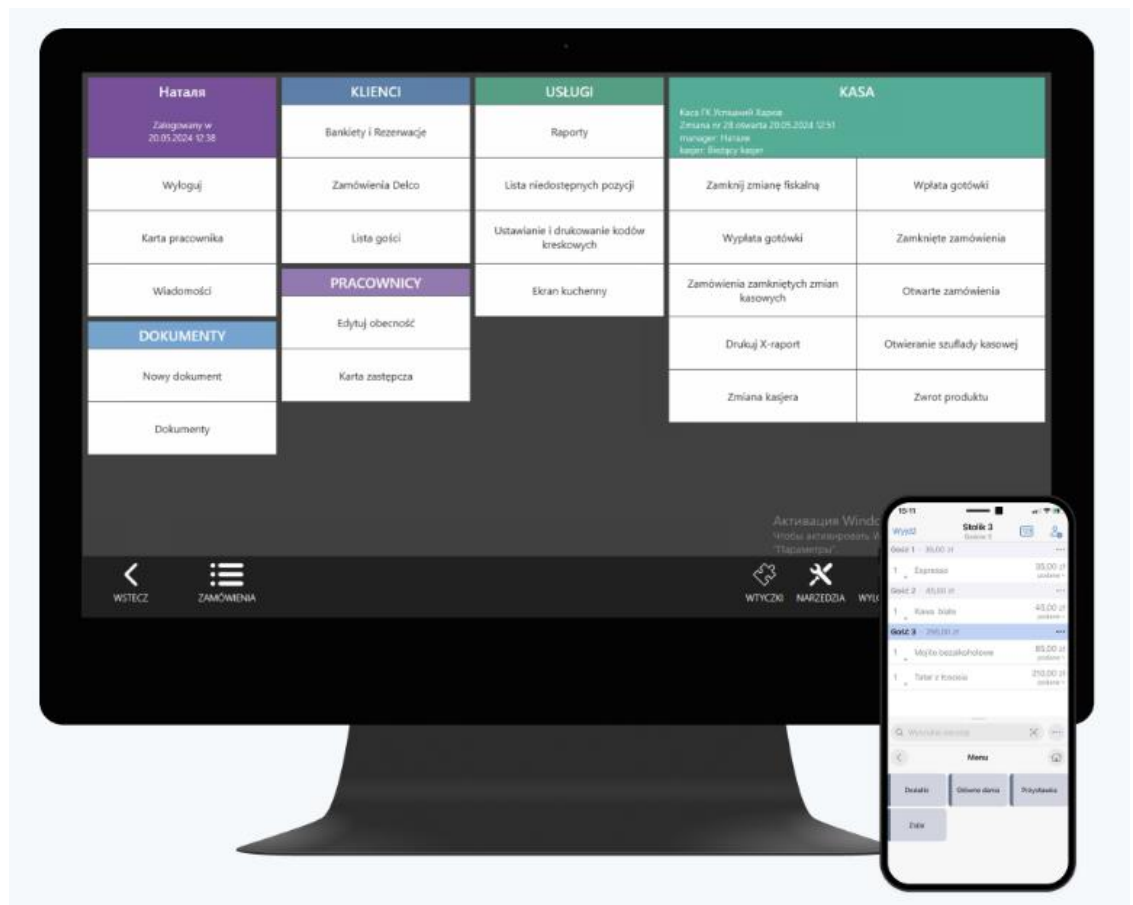


Рис. 1.5 – Приклад інтерфейсу Syrve Україна

Останньою серед розглянутих платформ є SERVIO POS — українське рішення, що пропонує комплекс інструментів для управління підприємствами харчової індустрії [15]. Система підтримує різні формати закладів і містить модулі для складського обліку, аналітики роботи з касовими операціями та програмою лояльності. Дане рішення часто встановлюється локально, що забезпечує закладу

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

повний контроль над даними, хоча й ускладнює масштабування та потребує технічного супроводу. До недоліків платформи можна віднести складніший інтерфейс та підвищену вартість при розширенні функціоналу додатковими модулями. На рис. 1.6 продемонстровано вигляд системи.

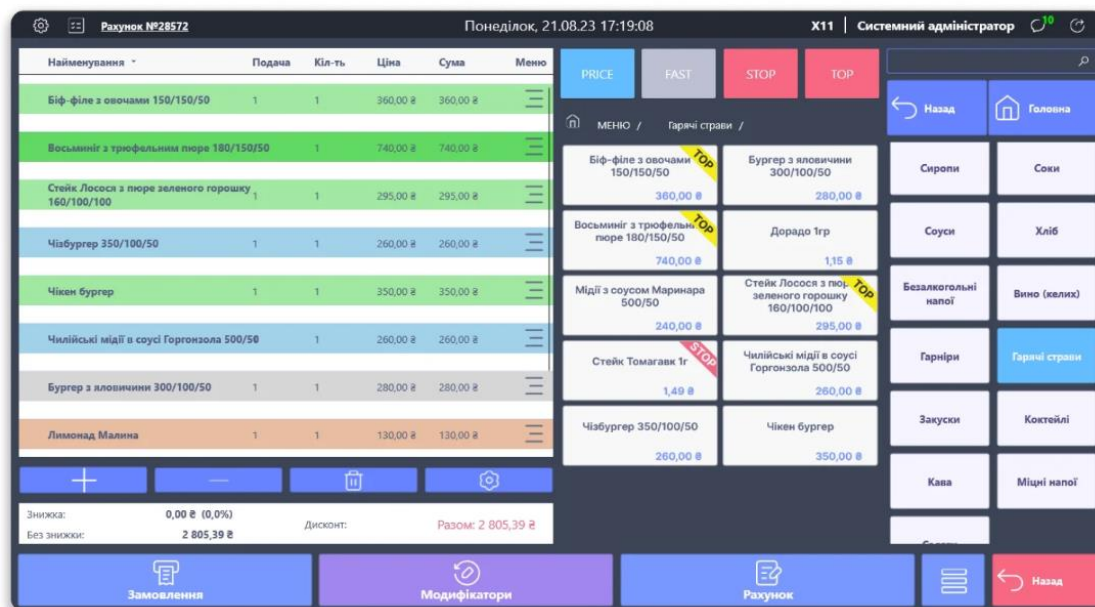


Рис. 1.6 – Інтерфейс SERVIO POS

Враховуючи наявні переваги, недоліки та відмінності у реалізації функціоналу, усі розглянуті аналоги спрямовані на забезпечення ефективної діяльності закладів харчової індустрії. Їх ключові особливості дають змогу визначити сильні сторони кожного рішення та використати їх під час розроблення власної системи. Узагальнене порівняння подано в таблиці 1.1.

Таблиця 1.1

Характеристика	Poster POS	SkyService POS	Syrve Україна	SERVIO POS	Моя система
Управління замовленнями	+	+	+	+	+
Управління столами	+	+	+	+	+
Цифрове меню	-	-	+/-	-	+
CRUD для меню	+	+	+	+	+
Облік	+	+	+	+	+/-
Хмарне рішення	+	+	+/локальне	локальне	+

Управління персоналом	+	+	+	+	+
Передача замовлень на кухню	+	+	+	+	+
Різні ролі користувачів	+	+	+	+	+
Аналітика та звітність	+	+	+	+	+
Робота на різних пристроях	+	+	+	+/-	+

У результаті проведеного аналізу встановлено, що кожна з розглянутих систем має власний підхід до автоматизації основних процесів та різний рівень функціональності. Виявлені сильні сторони стали основою для формування архітектури власної системи. Серед аналізованих платформ найбільш доцільними для використання як зразок стали Poster POS та Syrve Україна, оскільки вони поєднують зручність роботи персоналу та високий рівень автоматизації.

1.3. Вибір архітектури для вебсистеми

Проектування вебсистеми передбачає вибір архітектурного підходу. Для системи управління закладом харчування, яка включає модулі обліку замовлень, управління меню, персоналом, столами, звітністю та взаємодією між користувачами різних ролей, важливо обрати архітектуру, здатну ефективно обробляти паралельні запити, гарантувати цілісність даних і підтримувати масштабування. Враховуючи потреби, для реалізації програмного продукту було вибрано клієнт-серверну архітектуру, оскільки вона є найпоширенішим підходом для сучасних вебзастосунків і дозволяє розділити роботу між різними частинами системи [16].

Клієнтська частина реалізована засобами React, що забезпечує зрозумілий та інтуїтивний інтерфейс, дозволяє виконувати асинхронні запити до сервера та

оновлювати дані без перезавантаження сторінки, що добре впливає на полегшення роботи користувача та масштабованість фронтенду[5].

Серверна частина побудована на Spring Boot, який містить інструменти для швидкої розробки, підтримки REST API та стабільної роботи з базою даних [17]. Архітектура бекенду організована за принципами Spring MVC, де логіка розподілена між контролерами, сервісами та репозиторіями, що значно полегшує тестування й подальшу підтримку системи.

Взаємодія між клієнтом і сервером здійснюється через REST API, що дозволяє отримувати та передавати дані у форматі JSON, який є простим та зручним [3]. Для передачі запитів використовуються стандартні HTTP-методи (GET, POST, PUT, DELETE), що повністю відповідає CRUD-операціям управління даними. REST-підхід робить клієнта і сервер незалежними один від одного, тому у фронтенді можна замінювати окремі частини системи без зміни всього застосунку (рис 1.7).

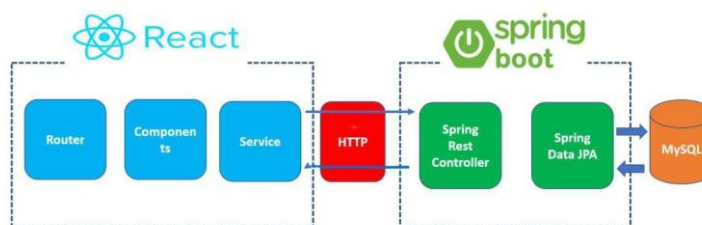


Рис. 1.7 – Структура взаємодії клієнтської та серверної частин вебсистеми

Для роботи з базою даних використовується Spring Data JPA, який спрощує взаємодію з MySQL та дає можливість працювати з даними без написання складних SQL-запитів. Це робить створювання сутностей, репозиторіїв та виконання запитів значно простішим, водночас система не втрачає швидкодії [4].

Окремим елементом архітектури є JWT-автентифікація, яка забезпечує безпечну передачу даних про користувача та керування його доступом до системи. Її використання відповідає рекомендаціям Spring Security і є стандартом для REST-орієнтованих застосунків [11].

Таким чином, клієнт-серверна архітектура разом із технологіями React, Spring Boot та MySQL дозволяє створити зручну та продуктивну вебсистему, яка може працювати при великій кількості користувачів. Сервер швидко обробляє

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

HTTP-запити та взаємодіє з базою даних, а клієнтська частина оновлює інтерфейс. Крім того, Spring Boot дає можливість розділяти логіку на модулі, тому розширення функціоналу або внесення змін виконіються легко.

Обрана архітектура забезпечує баланс між швидкістю роботи, зручністю та простотою розробки, що робить її доцільною для створення системи управління закладом харчування.

1.4. Обґрунтування вибору інструментальних засобів

Розроблення вебдодатку управління закладом харчування потребує вибору інструментальних засобів, які будуть відповідати вимогам надійності, продуктивності, масштабованості та зручності підтримки програмного продукту. Оскільки система складається з клієнтської та серверної частин і повинна працювати з пов'язаними між собою даними, важливо підібрати технології, які добре взаємодіють одна з одною та дозволяють реалізувати необхідний функціонал без зайвих ускладнень.

Для створення клієнтської частини обрано технологію React. Ця технологія дає змогу створити зрозумілий, сучасний та інтуїтивний інтерфейс, що швидко реагує на дії користувача без повного. React має детальну документацію та забезпечує гнучкість під час розширення функціоналу, що робить його зручним варіантом для подібних вебзастосунків [5].

Оскільки інтерфейс системи містить багато таблиць, форм, кнопок та модальних вікон, для стилізації було використано фреймворк Bootstrap. Завдяки готовим компонентам цей фреймворк дозволив швидко оформити вигляд системи та забезпечити єдиний стиль у всіх розділах. Bootstrap легко інтегрується з React, тому оформлення інтерфейсу не вимагало складних стилів і не відволікало від основної логіки роботи [18].

Серверна частина реалізована на основі Spring Boot. Цей фреймворк забезпечує швидкий старт проекту завдяки автоматичній конфігурації та має всі необхідні засоби для створення REST-орієнтованих вебдодатків. Використання Spring MVC дозволило розділити логіку програми на окремі частини та зробити систему більш зрозумілою й зручною в подальшій підтримці. Для вебдодатку, де

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

одночасно працюють користувачі різних ролей і відправляється багато запитів, Spring Boot є оптимальним варіантом завдяки своїй продуктивності та стабільності [17]. Щоб підтвердити доцільність вибору технологій, проведено порівняння з Node.js із фреймворком Express. в таблиці 1.2.

Таблиця 1.2

Характеристика	Spring Boot	Node.js(Express)
Мова	Java	JavaScript
Продуктивність	Висока	Середня
Безпека	Вбудована Spring Security	Потрібні зовнішні модулі
Маштабованість	Висока	Висока
Робота з БД	Доступ через JPA	ORM

Обидва фреймворки часто використовують у веброзробці, але Spring Boot має більш чітку структуру та вбудовані інструменти для безпеки, що важливо для систем, які працюють з даними користувачів. Node.js добре підходить для простіших або менших проектів, але не містить так багато готових можливостей для складних бізнес-систем. Тому для розробки даного програмного забезпечення, Spring Boot підходить краще

Для зберігання даних обрано MySQL, оскільки вона є однією з найпоширеніших реляційних баз даних. Вона легко інтегрується зі Spring Boot, забезпечує високу швидкодію, підтримку транзакцій та просте адміністрування. У системі, де дані мають чіткі зв'язки реляційна модель є найбільш доцільною [6]. Порівняння баз даних в таблиці 1.3.

Таблиця 1.3

Характеристика	MySQL	PostgreSQL	MongoDB
Тип	Реляційна	Реляційна	Документна
Складність налаштування	Низька	Середня	Середня
Інтеграція з Spring	Дуже проста	Проста	Через додаткові драйвери
Підходить для CRUD	Так	Так	Частково
Підтримка транзакцій	Висока	Висока	Низька

Під час аналізу також розглядались PostgreSQL та MongoDB. PostgreSQL є потужною та гнучкою СУБД, однак потребує складнішого налаштування, що робить її менш зручною для проекту. MongoDB є документною базою і підходить для систем із менш структурованими даними, тому вона не відповідає вимогам цього вебдодатку. Через це MySQL була обрана як сама зручна технологія для роботи з даними у системі.

Висновки до першого розділу

У першому розділі було обґрунтовано актуальність створення сучасної системи управління закладом харчування. Сформульовано мету роботи та окреслено основні вимоги до майбутнього програмного забезпечення з урахуванням потреб персоналу та клієнтів. Також проведено аналіз існуючих рішень, таких як Poster POS, SkyService POS, Syrve Україна та SERVIO POS. Це дозволило побачити їхні переваги та недоліки й зрозуміти, які можливості варто реалізувати у власній системі.

На основі отриманих висновків було обрано клієнт-серверну архітектуру, оскільки вона забезпечує зручну взаємодію між клієнтом і сервером і добре підходить для роботи з великою кількістю даних. Серверну частину вирішено реалізувати на Java із використанням Spring Boot, а клієнтської частини на React. Для збереження даних обрано MySQL як надійну та поширену реляційну базу.

Таким чином, у першому розділі сформовано повне розуміння того, яку систему необхідно створити, яким має бути її функціонал і обрані саме такі технології для реалізації вебсистеми управління закладом харчування.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи управління закладом харчування

Система управління закладом харчування надає зручний веб-інтерфейс, який можуть одночасно використовувати різні категорії користувачів, такі як : адміністратори, менеджери, офіціанти та кухарі. Кожен користувач отримує доступ тільки до тих функцій, що відповідають його ролі.

Щоб наочно показати взаємодію всіх учасників та ключові бізнес-процедури, було створено діаграму варіантів використання [19] (рис. 2.1), де чітко відображені ролі, їхні основні сценарії дій та зв'язки між ними. Завдяки цій діаграмі можна легко відстежити послідовність операцій: від входу в систему до формування звітів.

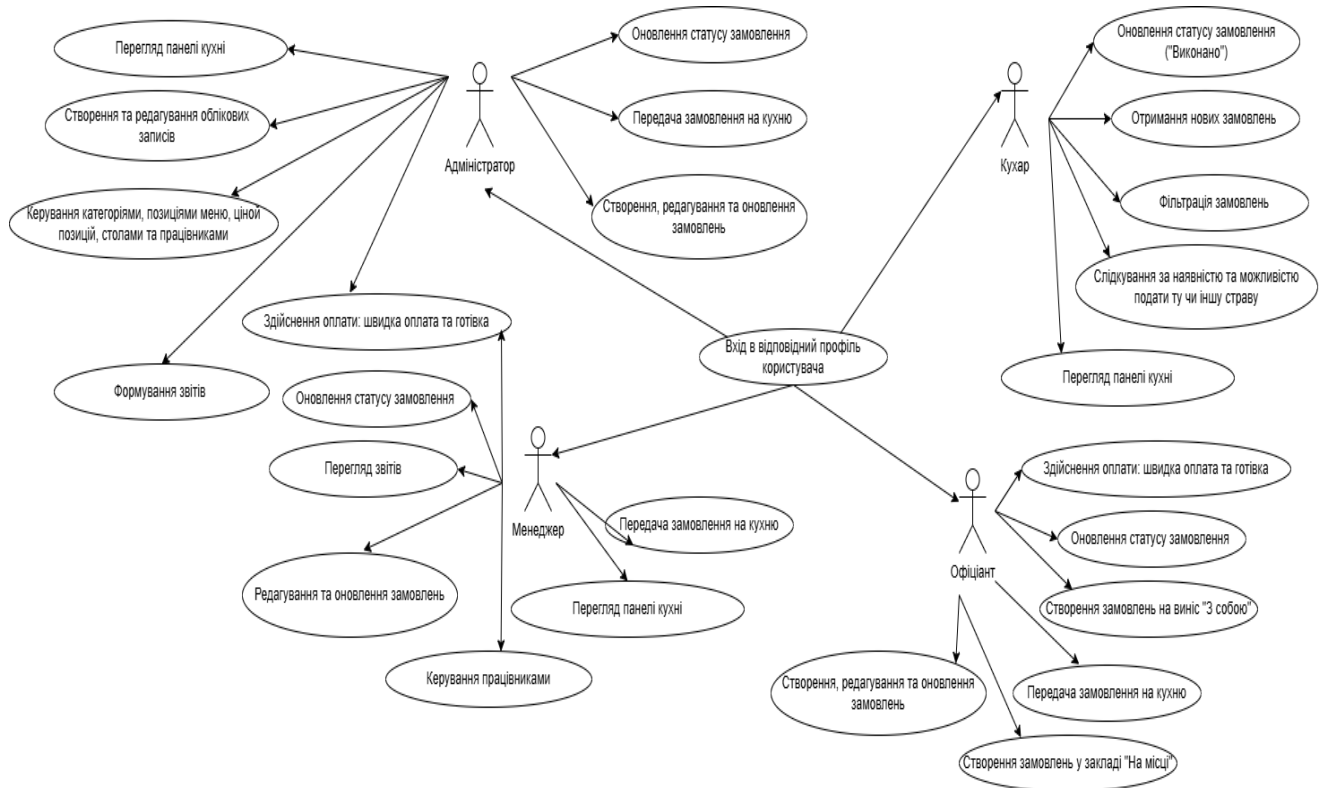


Рис. 2.1 – Діаграма варіантів використання

Опис діаграми варіантів використання наведено в таблицях 2.1 -2.8

Таблиця 2.1

Опис прецеденту «Вхід у відповідний профіль користувача»

Назва варіанту використання	Вхід у відповідний профіль користувача
-----------------------------	--

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

Основні актори	Адміністратор, менеджер, офіціант, кухня
Короткий опис	Авторизація передбачає, що користувач відкриває сторінку входу та вводить свої облікові дані (логін і пароль). Система перевіряє коректність введеної інформації, після чого визначає роль користувача та надає йому доступ до відповідного інтерфейсу. Залежно від прав доступу це може бути панель адміністратора, менеджера, офіціанта або кухаря. У випадку успішної автентифікації система завантажує персональну панель та відкриває користувачеві можливість виконувати дії, передбачені його роллю.

Таблиця 2.2

Опис прецеденту «Створення, редагування та оновлення замовлень»

Назва варіанту використання	Створення, редагування замовлень
Основні актори	Адміністратор, менеджер, офіціант
Короткий опис	Для створення нового замовлення користувач відкриває вкладка Система та обирає тип замовлення «На місці» або «З собою». Після цього він формує замовлення, додаючи необхідні позиції меню та за потреби обрати кількість або змінити вибрані страви. Система дозволяє редагувати вміст замовлення. Після завершення формування користувач зберігає зміни і замовлення фіксується у системі.

Таблиця 2.3

Опис прецеденту «Створення замовлення в закладі «На місці» »

Назва варіанту використання	Створення замовлення в закладі «На місці»
Основні актори	Адміністратор, офіціант
Короткий опис	Користувач обирає вільний стіл у системі, після чого формує склад замовлення, додаючи потрібні позиції меню. Коли замовлення повністю сформоване, користувач передає його на кухню для подальшого приготування. У результаті замовлення автоматично прив'язується до вибраного столу, що дозволяє відстежувати його статус.

Таблиця 2.4

Опис прецеденту «Створення замовлення на виніс «З собою»»

Назва варіанту використання	Створення замовлення на виніс «З собою»
Основні актори	Адміністратор, офіціант
Короткий опис	Створення замовлення «З собою» передбачає оформлення замовлення без прив'язки до конкретного столу. У цьому випадку користувач обирає у системі тип замовлення «З собою» та при потребі у додаткове поле вводить додаткову інформацію про клієнта та додає необхідні позиції меню. Після формування списку страв замовлення передається на кухню для приготування. Потім замовлення переходить в обробку кухні та відображається у панелі кухаря як нове замовлення на виніс.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Опис прецеденту «Передача замовлення на кухню»

Назва варіанту використання	Передача замовлення на кухню
Основні актори	Офіціант, менеджер, адміністратор
Короткий опис	Після формування замовлення користувач перевіряє його коректність. Коли замовлення готове до виконання, користувач натискає кнопку «Передати на кухню», після чого система змінює його статус на «Готується». Замовлення одразу з'являється у панелі кухні, де кухарі можуть переглянути його та розпочати процес приготування.

Таблиця 2.6

Опис прецеденту «Отримання нових замовлень»

Назва варіанту використання	Отримання нових замовлень
Основні актори	Кухня
Короткий опис	Після передачі замовлення на кухню воно автоматично відображається на інформаційній панелі кухні, де користувач може переглянути всі нові надходження. Користувач одразу бачить список замовлень, які очікують на приготування, та їх вміст. Дане замовлення можна розрукувати для зручного сприйняття. І після виконання всього замовлення користувач натискає кнопку «Виконано», після цього змінюється статус замовлення на «Виконано» і замовлення вже готове до оплати.

Опис прецеденту «Оновлення статусу замовлення»

Назва варіанту використання	Оновлення статусу замовлення
Основні актори	Адміністратор, менеджер, офіціант, кухня
Короткий опис	У процесі роботи користувач з кухні обирає потрібне замовлення, коли воно завершене, зі списку та змінює його статус на «Виконано». Після оновлення статусу система автоматично передає зміни. Користувачі, що працюють з самою системою при створення замовлення і передачі його на кухню автоматично встановлюється статус «Готується». І після здійснення оплати статус змінюється на «Сплачено». Завдяки цьому користувачі мають актуальні дані про стан замовлення, що сприяє узгодженості роботи та підвищує ефективність обслуговування.

Опис прецеденту «Здійснення оплати: швидка оплата та готівка»

Назва варіанту використання	Здійснення оплати: швидка оплата та готівка
Основні актори	Адміністратор, менеджер, офіціант, кухня
Короткий опис	Завершення роботи із замовленням передбачає його оплату. Користувач обирає замовлення, що потрібно оплатити, після чого визначає спосіб розрахунку : швидка оплата або оплата готівкою. Якщо клієнт розраховується готівкою, у систему вводиться отримана сума і програма

	автоматично розраховує решту. Після успішної оплати система змінює статус замовлення на «Оплачено».
--	---

Функціональні вимоги

Функціональні вимоги визначають, які саме дії повинна підтримувати система та як саме повинні працювати її основні модуля. Основна мета полягає в тому, щоб автоматизувати ключові процеси у закладі та створити зручні інструменти для персоналу під час роботи із замовленнями, меню, оплатами та звітністю.

Авторизація з розмежуванням прав доступу. Це означає, що адміністратор, менеджер, офіціант та кухар входять у систему під своїми обліковими записами та бачать лише той функціонал, який відповідає їхнім обов'язкам.

Адміністратор має найширші можливості: він створює й редагує користувачів, призначає ролі, керує категоріями, позиціями меню, списком столів, працівниками, звітність та загальними налаштуваннями. Адміністратор також відповідає за контроль роботи всієї системи.

Робота із замовленнями. Офіціант повинен мати можливість створити нове замовлення, визначити його тип: «На місці» або «З собою», додавати чи змінювати позиції меню та передавати сформоване замовлення на кухню. Після передачі замовлення кухар бачить його у своїй панелі та може перейти до приготування. У процесі виконання система повинна підтримувати зміну статусів : від «Готується» до «Виконано» та «Сплачено», щоб усі учасники процесу бачили актуальну інформацію. Теж саме може робити й адміністратор, а менеджер редагувати.

Оплата замовлення. Система має підтримувати як швидку оплату, так і розрахунок готівкою з автоматичним визначенням решти. Після завершення оплати замовлення повинно змінити статус на «Сплачено».

Менеджер також контролює роботу персоналу, слідкує за виконанням замовлень і може втручатися у процес у разі потреби. Додатково менеджер має

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

доступ до аналітики, включаючи перегляд звітів та статистики продажів, що допомагає оцінювати навантаження залу, продуктивність кухні.

Кухарі повинні отримувати список нових замовлень, переглядати позиції, які містяться в ньому та оновлювати статус, коли процес приготування завершений.

Отже, функціональні вимоги формують повний цикл роботи закладу харчування та забезпечуючи зручність і узгодженість роботи всіх ролей у системі.

Нефункціональні вимоги

Нефункціональні вимоги описують якість роботи системи, її зручність, надійність та технічні характеристики.

Вебсистема повинна швидко реагувати на дії користувача без затримок. Оскільки заклади громадського харчування працюють в динамічному середовищі, сторінки та функції повинні швидко завантажуватися.

Важливим аспектом є зручність використання. Інтерфейс повинен бути зрозумілим та інтуїтивним, щоб працівники могли легко розібратися у роботі з програмою. Складні або перевантажені елементи інтерфейсу можуть сповільнити роботу персоналу і створити додаткові труднощі під час обслуговування гостей, тому інтерфейс повинен бути максимально простим і логічним..

Також однією з важливих вимог є те, що система повинна забезпечувати надійність збереження даних. Усі замовлення, оплати та зміни статусів мають фіксуватися коректно й не зникати навіть у випадку помилкових дій користувача або технічних збоїв. Вебзастосунок повинен залишатися працездатним та швидко відновлювати функціонування після можливих проблем.

Окремо слід виділити вимоги до інформаційної безпеки. Дані користувачів, включаючи їх паролі та інформацію про ролі, повинні бути захищені. Доступ до системи має здійснюватися через безпечний механізм авторизації на основі JWT-токенів. Система повинна чітко дотримуватися розмежування прав доступу, щоб жоден користувач не міг отримати більше можливостей, ніж передбачено його роллю.

Ще один критерій це масштабованість. Програмний продукт повинен мати можливість працювати зі збільшеною кількістю користувачів та обробляти

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

більший потік замовлень без зниження продуктивності. Це дає можливість використовувати систему як у невеликих кафе, так і у великих ресторанах з високим навантаженням.

Таким чином, нефункціональні вимоги забезпечують якість роботи системи та роблять її стабільною, безпечною, зручною і готовою до роботи в реальних умовах закладів харчування.

Системні вимоги

Системні вимоги визначають технічні параметри, необхідні для роботи з вебсистемою, а також програмне забезпечення, яке повинно підтримувати роботу застосунку.

Оскільки система розроблена як вебзастосунок, вона має бути сумісною з різними браузерами та пристроями. Тому важливо, щоб клієнтська частина коректно працювала у сучасних кросбраузерних середовищах, таких як Google Chrome, Mozilla Firefox та Microsoft Edge.

Для роботи клієнтської частини достатньо комп'ютера з 4 ГБ оперативної пам'яті та звичайним двоядерним процесором. Система потребує стабільного інтернету, бо відкривається через браузер. Серверна частина працює на Spring Boot, тому потрібна встановлена Java. Для її роботи вистачить сервера з кількома ядрами та приблизно 4 ГБ оперативної пам'яті, щоб система могла обробляти запити без затримок.

Система використовує базу даних MySQL, тому сервер або хостинг повинні підтримувати її запуск. Обов'язковою вимогою є налаштування регулярного резервного копіювання, щоб запобігти втраті важливих даних у разі технічних збоїв або непередбачених ситуацій.

2.2. Розробка бази даних системи

База даних є ключовою частиною системи управління закладом харчування, оскільки саме вона зберігає інформацію про категорії, меню, замовлення, персонал, столи. Мета проектування бази даних полягає в тому, щоб забезпечити надійне зберігання інформації, зручність доступу до неї та можливість подальшого масштабування системи. Для реалізації програмного рішення було обрано

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

реляційну СУБД MySQL тому, що вона підтримує зрозумілу побудову таблиць, чітко визначені зв'язки між сутностями та стандарт SQL, який детально описано в офіційній документації [6].

Під час моделювання бази даних використовувалась реляційна модель, що передбачає поділ інформації на таблиці з первинними ключами, а зв'язки між ними реалізуються за допомогою зовнішніх ключів [20]. Такий підхід допомагає уникнути плутанини та забезпечує логічну організацію даних.

Щоб мінімізувати дублювання інформації, застосовано правила нормалізації відповідно до рекомендацій Microsoft [21]. У результаті дані було розділено на логічні групи: окремі таблиці для категорій, продуктів, замовлень, користувачів, ролей та позицій меню. Нормалізація зробила структуру простою для подальшого розширення і зручною в обслуговуванні.

Для збереження таблиць у вебсистемі використовується рушій InnoDB, який підтримує транзакції та гарантує дотримання правил ACID [22, 23]. Також, InnoDB підтримує зовнішні ключі та автоматичні дії при видаленні чи зміні пов'язаних записів, що допомагає правильно підтримувати зв'язки між таблицями. У підсумку створена база даних забезпечує логічний поділ інформації, надійність транзакцій, можливість масштабування та ефективну обробку запитів.

На рисунку 2.2 зображено діаграма бази даних розробленої системи управління закладом харчування. Вона об'єднує основні сутності, що використовуються у роботі застосунку, та показує логічні зв'язки між ними. Всього 8 таблиць та кожна таблиця відповідає окремому об'єкту системи та містить дані, що стосуються його призначення: категорії, позиції меню, доступні розміри та ціни, замовлення, позиції замовлень, персонал, користувачі та столи.

Модель бази даних побудована так, щоб інформація зберігалася впорядковано та логічно. Таблиці пов'язані між собою зовнішніми ключами, тому дані залишаються цілісними і всі операції створення, редагування та обробки замовлень виконуються правильно. Логічні зв'язки між таблицями дають можливість використовувати дані для подальшої статистики, аналітики та формування звітів.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

8. Order_items – зберігає перелік усіх позицій, що входять до конкретного замовлення, разом із кількістю та цінами.

Опис кожної з таблиць бази даних наведений у таблицях 2.9 – 2.16.

Таблиця 2.9

Опис таблиці Users

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор користувача
username	varchar(255)	Логін користувача для авторизації
password	varchar(255)	Хешований пароль користувача
user_role	enum (ROLE_ADMIN, ROLE_KITCHEN, ROLE_MANAGER, ROLE_WAITER)	Роль користувача в системі

Таблиця 2.10

Опис таблиці Tables

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор столу
name	varchar(255)	Назва або номер столу

Таблиця 2.11

Опис таблиці Staff

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор працівника
active	bit(1)	Ознака активності працівника
name	varchar(255)	ПІБ працівник
phone	varchar(255)	Контактний номер
position	varchar(255)	Посада працівника

Таблиця 2.12

Опис таблиці Categories

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор категорії
name	varchar(255)	Назва категорії меню

Таблиця 2.13

Опис таблиці Products

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор позиції меню
image_url	varchar(255)	Шлях до зображення позиції меню
in_stock	bit(1)	Ознака наявності позиції меню
name	varchar(255)	Назва позиції меню
category_id	bigint	FK- Посилання на категорію

Таблиця 2.14

Опис таблиці Product_sizes

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор розміру позиції
price	decimal	Вартість конкретного розміру позиції
size_label	varchar(255)	Назва розміру
product_id	bigint	FK- Відповідність на позицію

Таблиця 2.15

Опис таблиці Orders

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор замовлення
created_at	datetime	Час створення замовлення
customer_name	varchar(255)	Ім'я клієнта

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

order_number	varchar(255)	Номер замовлення
order_type	enum (DINE_IN, TAKEAWAY)	Тип замовлення
status	enum (PREPARING, DONE, PAID)	Статус замовлення
total_amount	decimal	Загальна сума
updated_at	datetime	Час останнього оновлення
table_id	bigint	FK- Посилання на стіл
waiter_id	bigint	FK- Посилання на офіціанта

Таблиця 2.16

Опис таблиці Order_items

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор позиції замовлення
product_name	varchar(255)	Назва позиції під час замовлення
quantity	int	Кількість одиниць даної позиції
size_label	varchar(255)	Обраний розмір позиції
total_price	decimal	Загальна вартість позиції
unit_price	decimal	Ціна однієї одиниці
order_id	bigint	FK- Посилання на замовлення
product_id	bigint	FK- Посилання на позиції
size_option_id	bigint	FK- Посилання на варіант розміру

2.3. Проектування та реалізація алгоритмів роботи системи

Проектування алгоритмів для системи управління закладом харчування проводилося з метою спростити користування програмою, зменшити кількість можливих помилок та прискорити виконання щоденні операції. Щоб алгоритми

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

працювали правильно, необхідно розуміти реальні бізнес-процеси закладу, типи користувачів, можливі ризики та технічні обмеження самої системи. На початковому етапі були визначені основні сценарії взаємодії, після чого кожен з них був детально проаналізований, розбитий на окремі дії та поданий у вигляді діаграм активності. Такий підхід дав змогу узгодити логіку між фронтендом і бекендом та забезпечити узгоджену роботу всієї системи.

Під час розроблення алгоритмів увага приділялася тому, щоб кожен крок алгоритму був логічно завершеним, зрозумілим та не створював зайвої складності. Важливим елементом стали перевірки на помилки, оскільки система має забезпечувати стабільність роботи та коректно реагувати на неправильний ввід або нестандартні ситуації. Оскільки в системі передбачені різні ролі користувачів, значну роль під час проектування відігравала також безпека та правильна реалізація перевірки прав доступу.

Далі розглядаються два алгоритми, від яких залежить коректна робота всієї системи. Алгоритм створення замовлення зображено на рисунку 2.3, також описано детально його пункти.

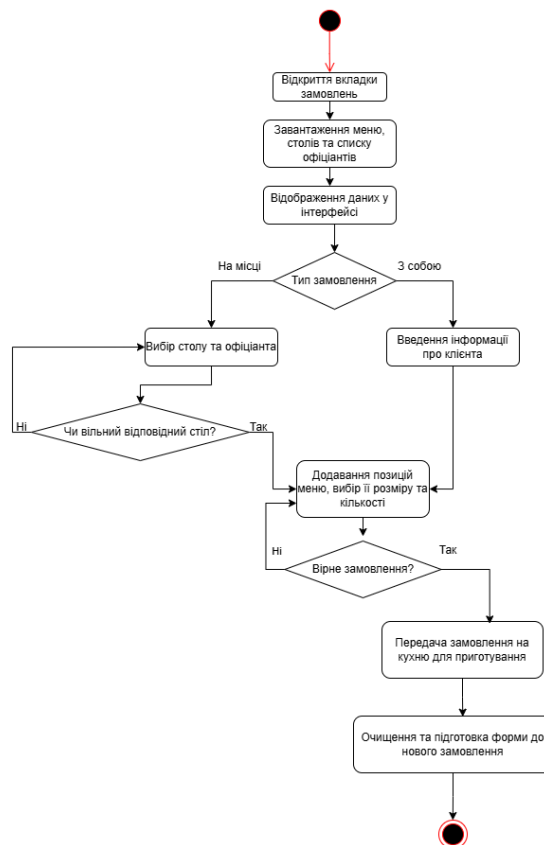


Рис. 2.3 – Діаграма активності оформлення замовлення

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

Перший описує процес оформлення замовлення клієнта: від вибору типу замовлення та позицій меню до збереження сформованого замовлення у базі даних і передавання його на кухню для подальшої обробки.

Алгоритм створення замовлення описує шлях занесення його у вебсистему до появи замовлення на кухні. Спочатку користувач відкриває модуль замовлень, після чого система завантажує з сервера необхідні дані: список столів, офіціантів та меню. Коли дані відображені, користувач обирає тип замовлення. Якщо клієнт обідає в закладі, потрібно вибрати стіл. Якщо замовлення оформлюється «з собою», можна вказати ім'я клієнта.

Далі формується склад замовлення: додаються позиції меню, визначається їхня кількість. Система одразу оновлює проміжну вартість замовлення, щоб користувач бачив актуальну суму. Після цього працівник натискає кнопку створення замовлення, і інтерфейс надсилає сформовані дані на сервер.

На бекенді контролер перевіряє правильність отриманих даних: тип замовлення, наявність столу, список позицій та відповідність цих даних записам у базі. Якщо виявлено помилку, система повертає повідомлення, яке відображається користувачу. Якщо всі дані коректні, сервіс створює новий об'єкт замовлення, формує перелік позицій і зберігає це в базі даних.

Після успішного збереження сервер повертає інформацію про нове замовлення, яке автоматично відображається в модулі кухні. Інтерфейс оновлює список активних замовлень і готує форму до наступного введення. Таким чином алгоритм забезпечує повний цикл оформлення замовлення, роблячи процес швидким і зрозумілим для персоналу.

А також алгоритм авторизації показано на рисунку 2.4, за допомогою діаграми активності. Він перевіряє права доступу користувачів, що забезпечує коректний вхід до системи, створення та перевірку JWT-токена, а також визначення рівня доступу до функціональних модулів відповідно до ролі працівника.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

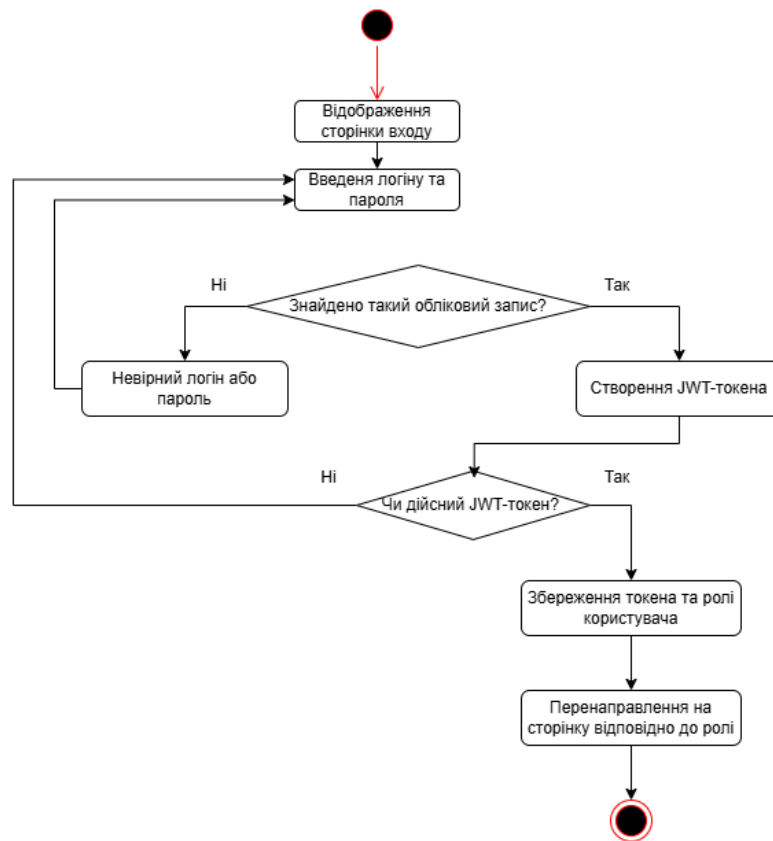


Рис. 2.4 – Діаграма активності авторизації

Алгоритм авторизації відповідає за те, щоб у систему могли увійти тільки ті користувачі, які мають відповідні права. Спочатку користувач переходить на сторінку входу, де вводить логін і пароль. Після натискання кнопки «Увійти» ці дані надсилаються на сервер.

Бекенд отримує запит і намагається знайти користувача в базі даних за введеним логіном. Якщо такого користувача немає або пароль не збігається, система повертає повідомлення про помилку, і користувач бачить, що авторизація неуспішна.

Якщо дані правильні, сервер створює JWT-токен, у якому зберігається інформація про самого користувача та його роль у системі. Цей токен повертається на фронтенд, де зберігається у локальному сховищі й додається до всіх подальших запитів.

Коли користувач намагається отримати доступ до захищених сторінок, сервер перевіряє токен: чи дійсний він, чи не прострочений, чи має користувач потрібну роль. Якщо перевірка успішна, запит виконується. Якщо ні, то

повертається статус 401 або 403, а користувач бачить повідомлення про відмову в доступі.

2.4. Реалізація функціоналу системи управління закладом харчування

Реалізація функціоналу системи управління закладом харчування базується на багаторівневій архітектурі, яка забезпечує чітке розмежування обов'язків між інтерфейсом користувача, бізнес-логікою та рівнем доступу до даних. У центрі вебзастосунку знаходиться Spring Boot-сервер, що відповідає за обробку запитів від клієнта, а React-клієнт забезпечує інтерактивну взаємодію з користувачем. Такий підхід відповідає сучасним вимогам до розробки вебсистем [3].

Внутрішня структура бекенду побудована відповідно: Controller–Service–Repository–Entity(Models) [24]. Контролери відповідають лише за приймання і повернення HTTP-запитів, працюючи лише з передачею даних, а не з їх опрацюванням. DTO-класи забезпечують розділення зовнішніх даних від внутрішніх сутностей, що сприяє підвищенню безпеки та гнучкості системи [25]. Репозиторії, реалізовані за допомогою Spring Data JPA [4], дозволяють взаємодіяти з MySQL без ручного написання SQL-запитів і використовують автоматичне формування методів за іменами.

Сутності у проекті відображають таблиці MySQL, а зв'язки між ними задаються через анотації @OneToMany, @ManyToOne та @ManyToMany. Для коректного оновлення або видалення пов'язаних записів застосовуються каскадні правила та @OnDelete. Перетворення сутностей у DTO та навпаки здійснює ModelMapper, що відповідає поширеному у веброзробці патерну «Mapper» [26] і дає змогу уникати дублювання коду та зменшувати залежність між шарами.

Фронтенд побудований на компонентному підході React [5]. Використання станів та React Hooks забезпечує швидке оновлення інформації без перезавантаження сторінки. Компоненти надсилають запити до серверу, отримують JSON-відповідь і відображають її у зручному для користувача вигляді.

Важливе місце у системі займає авторизація та контроль доступу. У системі застосовується Spring Security, який створює та перевіряє JWT-токени, визначає ролі користувачів і контролює, до яких ендпоінтів вони мають доступ [11]. Після

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

входу користувач отримує токен, у якому збережена інформація про його роль. Цей токен автоматично додається до заголовка Authorization під час кожного запиту. На рівні конфігурації визначено, які операції доступні адміністраторам, менеджерам, офіціантам та кухарям. Це гарантує, що кожен працівник бачить тільки ті частини системи, які належать до його обов'язків.

Для роботи із зображеннями страв реалізовано окремий конфігураційний клас FileStorageConfig. Він визначає місце збереження файлів, перевіряє коректність їх типу та створює унікальні назви під час завантаження. Такий підхід дозволяє зберігати фотографії у файловій системі, що є ефективнішим та зручнішим порівняно зі зберіганням їх у базі даних.

Модуль оформлення замовлення є одним із ключових елементів системи, оскільки він безпосередньо забезпечує роботу офіціантів та передачу замовлень на кухню. Після вибору типу замовлення, столу та позицій меню на фронтенді формується структура замовлення, яка надсилається на сервер. Контролер перевіряє правильність введених даних, а сервіс створює новий об'єкт замовлення та визначає його початковий статус. Після збереження всієї інформації у базі даних замовлення автоматично з'являється у модулі кухні, де кухарі одразу бачать список страв, що потребують приготування. Зміна статусів виконується сервісами й одразу відображається у фронтенді, що підтримує синхронну роботу залу та кухні.

Реалізація модуля керування меню охоплює створення, редагування та видалення категорій і позицій. Адміністратор може змінювати доступність страв, їхню ціну, опис або фотографію, а всі зміни миттєво фіксуються у базі даних та передаються на інтерфейс. Подібним чином працює модуль управління персоналом, де адміністратор може додавати нових працівників, змінювати їхні ролі та статуси, що одразу впливає на рівень доступу до функцій системи.

У модулі статистики сервер виконує необхідні запити й формує основні показники: кількість замовлень, суму продажів, найпопулярніші категорії або позиції. Отримані дані повертаються у форматі, зручному для подальшого відображення у вигляді таблиць чи графіків, що дозволяє адміністратору аналізувати роботу закладу.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

У проекті застосовано низку шаблонів проектування, що підвищують гнучкість та підтримуваність системи. Патерн «Repository» застосовується для інкапсуляції доступу до даних [27], а «DTO» і «Mapper» використовуються для розділення моделей та перетворення даних [26]. Принцип «Singleton» реалізовано у конфігураційних класах таких, як: ModelMapperConfig чи SecurityConfig [11]. Також принцип «Template Method» використано для уніфікації обробки помилок і відповідей контролерів. Завдяки цим підходам система залишається передбачуваною та легко розширюваною, а зміни в одному модулі не спричиняють небажаних змін в інших частинах застосунку.

Обрана архітектура забезпечує те, що кожен рівень системи має лише одну причину для модифікації. Контролери оновлюються тоді, коли змінюється REST-контракт, сервіси змінюються у випадку корекції бізнес-логіки, а репозиторії оновлюються тоді, коли змінюється структура моделі даних. Такий підхід відповідає принципам архітектури та спрощує тестування, розширення і подальшу підтримку проекту.

Висновки до другого розділу

У другому розділі дипломної роботи було виконано комплексне опрацювання структури та логіки роботи системи управління закладом харчування. Спочатку були визначені ролі користувачів і основні сценарії їх взаємодії із системою. За допомогою діаграми варіантів використання вдалося наочно показати ключові бізнес-процеси та сформулювати функціональні, нефункціональні та системні вимоги до програмного продукту.

Далі була побудована та реалізована база даних на основі MySQL, у якій інформацію поділено на логічно пов'язані таблиці для користувачів, персоналу, столів, категорій, позицій меню, замовлень і позицій замовлення. Налаштовані між таблицями зв'язки забезпечили цілісність даних і коректне виконання всіх операцій, що створило надійну основу для подальшого функціонування системи.

Також були розроблені алгоритми роботи системи, зокрема оформлення замовлення та авторизації користувача. Діаграми активності допомогли визначити послідовність дій і врахувати можливі помилки ще до реалізації. На основі

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

розроблених алгоритмів була створена багаторівнева архітектура, що включає фронтенд на React, бекенд на Spring Boot і базу даних MySQL. Реалізовано розмежування доступу за ролями та основні модулі системи, такі як: замовлення, меню, персонал і статистика.

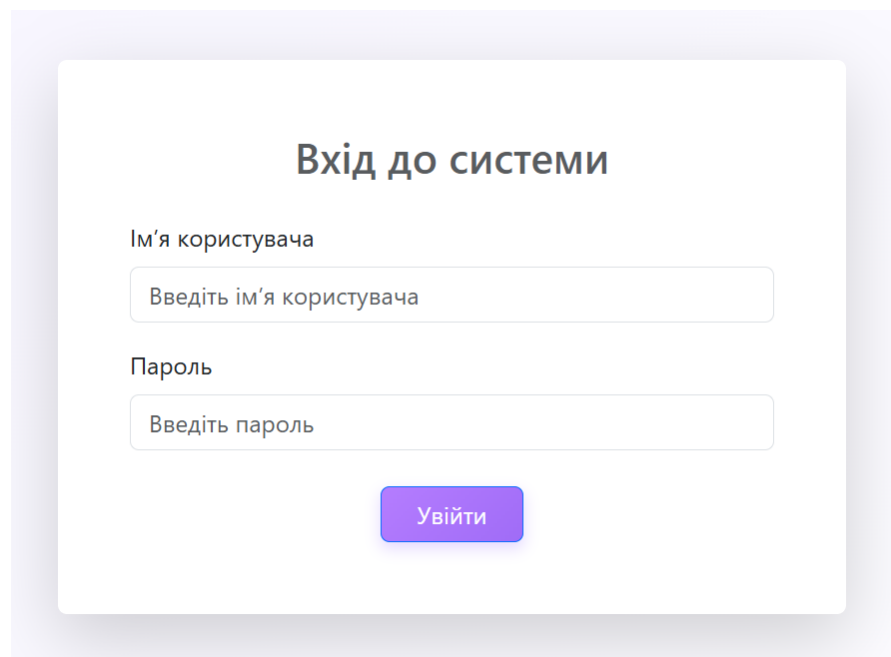
		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИТЕМОЮ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ

3.1. Структура інтерфейсу вебсистеми управління закладом харчування

Інтерфейс розробленої системи побудований таким чином, щоб забезпечити зручну роботу персоналу та швидкий доступ до основних функцій. Усі елементи оформлені в одному стилі, що спрощує навігацію та робить взаємодію зі сторінками інтуїтивно зрозумілою. Кожен модуль системи продуманий так, щоб користувач міг виконувати свої завдання з мінімальними витратами часу та без зайвих переходів між вкладками.

На рис. 3.1 показано стартове вікно авторизації, яке з'являється перед початком роботи користувача із системою.



Вхід до системи

Ім'я користувача

Введіть ім'я користувача

Пароль

Введіть пароль

Увійти

Рис. 3.1 – Вікно авторизації користувача

На даному екрані користувач вводить свої дані – логін користувача та пароль. Це відкриває доступ до системи, а логін перевіряється на бекенді через Spring Boot. Після натискання кнопки «Увійти» система надсилає запит до серверу, який перевіряє правильність введених даних та формує JWT-токен. Якщо авторизація успішна, користувача перенаправляє на головну панель згідно з його роллю (офіціант, адміністратор, кухар або менеджер). У разі помилки система показує повідомлення про неправильний логін або пароль.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

На рис. 3.2–3.5 показано те, як система автоматично формує інтерфейс відповідно до ролі авторизованого користувача. Після входу в систему сервер повертає JWT-токен, у якому зашифрована інформація про роль. На основі цих даних React будує ліву панель навігації та визначає, які розділи і функції повинні бути доступними саме в цьому обліковому записі. Такий підхід значно спрощує роботу персоналу, оскільки кожен бачить тільки ті інструменти, які йому дійсно потрібні.

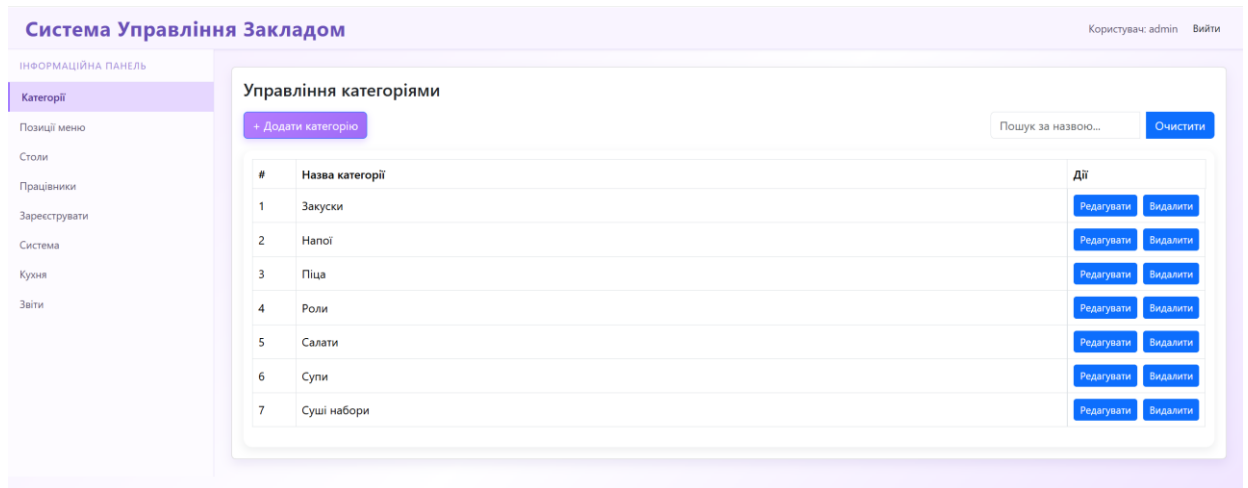


Рис. 3.2 – Панель адміністратора

Панель адміністратора містить повний набір можливостей: робота з категоріями та стравами, управління столами, персоналом, реєстрація нових профілів, замовлення, модуль кухні та звіти. Адміністратор має доступ до всіх функцій, тому його інтерфейс є найбільш розширеним.

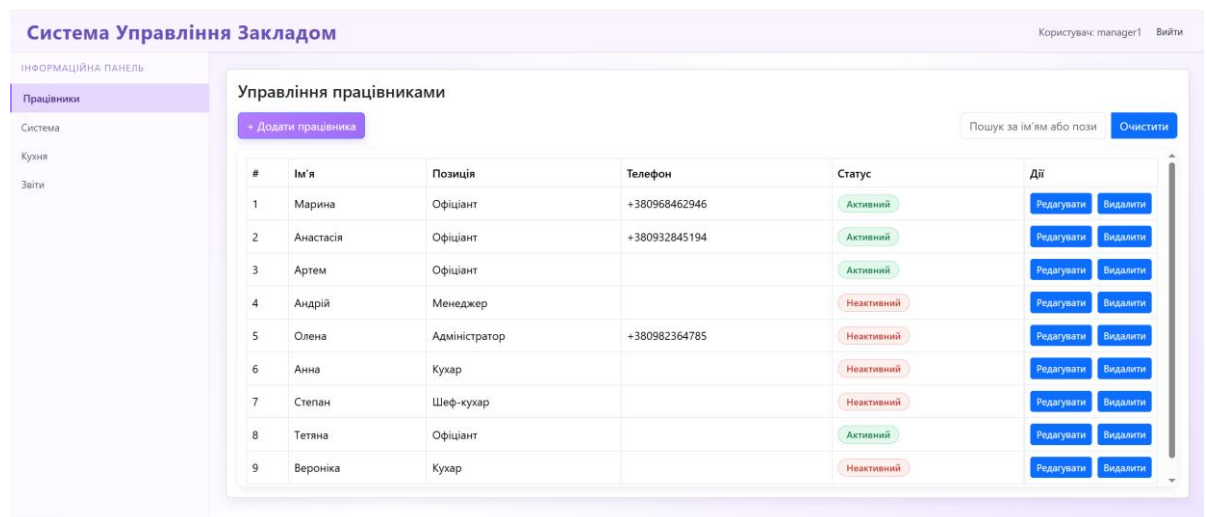


Рис. 3.3 – Панель менеджера

У менеджера можливостей дещо менше, ніж у адміністратора: система приховує ті вкладки, які безпосередньо не стосуються його обов’язків. Менеджер працює з персоналом, системою замовлень, кухнею та звітами, але не має доступу до системних налаштувань.

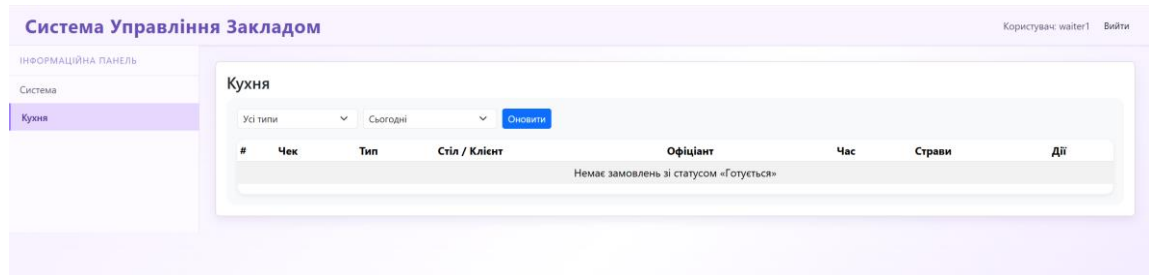


Рис. 3.4 – Панель офіціанта

На рис. 3.4 наведено інтерфейс офіціанта, тут панель спрощена. Залишилися лише ті розділи, що пов’язані зі створенням замовлень і передачею їх на кухню. Така структура дозволяє офіціантам швидко орієнтуватися та виконувати свою роботу без зайвих дій.

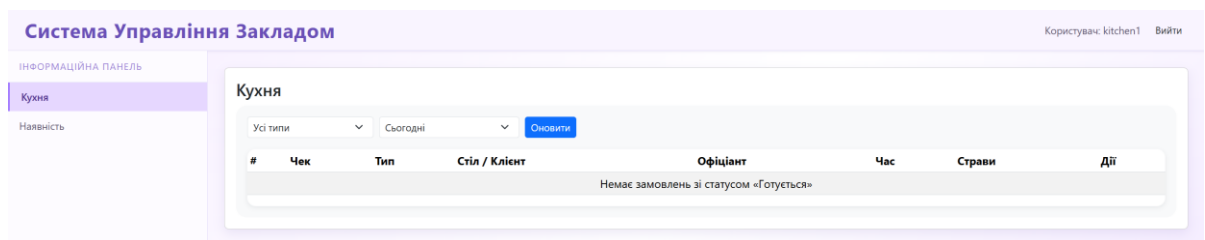


Рис. 3.5 – Панель кухні

Кухня бачить тільки модуль «Кухня» та, розділ «Наявність продуктів». Це дозволяє зосередитися виключно на приготуванні страв та контролі замовлень. Усі замовлення автоматично відображаються залежно від їхнього статусу, тому працівник кухні працює з актуальною інформацією.

На рис. 3.6 продемонстровано вкладку «Управління категоріями». У центральній частині сторінки розміщена таблиця з усіма категоріями меню: «Закуси», «Напої», «Піца», «Роли» та інші. Для кожної категорії доступні дії «Редагувати» та «Видалити». Через кнопку «+ Додати категорію» можна створити нову категорію. Пошук за назвою дає можливість швидко знаходити потрібну категорію.

Управління категоріями

+ Додати категорію

Пошук за назвою...

Очистити

#	Назва категорії	Дії
1	Закуси	Редагувати Видалити
2	Напої	Редагувати Видалити
3	Піца	Редагувати Видалити
4	Роли	Редагувати Видалити
5	Салати	Редагувати Видалити
6	Супи	Редагувати Видалити
7	Суші набори	Редагувати Видалити

Рис. 3.6 – Вкладка «Управління категоріями»

На рис. 3.7 подано форму створення нової категорії, яка відкривається після натискання відповідної кнопки. Користувач вводить назву нової категорії та зберігає її. На бекенді відбувається перевірка на дублювання, після чого новий запис додається до бази даних. Інтерфейс автоматично оновлюється та показує нову категорію в загальному списку.

Додати категорію

Назва категорії

Введіть назву...

Відмінити

Зберегти

Рис. 3.7 – Додати категорію

На рис. 3.8 наведено вікно редагування категорії, де користувач може змінити її назву. Система завантажує актуальні дані існуючої категорії в текстове поле. Після внесення змін і натискання кнопки «Зберегти». Якщо оновлення успішне, нова назва відображається у таблиці.

Редагувати категорію

Назва категорії

Салати

Відмінити

Зберегти

Рис. 3.8 – Редагування категорії

На рис. 3.9 показано інтерфейс, де адміністратор може переглядати та редагувати всі наявні позиції меню закладу. На сторінці у вигляді таблиці відображаються всі страви та напої. Для кожної позиції зазначено назву, до якої

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

категорії вона належить, а також поточний статус «в наявності» чи ні. Це дозволяє швидко оцінити актуальність меню. Праворуч розміщені кнопки «Редагувати» та «Видалити», що дають можливість оновити або видалити дані без перезавантаження сторінки. Кнопка «+ Додати позицію» відкриває форму, де можна створити нову страву з усіма характеристиками. Доступний також пошук, який допомагає швидко знайти потрібну позицію серед великого списку та фільтр для зазначення скільки позицій відображати.

Управління позиціями меню

+ Додати позицію

15 Пошук за назвою Очистити

#	Назва	Категорія	Статус	Дії
1	Сільвана	Піца	В наявності	Редагувати Видалити
2	Зелений дракон	Роли	В наявності	Редагувати Видалити
3	Філадельфія	Роли	В наявності	Редагувати Видалити
4	Вершковий вугор	Роли	В наявності	Редагувати Видалити
5	Пепсі	Напої	В наявності	Редагувати Видалити
6	Грильята	Піца	В наявності	Редагувати Видалити
7	Дьябола	Піца	В наявності	Редагувати Видалити
8	Моршинська негазована	Напої	В наявності	Редагувати Видалити
9	Сік	Напої	В наявності	Редагувати Видалити

Рис. 3.9 – Вкладка «Управління позиціями меню»

На рис. 3.10–3.11 наведено інтерфейс створення нової позиції меню та приклад заповненої форми. У формі адміністратор вказує основні характеристики страви: назву, категорію та стан доступності у меню. Окремий блок призначений для додавання розмірів і відповідних цін, що дає змогу створювати кілька варіантів однієї позиції. Нижче розміщено секцію для завантаження зображення. Система дозволяє вибрати файл із комп'ютера і також можна вставити URL уже наявної картинки. На рис. 3.11, видно, як форма виглядає після заповнення реальними даними: введено назву страви, обрано категорію, додано розміри та ціни, а також завантажено фото, яке автоматично відображається у попередньому перегляді. Це значно полегшує роботу адміністратора, адже дає можливість одразу побачити, як позиція виглядатиме у системі. Після збереження всі дані передаються на сервер, проходять перевірку та зберігаються у базі даних, що забезпечує швидке й зручне оновлення меню закладу.

Додати позицію

Назва

Введіть назву позиції

Категорія

Оберіть категорію

В наявності

Розміри та ціни

Розмір (наприклад, 1 шт, 0.5 л)

Ціна

+

Картинка

Вибрати файл

Файл не вибрано

URL зображення

Відмінити

Зберегти

Рис. 3.10 – Додати позицію

Додати позицію

Назва

Салат Каліфорнія

Категорія

Салати

В наявності

Розміри та ціни

Розмір (наприклад, 1 шт, 0.5 л)

Ціна

+

Розмір	Ціна	Дії
170г	250	Відмінити

Картинка

Вибрати файл

saladCalifornia.png

/uploads/products/df3ccc4a-6be1-4f9a-ae78-1611516fa35b.png

Попередній перегляд:



Відмінити

Зберегти

Рис. 3.11 – Заровнена форма «Додати позицію»

На рис. 3.12 показано форму редагування позиції, яка відкривається після натискання кнопки. У формі автоматично підтягуються всі поточні дані: назва страви, категорія, доступність, розміри з цінами та прикріплене зображення. Користувач може змінити будь-який параметр. Наприклад, оновити ціну, додати новий розмір або замінити фотографію. Після натискання кнопки «Зберегти» оновлюється інформація цієї позиції і одразу відображається у загальному списку позицій.

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Редагувати позицію

Назва

Вершковий вугор

Категорія

Роли

В наявності

Розміри та ціни

Розмір (наприклад, 1 шт, 0.5 л)

Ціна

+

Розмір	Ціна	Дії
280г	239	Видалити

Картинка

Вибрати файл

Файл не вибрано

/uploads/products/fdaddc91-9ad5-4fc7-a27c-17872249eb8e.png

Попередній перегляд:

Відмінити

Зберегти

Рис. 3.12 – Редагувати позицію

На рис. 3.13 подано сторінку для роботи зі столами у залі закладу. У цьому розділі адміністратор може додавати нові столи або видаляти ті, що більше не використовуються. Таблиця містить нумерацію та назву кожного столу, а в колонці «Дії» передбачена можливість видалення. Таке управління потрібне для правильної роботи офіціантів, адже під час створення замовлення система має знати, які столи доступні.

Управління столами

+ Додати стіл

#	Назва столу	Дії
1	Стіл 1	Видалити
2	Стіл 10	Видалити
3	Стіл 2	Видалити
4	Стіл 3	Видалити
5	Стіл 4	Видалити
6	Стіл 5	Видалити
7	Стіл 6	Видалити
8	Стіл 7	Видалити
9	Стіл 8	Видалити

Рис. 3.13 – Вкладка «Управління столами»

На рис. 3.14 представлено форму, яка використовується для створення нового столу. Достатньо ввести назву і після натискання кнопки «Зберегти» новий запис

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

одразу з'являється у списку. Система автоматично перевіряє, щоб не було дублювання номерів столів.

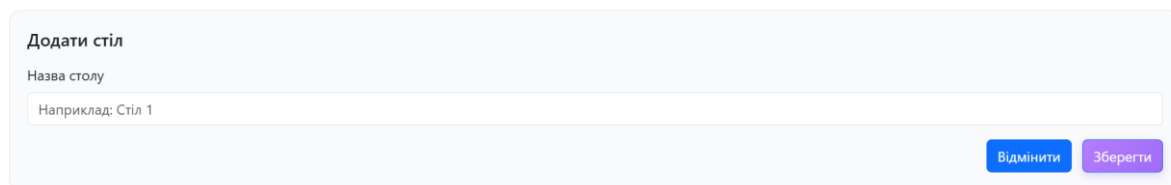
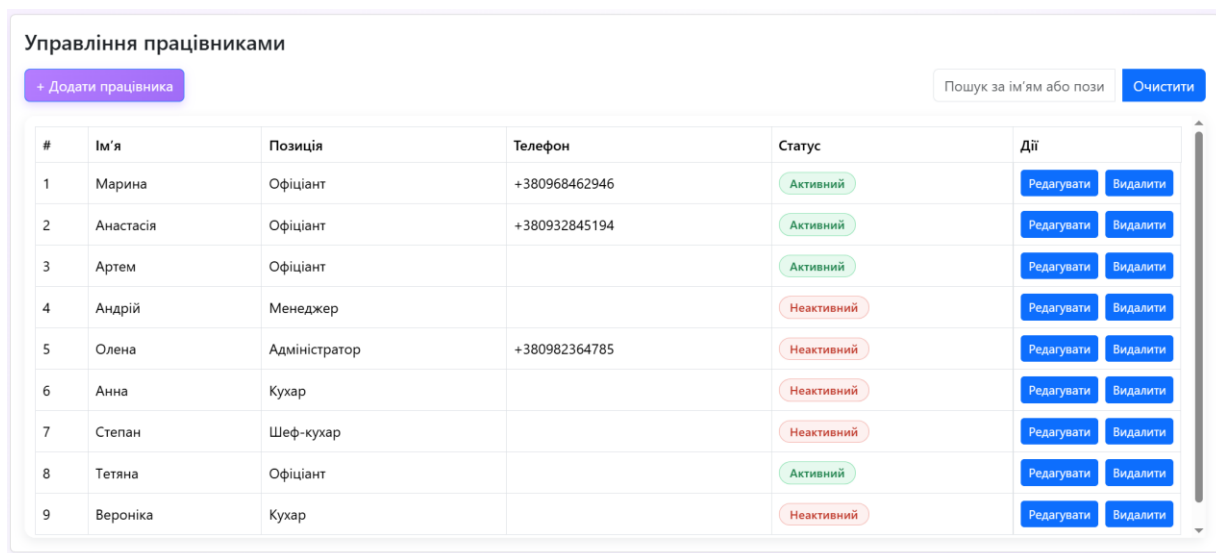


Рис. 3.14 – Додати стіл

На рис. 3.15 показано інтерфейс управління персоналом, який доступний адміністратору та менеджеру. У таблиці відображено основні дані: ім'я, посада, телефон та статус активності. Кольорові позначки, а саме зелений означає активний, особливо це важливо для офіціантів, щоб бачити хто на зміні, а червоний це неактивний. Кожен запис можна відредагувати або видалити. Через кнопку «+ Додати працівника» відкривається форма створення нового облікового запису.



#	Ім'я	Позиція	Телефон	Статус	Дії
1	Марина	Офіціант	+380968462946	Активний	Редагувати Видалити
2	Анастасія	Офіціант	+380932845194	Активний	Редагувати Видалити
3	Артем	Офіціант		Активний	Редагувати Видалити
4	Андрій	Менеджер		Неактивний	Редагувати Видалити
5	Олена	Адміністратор	+380982364785	Неактивний	Редагувати Видалити
6	Анна	Кухар		Неактивний	Редагувати Видалити
7	Степан	Шеф-кухар		Неактивний	Редагувати Видалити
8	Тетяна	Офіціант		Активний	Редагувати Видалити
9	Вероніка	Кухар		Неактивний	Редагувати Видалити

Рис. 3.15 – Вкладка «Управління працівниками»

На рис. 3.16–3.17 показано інтерфейси додавання нового працівника та редагування даних уже існуючого співробітника.

У формі створення працівника необхідно ввести основну інформацію: ім'я, посаду та номер телефону. Також передбачений перемикач «Активний», який визначає, чиє він на зміні чи ні. Після натискання «Зберегти» формується новий запис. На прикладі, наведеному на рис. 3.17, видно, як виглядає форма редагування. Усі дані існуючого працівника автоматично завантажуються у відповідні поля, що дозволяє швидко оновити інформацію.

Додати працівника

Ім'я

Введіть ім'я

Позиція

Почніть вводити позицію...

Номер телефону

+380...

☒ Активний

Відмінити

Зберегти

Рис. 3.16 – Додати працівника

Редагувати працівника

Ім'я

Анна

Позиція

Кухар

Номер телефону

+380...

☐ Активний

Відмінити

Зберегти

Рис. 3.17 – Редагувати працівника

Реєстрація нового користувача

Ім'я користувача

Введіть ім'я користувача

Пароль

Введіть пароль мінімум на 5 символів

Роль

Менеджер

Перегляд облікових записів

Очистити

Зареєструвати

Рис. 3.18 – Вкладка «Реєстрація нового користувача»

На рис. 3.18 представлено інтерфейс, який дозволяє адміністратору створювати нові облікові записи. У формі потрібно ввести ім'я користувача, пароль та вибрати роль, яка буде присвоєна цьому профілю. Доступна також кнопка «Перегляд облікових записів», що відкриває список усіх профілів. Після натискання кнопки «Зареєструвати» створюється запис у базі даних і генерується можливість авторизації.

На рис. 3.19–3.20 наведено вікно зі списком облікових записів системи та інтерфейс їх редагування. У вікні, зображеному на рис. 3.19, відображається

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

перелік усіх існуючих облікових записів із зазначенням імені користувача та ролі, яку він виконує у системі. Завдяки наявним кнопкам «Редагувати» та «Видалити» адміністратор може швидко змінювати дані або видаляти облікові записи, які більше не використовуються. На рис. 3.20 продемонстровано, як виглядає форма редагування даних вибраного користувача. Поля форми автоматично заповнюються актуальною інформацією і адміністратор за потреби може змінити роль користувача, встановити новий пароль або залишивши поле порожнім, якщо оновлення пароля не потрібно. І в кінці обов’язково треба зберегти зміни.

Облікові записи користувачів					✕	
#	Ім'я користувача	Роль	Дії			
1	admin	ROLE_ADMIN	Редагувати	Видалити		
2	waiter1	ROLE_WAITER	Редагувати	Видалити		
3	manager1	ROLE_MANAGER	Редагувати	Видалити		
4	manager2	ROLE_MANAGER	Редагувати	Видалити		
5	kitchen1	ROLE_KITCHEN	Редагувати	Видалити		

Рис. 3.19 – Облікові записи користувачів

Редагування користувача: **manager1**

Ім'я користувача

Роль

Менеджер

Новий пароль

Якщо ви не хочете змінювати пароль, залиште поле порожнім.

Скасувати

Зберегти

Рис. 3.20 – Редагування користувача

На рис. 3.21 продемонстровано основну панель створення замовлень. Ліва частина містить список категорій меню. У центрі відображаються позиції обраної категорії з цінами та фото. Праворуч розташована форма замовлення, де офіціант обирає стіл, вказує своє ім'я та переглядає список страв, доданих до чека. Поки

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

жодної позиції не вибрано, відображається напис «Чек порожній». Це дозволяє швидко формувати замовлення та передавати його на кухню.

Замовлення

На місці З собою Передати на кухню Список чеків Вийти

Усі Закуски Напої Піца Роли Салати Супи Суші набори

Пошук за назвою... Очистити

Сільвана	Зелений дракон	Філадельфія	Вершковий вугор
560г (215 грн)	280г (290 грн)	280г (265 грн)	280г (239 грн)
Пепсі	Грильята	Дьябола	Моршинська негазована
500мл (55 грн) 1.25л (90 грн)	560г (250 грн)	535г (215 грн)	500мл (42 грн)

Сума: 0.00 грн Швидко оплата Готівка

Рис. 3.21 – Вкладка «Замовлення»

На рис. 3.22 показано фільтрацію позицій меню за категорією. Після вибору категорії «Піца» система показує тільки ті страви, що належать до цієї групи. Це спрощує роботу офіціанта і він бачить лише потрібні позиції та може швидко додати їх у замовлення. Функція пошуку допомагає миттєво знайти страву за назвою.

Замовлення

На місці З собою Передати на кухню Список чеків Вийти

Усі Закуски Напої **Піца** Роли Салати Супи Суші набори

Пошук за назвою... Очистити

Сільвана	Грильята	Дьябола
560г (215 грн)	560г (250 грн)	535г (215 грн)

Сума: 0.00 грн Швидко оплата Готівка

Рис. 3.22 – Категорії «Піца»

На рис. 3.23 показано, як виглядає інтерфейс створення замовлення «На місці». Після натискання на позицію вона автоматично додається до правої частини сторінки, де формується чек. У наведеному прикладі, коли це замовлення «На місці» офіціант обов'язково обирає стіл, а в полі «Офіціант» вказується офіціанта, який за нього відповідальний. Система включає до чека всі вибрані страви та одразу показує їхню назву, розмір, кількість, ціну за одиницю та суму за позицію. Кількість можна змінювати за допомогою кнопок збільшення або зменшення, а

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

зайві позиції легко видалити. Під таблицею система автоматично підраховує загальну суму замовлення. У нижній частині розміщено кнопки для оформлення оплати: «Швидка оплата» або «Готівка», що дозволяє завершити створення замовлення за кілька секунд.

Замовлення

На місці З собою Передати на кухню Список чеків

Усі Закуски Напої Піца Роли Салати Супи Суші набори

Пошук за назвою... Очистити

Сільвана 560г (215 грн)

Зелений дракон 280г (290 грн)

Філадельфія 280г (265 грн)

Вершковий вугор 280г (239 грн)

Пепсі 500мл (55 грн) 1.25л (90 грн)

Грильята 560г (250 грн)

Дьябола 535г (215 грн)

Моршинська негазована 500мл (42 грн)

Стил Стил 3 Офіціант Марина (Офіціант)

#	Назва	К-сть	Ціна	Сума
1	Зелений дракон 280г	1	290	290.00
2	Грильята 560г	1	250	250.00
3	Пепсі 500мл	2	55	110.00

Сума: 650.00 грн Швидка оплата Готівка

Рис. 3.23 – Замовлення «На місці»

Замовлення

На місці З собою Передати на кухню Список чеків

Усі Закуски Напої Піца Роли Салати Супи Суші набори

Пошук за назвою... Очистити

Пепсі 500мл (55 грн) 1.25л (90 грн)

Грильята 560г (250 грн)

Дьябола 535г (215 грн)

Моршинська негазована 500мл (42 грн)

Сік 950мл (140 грн)

Теплий сет 850г 24шм (579 грн)

Запечений сет 1100г 32шм (810 грн)

Старт сет 1400г 40шм (1100 грн)

Клієнт Марія

#	Назва	К-сть	Ціна	Сума
1	Теплий сет 850г 24шм	1	579	579.00
2	Сік 950мл	1	140	140.00

Сума: 719.00 грн Швидка оплата Готівка

Рис. 3.24 – Замовлення «З собою»

На рис. 3.24 наведено заповненого замовлення «З собою». У правій панелі необхідно вказати інформацію про клієнта, наприклад ім'я, після чого відображаються страви, які офіціант додав у чек, разом з їхніми кількостями та сумою. У наведеному прикладі формується замовлення «З собою», тому система автоматично приховує поле вибору столу та орієнтується лише на дані клієнта. Такий підхід дає можливість офіціанту швидко створювати замовлення на виніс, не

виконуючи зайвих дій, і забезпечує коректне формування чека саме для цього типу замовлень.

На рис. 3.25 зображено модальне вікно зі списком створених замовлень та їх чеків. Тут можна переглянути всі чеки за певний період, застосувати фільтри за типом, статусом і порядком відображення. У таблиці вказано номер замовлення, тип, стіл, офіціанта, клієнта час та статус. Кнопки дій дозволяють відкрити чек, роздрукувати його або виконати інші операції.

Список чеків ×

Усі ▼
Усі статуси ▼
Нові зверху ▼
Оновити

#	Номер замовлення	Стіл	Офіціант	Клієнт	Час	Тип	Статус	Дії
1	#E53C7201	-	-	Марія	11.12.2025 01:30	3 собою	Готується	+ ✎ 🖨
2	#8CC16227	Стіл 3	Марина (Офіціант)	-	11.12.2025 01:29	На місці	Готується	+ ✎ 🖨

Рис. 3.25 – Вікно «Список чеків»

Кухня

Усі типи ▼
Сьогодні ▼
Оновити

#	Чек	Тип	Стіл / Клієнт	Офіціант	Час	Страви	Дії
1	#8CC16227 Готується	На місці	Стіл 3	Марина (Офіціант)	01:29	- Зелений дракон (280г) × 1 - Грильята (560г) × 1 - Пепсі (500мл) × 2	Надрукувати Виконано
2	#E53C7201 Готується	3 собою	Марія	-	01:30	- Теплий сет (850г 24шм) × 1 - Сік (950мл) × 1	Надрукувати Виконано

Рис. 3.26 – Вкладка «Кухня»

На рис. 3.26 наведено інтерфейс для кухонної панелі, де відображаються замовлення зі статусом «Готується». У списку зазначені всі необхідні дані: номер чека, тип замовлення, стіл чи клієнт, офіціант, час та перелік страв. Кухар може відмітити замовлення як виконане або надрукувати його для роботи. Оновлення відбувається в реальному часі, що забезпечує синхронність між офіціантами та кухнею.

На рис. 3.27 продемонстровано редагування замовлення, коли додаються ще позиції. У правій панелі вказано обраний стіл і офіціанта, які вже закріплені за відповідним замовленням і їх змінити не можна. Додаються нові позиції, в той самий час змінюється загальна вартість замовлення і після завершення усіх дій обов'язково потрібно передати на кухню.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Замовлення

На місці
З собою
Передати на кухню
Список чеків

Усі

Закуси

Напої

Піца

Роли

Салати

Супи

Суші набори

Пошук за назвою...

Очистити

Сік

950мл (140 грн)

Теплий сет

850г 24шт (579 грн)

Запечений сет

1100г 32шт (810 грн)

Старт сет

1400г 40шт (1100 грн)

Том ям

350г (230 грн)

Борщ

350г (169 грн)

Суп удон

400г (160 грн)

Грецький салат

250г (250 грн)

Тайський салат

220г (280 грн)

Салат з лососем

150г (185 грн)

Крильця по азійськи

150г (185 грн)

Годза з куркою

100г (200 грн)

Вийти

Стіл

Стіл 3

Офіціант

Марина (Офіціант)

#	Назва	К-сть	Ціна	Сума	
1	Зелений дракон 280г	1	290	290.00	✕
2	Грильята 560г	1	250	250.00	✕
3	Пепсі 500мл	2	55	110.00	✕
4	Салат з лососем 150г	1	185	185.00	✕

Сума: 835.00 грн (сума вибраного чека)

Швидка оплата Готівка

Рис. 3.27 – Редагування замовлення

На рис. 3.28 наведено вигляд друкованої версії чека, який формується для кухні. У документі автоматично зазначаються дата, час, номер замовлення, стіл та ім'я офіціанта, що оформляв замовлення. Далі подано список страв з урахуванням розміру та кількості. Цей чек використовується для роботи кухні, що дозволяє підтримувати чіткий облік замовлень та уникати помилок у комунікації.

Чек №8CC16227

Дата: 11.12.2025

Час: 01:29

Стіл: Стіл 3

Офіціант: Марина (Офіціант)

#	Назва	К-сть
1	Зелений дракон (280г)	1
2	Грильята (560г)	1
3	Пепсі (500мл)	2
4	Салат з лососем (150г)	1

Рис. 3.28 – Чек на кухні

На рис. 3.29 показано модальне вікно зі списком чеків зі статусом «Виконано». У таблиці видно, що обидва замовлення змінили свій статус на «Виконано» це означає, що їх вже віднесли клієнтові та вони вже готові до оплати.

Список чеків

Усі

Усі статуси

Нові зверху

Оновити

#	Номер замовлення	Стіл	Офіціант	Клієнт	Час	Тип	Статус	Дії
1	#E53C7201	-	-	Марія	11.12.2025 01:30	З собою	Виконано	✚ ✎ 🖨
2	#8CC16227	Стіл 3	Марина (Офіціант)	-	11.12.2025 01:29	На місці	Виконано	✚ ✎ 🖨

Рис. 3.29 – Статус «Виконано»

На рис. 3.30–3.31 зображено процес оформлення оплати готівкою та повідомлення про її успішне завершення. У модальному вікні, зображеному на рис. 3.30, відображається сума, яку потрібно сплатити, а також поле для введення коштів, отриманих від клієнта. Після введення суми система автоматично розраховує решту та пропонує підтвердити оплату, що робить процес готівкового розрахунку простим, наочним та мінімізує можливість помилки. Поруч із цим передбачена і кнопка «Швидка оплата», яка дозволяє завершити операцію миттєво, еоли наприклад це оплата карткою. На рис. 3.31 видно повідомлення, яке з’являється після успішного підтвердження: система відображає суму замовлення, введені клієнтом кошти та розраховану решту, що підтверджує коректність проведеної операції. Після завершення оплати чек автоматично закривається.

Оплата готівкою

Сума до сплати: **719.00 грн**

Отримано від клієнта (грн)

750

Скасувати

Підтвердити оплату

Рис. 3.30 – Оплата готівкою

Замовлення

Оплата готівкою успішна. Сума: 719.00 грн, отримано: 750.00 грн, решта: 31.00 грн.

На місці

З собою

Передати на кухню

Список чеків

Вийти

Усі

Закуси

Напої

Піца

Роли

Салати

Суші

Суші набори

Пошук за назвою...

Очистити

Сільвана

560г (215 грн)

Зелений дракон

280г (290 грн)

Філадельфія

280г (285 грн)

Вершковий вугор

280г (239 грн)

PEPER

Кришніс

Клієнт

Ім'я клієнта...

#

Назва

К-сть

Ціна

Сума

Чек порожній

Сума: 0.00 грн

Швидка оплата

Готівка

Рис. 3.31 – Успішна оплата готівкою

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

На рис. 3.32 показано, як змінюється список чеків після завершення оплати. У таблиці поруч із замовленнями з'являється статус «Сплачено», що дає змогу легко відстежувати завершені замовлення.

Список чеків ×

Усі ▾
Усі статуси ▾
Нові зверху ▾
Оновити

#	Номер замовлення	Стіл	Офіціант	Клієнт	Час	Тип	Статус	Дії
1	#E53C7201	-	-	Марія	11.12.2025 01:30	З собою	Сплачено	+ ✎ 🖨
2	#8CC16227	Стіл 3	Марина (Офіціант)	-	11.12.2025 01:29	На місці	Сплачено	+ ✎ 🖨

Рис. 3.32 – Статус «Сплачено»

На рис. 3.33 наведено друковану версію чека для клієнта. На відміну від замовлень «На місці», у чеку зазначається ім'я клієнта замість столу. Вказано точний час та список страв з кількістю і сумою. Чек розрахований на видачу клієнту, тому містить всю необхідну інформацію для підтвердження покупки.

Чек №E53C7201

Дата: 11.12.2025

Час: 01:30

Тип замовлення: З собою

Клієнт: Марія

#	Назва	К-сть	Ціна	Сума
1	Теплий сет (850г 24шм)	1	579.00	579.00
2	Сік (950мл)	1	140.00	140.00
Разом:				719.00 грн

Рис. 3.33 – Чек

На рис. 3.34 показано вкладку «Наявність продуктів», яка дозволяє контролювати, чи доступна та чи інша позиція меню. У таблиці подано назви продуктів та їхній статус. Якщо певної страви тимчасово немає, кухня контролює та може змінити її доступність, і вона автоматично зникне зі списку при створенні замовлення. Це допомагає уникати ситуацій, коли офіціант додає у чек страву, якої немає на кухні.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Наявність продуктів			
20	Пошук за назвою	Очистити	
#	Назва	Статус	Дії
1	Сільвана	В наявності	
2	Зелений дракон	В наявності	
3	Філадельфія	В наявності	
4	Вершковий вугор	В наявності	
5	Пепсі	В наявності	
6	Грильята	В наявності	
7	Дьябола	В наявності	
8	Моршинська негазована	В наявності	
9	Сік	В наявності	

Рис. 3.34 – Вкладка «Наявність продуктів»

На рис. 3.35 наведено форму редагування наявності для окремої позиції. У формі відображається назва продукта та перемикач «Є в наявності». Кухня може швидко змінити статус, після чого зміни одразу застосовуються у всій системі.

Редагувати наявність

Назва

Філадельфія

☒ Є в наявності

Відмінити Зберегти

Рис. 3.35 – Редагувати наявність

3.2. Статистика продажів та загальна аналітика системи

Для ефективного управління закладом харчування важливо не лише приймати замовлення, а й мати можливість аналізувати результати роботи за певний період. У розробленій системі реалізовано модуль звітності, який дозволяє переглядати динаміку продажів, кількість оформлених замовлень, суму виручки, а також популярність окремих страв чи категорій меню. Такий підхід дає змогу керівництву швидко оцінювати завантаженість закладу, визначати найприбутковіші напрямки. Усі звіти формуються автоматично на основі даних із бази MySQL та відображаються у наочному вигляді за допомогою таблиць і графіків.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

На рис. 3.36 зображено головне вікно модуля аналітики, де можна обрати період та отримати зведені дані про продажі. У верхній частині розташовані два поля: початкова та кінцева дата, користувач задає потрібний інтервал, якщо довго залишається на сторінці то натискає кнопку «Оновити звіт» для оновлення даних. Нижче розміщені три вкладки, які дозволяють переглядати продажі у різних напрямках: за категоріями, за окремими позиціями меню та у вигляді загального зведеного звіту. У центрі сторінки відображено графік загальної суми продажів по днях, що допомагає оцінити динаміку роботи закладу та виявити найбільш чи найменш активні дні.

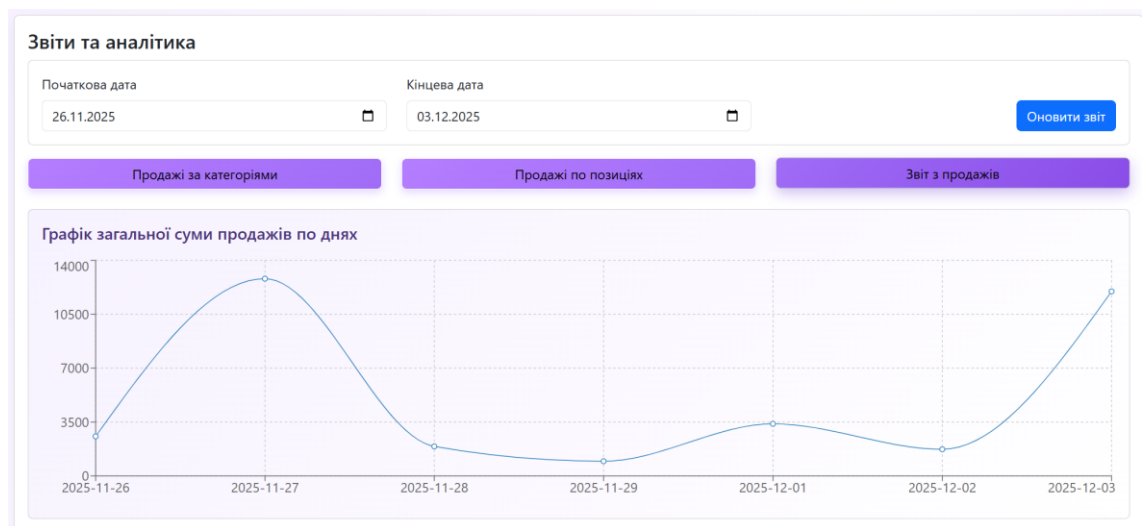


Рис. 3.36 – Вкладка «Звіти та аналітика»

На рис. 3.37 наведено детальну таблицю, яка відображає кількість замовлень і суму виручки по кожному дню обраного періоду. Для кожної дати система показує два основні показники: скільки замовлень було створено та на яку суму вони становили. Такий формат дозволяє швидко оцінити рівень активності закладу та порівнювати результати між різними днями. Окрема кнопка «Надрукувати звіт» дає можливість отримати паперову копію документа.

Звіт з продажів				Надрукувати звіт
#	Дата	Кількість замовлень	Сума, грн	
1	2025-11-26	3	2570	
2	2025-11-27	14	12809	
3	2025-11-28	2	1920	
4	2025-11-29	1	957	
5	2025-12-01	2	3386	
6	2025-12-02	1	1733	
7	2025-12-03	11	11982	

Рис. 3.37 – Звіт з продажів

На рис. 3.38 продемонстровано друковану версію звіту, яку формує система на основі вибраного періоду. У звіті міститься таблиця з розподілом продажів по днях, а також підсумкова інформація, а саме загальна кількість замовлень та загальна сума виручки за період. Друкована форма дозволяє вести архів звітів та переглядати динаміку продажів за тривалий час.

Звіт з продажів			
Період: 26.11.2025 — 03.12.2025			
№	Дата	Кількість замовлень	Сума, грн
1	2025-11-26	3	2570
2	2025-11-27	14	12809
3	2025-11-28	2	1920
4	2025-11-29	1	957
5	2025-12-01	2	3386
6	2025-12-02	1	1733
7	2025-12-03	11	11982

Всього замовлень: **34**
 Загальна сума: **35357** грн

Рис. 3.38 – Друкована версія «Звіт з продажів»

На рис. 3.39 зображено таблицю, у якій відображається статистика продажів кожної окремої страви меню. У цьому звіті зазначається назва продукту, його розмір, кількість проданих одиниць та загальна сума, яку заклад отримав від цієї позиції. Такий формат дає можливість визначити найпопулярніші та найменш затребувані страви, що допомагає коригувати меню, оптимізувати закупівлі та планувати витрати.

Продажі по позиціях меню					Надрукувати звіт
#	Назва продукту	Розмір	Кількість	Сума, грн	
1	Теплий сет	850г 24шм	2	1158	
2	Зелений дракон	280г	1	290	
3	Грильята	560г	1	250	
4	Вершковий вугор	280г	1	239	
5	Пепсі	500мл	4	220	
6	Гюдза з куркою	100г	1	200	
7	Салат з лососем	150г	1	185	
8	Суп удон	400г	1	160	
9	Сік	950мл	1	140	
10	Страва	123	1	123	
11	Сирні палички	150г	1	112	

Рис. 3.39 – Продажі по позиціях меню

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

На рис. 3.40 показано аналітику продажів, згруповану за категоріями меню. У верхній частині розміщено кругову діаграму, яка наочно демонструє частку кожної категорії в загальному обсязі продажів. Нижче подано таблицю, де відображені категорії, кількість продажів та сума виручки по кожній з них.

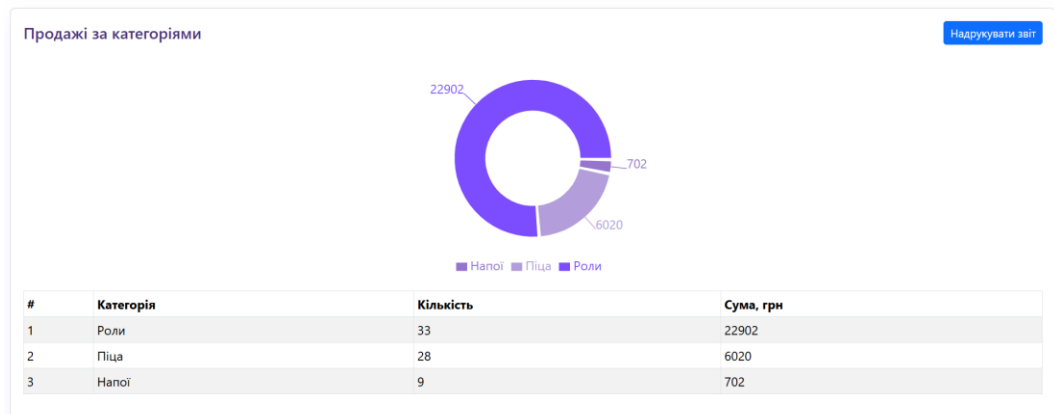


Рис. 3.40 – Продажі за категоріями

3.3. Тестування вебсистеми

Тестування програмного забезпечення є важливим етапом розробки, оскільки дозволяє виявити помилки до того, як система почне працювати з реальними даними закладу харчування. Для розробленої вебсистеми проводилось переважно функціональне тестування інтерфейсу та основних бізнес-процесів. Перевірка виконувалася вручну через вебінтерфейс, при цьому особлива увага приділялася коректності обробки вхідних даних, реакції системи на помилки та відповідності отриманих результатів очікуваній логіці роботи.

У таблиці 3.1 описані тест-кейси даної вебсистеми.

Таблиця 3.1

№	Назва тесту	Вхідні дані	Очікуваний результат	Статус
1	Авторизація з коректними даними	Логін: manager2 Пароль: 9846372	користувач успішно входить у систему, отримує JWT-токен, відкривається головна панель менеджера з повним набором доступних вкладок	Pass

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

2	Авторизація з некоректним паролем	Логін: manager2 Пароль: 111111	система не пускає користувача в особистий кабінет, відображається повідомлення про помилку авторизації, токен не видається	Pass
3	Створення нової категорії меню	Супи	категорія з'являється у списку на сторінці «Категорії», її можна використати під час створення позицій меню, дублювання назв не допускається	Pass
4	Зміна статусу замовлення на кухні	на сторінці «Кухня» для замовлення зі статусом «Готується» натискається кнопка «Виконано»	статус замовлення змінюється на «Виконано», воно зникає зі списку активних замовлень на кухні, а в загальному списку чеків відображається як виконане	Pass
5	Зміна наявності продукту	у вкладці «Наявність продуктів» для позиції «Філадельфія» вимикається перемикач «Є в наявності»	статус страви змінюється на «Немає в наявності», у модулі замовлень ця позиція стає недоступною для вибору, раніше оформлені чеки не змінюються	Pass

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

6	Видалення профіля й перевірка неможливості входу	у вкладці «Зареєструвати» обліковий запис видалається, після чого виконується спроба увійти під його логіном і паролем	профіль зникає зі списку, система не дозволяє авторизуватися видаленому обліковому запису та повертає повідомлення про помилку авторизації	Pass
7	Неможливість вибору столу, який уже зайнятий активним замовленням	спроба створити нове замовлення «На місці» для столу, який уже має замовлення	система не дозволяє повторно призначити цей стіл, показує повідомлення про те, що стіл уже зайнятий; офіціант повинен вибрати інший стіл або завершити поточне замовлення	Pass
8	Видалення позиції меню з категорії продуктів	у вкладці «Позиції меню» вибрано страву, після чого натиснуто «Видалити»	позиція зникає зі списку меню та перестає відображатися у модулі замовлень, якщо вона не використовується в активному замовленні	Pass
9	Неможливість здійснити оплату для замовлення зі статусом «Готується»	обрано чек зі статусом «Готується», у якого статус ще не змінено кухнею	кнопки оплати викликають повідомлення, що замовлення ще не готове до оплати	Pass

10	Перевірка коректності розрахунку суми замовлення при зміні кількості страв	у сформованому замовленні кількість страв змінюється за допомогою кнопок «+» та «-»	система миттєво перераховує суму для кожної позиції та загальну суму чека; якщо кількість стає 0, то позиція видаляється; жодних арифметичних помилок не виникає, підсумкова сума оновлюється правильно	Pass
11	Перевірка фільтрації та оновлення списку чеків за статусом і типом замовлення	у модальному вікні “Список чеків” вибрано різні фільтри: тип, статус, порядок сортування	у таблиці залишаються лише ті чеки, які відповідають вибраним критеріям. Після зміни фільтрів система завантажує оновлений список без перезавантаження сторінки	Pass
12	Спроба видалити стіл, який використовується в активному замовленні	у вкладці «Столи» обирається стіл, який наразі прив’язаний до замовлення зі статусом «Готується» «Виконано»	система не дозволяє видалити такий стіл і виводить повідомлення про те, що він використовується; таким чином забезпечується цілісність даних та уникнення помилок у замовленнях	Pass

Висновки до третього розділу

У третьому розділі дипломної роботи було виконано практичну реалізацію вебсистеми управління закладом харчування та детально описано її інтерфейс і основні можливості. Було показано, як система адаптується під різні ролі користувачів, надаючи кожному доступ лише до тих функцій, які необхідні для його роботи. Завдяки цьому інтерфейс залишається зручним, зрозумілим та не перевантаженим зайвими елементами.

Окремо описано модулі управління категоріями меню, стравами, столами, персоналом та обліковими записами, що забезпечують гнучке налаштування системи під потреби закладу. Також у розділі було детально описано роботу із замовленнями для обслуговування в закладі та для видачі на виніс, включно з формуванням і друком чеків, оновленням статусів та проведенням оплати. Це демонструє, що система підтримує повний цикл обслуговування клієнта.

У розділі наведено приклади роботи модуля звітності, який дає змогу аналізувати продажі, оцінювати популярність страв та відстежувати загальну динаміку діяльності закладу. Завдяки автоматичному формуванню таблиць і графіків користувач отримує зрозумілу й актуальну інформацію.

Проведене тестування підтвердило стабільність роботи вебсистеми та коректність виконання її основних функцій. Тест-кейси показали, що система правильно реагує на дії користувача, коректно обробляє помилки та забезпечує логічну послідовність усіх бізнес-процесів.

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

ВИСНОВКИ

У ході виконання дипломної роботи було проведено дослідження та реалізовано вебсистему управління закладом харчування, що охоплювало весь цикл розробки. На початковому етапі було визначено актуальність теми та проаналізовано існуючі аналоги, що дало можливість сформулювати вимоги до функціоналу, архітектури та інструменти, які були необхідними для створення багатофункціональної системи. Було обґрунтовано вибір клієнт-серверної моделі, технологій Spring Boot, React, MySQL та механізму авторизації на основі JWT, що забезпечило безпечну та стабільну взаємодію між користувачем і сервером.

У процесі розробки було спроектовано структуру бази даних, визначено моделі об'єктів та логіку їхньої взаємодії. Реалізовані алгоритми охоплюють авторизацію користувачів, роботу з меню, створення та редагування замовлень, взаємодію з кухнею, оновлення статусів та формування чеків.

Під час розробки значну увагу було приділено створенню інтерфейсу, який враховує роль користувача і автоматично формує для нього відповідне робоче місце. Описані та реалізовані модулі управління категоріями, позиціями меню, столами, персоналом і обліковими записами забезпечують гнучку конфігурацію системи. Детально опрацьовано процеси оформлення замовлень «На місці» та «З собою», друку чеків, зміни статусів і проведення оплати, що дає змогу автоматизувати роботу персоналу та зменшити кількість помилок. А також модуль аналітики, що формує графіки та зведені звіти на основі даних із бази, що дозволяє керівництву оцінювати результати роботи закладу.

Під час тестування підтверджено коректність роботи системи та відповідність реалізованої логіки очікуваним результатам. Усі основні сценарії взаємодії користувача були успішно відтворені й перевірені. Система правильно реагує на помилки, забезпечує цілісність даних і не допускає некоректних операцій, що свідчить про її готовність до практичного використання.

		Рябо О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рябко О.Д., Єфремов Ю.М. Актуальність розробки системи управління закладами харчування // Комп'ютерні технології: інновації, проблеми, рішення: VIII Всеукраїнська науково-технічна конференція (м. Житомир, 02-03 грудня 2025р.). Житомир, 2025. (подано до друку).
2. Рябко О.Д., Єфремов Ю.М. Аналіз існуючих системи управління закладами харчування // Комп'ютерні технології: інновації, проблеми, рішення: VIII Всеукраїнська науково-технічна конференція (м. Житомир, 02-03 грудня 2025р.). Житомир, 2025. (подано до друку).
3. Microsoft. REST API Design Guide. URL: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design> (дата звернення: 25.11.2025).
4. Spring Data JPA Documentation. URL: <https://docs.spring.io/spring-data/jpa/reference/> (дата звернення: 25.11.2025).
5. React Official Documentation. URL: <https://react.dev/reference/react> (дата звернення: 25.11.2025).
6. MySQL Reference Manual. Oracle Corporation. URL: <https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 25.11.2025).
7. JWT.io. Introduction to JSON Web Tokens. URL: <https://www.jwt.io/introduction> (дата звернення: 26.11.2025).
8. GeeksForGeeks. JSON Web Token (JWT) Explained. URL: <https://www.geeksforgeeks.org/web-tech/json-web-token-jwt/> (дата звернення: 26.11.2025).
9. FreeCodeCamp. What Are JSON Web Tokens? URL: <https://www.freecodecamp.org/news/what-are-json-web-tokens-jwt/> (дата звернення: 26.11.2025).
10. Auth0. JWT Introduction. URL: <https://auth0.com/docs/secure/tokens/json-web-tokens> (дата звернення: 26.11.2025).
11. Spring Security Architecture. URL: <https://docs.spring.io/spring-security/reference/servlet/architecture.html> (дата звернення: 27.11.2025).

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

12. Poster POS. URL: <https://joinposter.com/ua> (дата звернення: 27.11.2025).
13. SkyService POS. URL: <https://skyservice.pro/en/main/> (дата звернення: 27.11.2025).
14. Syrve Україна. URL: <https://ua.restasystem.com/> (дата звернення: 28.11.2025).
15. SERVIO POS. URL: <https://expertsolution.com.ua/produkty/pz/servio-pos-dlya-avtomatyzatsiyi-restorannogo-biznesu/> (дата звернення: 28.11.2025).
16. GeeksForGeeks. Client–Server Architecture – System Design. URL: <https://www.geeksforgeeks.org/system-design/client-server-architecture-system-design/> (дата звернення: 29.11.2025).
17. Baeldung. Spring Boot Tutorials. URL: <https://www.baeldung.com/spring-boot> (дата звернення: 29.11.2025).
18. Bootstrap Documentation. URL: <https://getbootstrap.com/docs/> (дата звернення: 01.12.2025).
19. Visual Paradigm. What is Use Case Diagram? URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> (дата звернення: 02.12.2025).
20. What is a Relational Database (RDBMS)? Google Cloud. URL: <https://cloud.google.com/learn/what-is-a-relational-database> (дата звернення: 02.12.2025).
21. Description of the database normalization basics. Microsoft Support. URL: <https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description> (дата звернення: 02.12.2025).
22. MySQL InnoDB Storage Engine. Oracle Corporation. URL: <https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html> (дата звернення: 02.12.2025).
23. DataCamp – ACID Transactions Explained. URL: <https://www.datacamp.com/blog/acid-transactions> (дата звернення: 03.12.2025).

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

24. Spring Boot Documentation – Architecture & Web Layer. URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/web.html> (дата звернення: 05.12.2025).

25. Baeldung – DTO Pattern in Spring. URL: <https://www.baeldung.com/java-dto-pattern> (дата звернення: 05.12.2025).

26. ModelMapper Documentation. URL: <http://modelmapper.org/getting-started/> (дата звернення: 06.12.2025).

27. Repository Pattern – Martin Fowler: [Електронний ресурс]. – Режим доступу: <https://martinfowler.com/eaaCatalog/repository.html> (дата звернення: 06.12.2025).

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

JwtProvider.java

```

package com.riabko.diploma.restaurant_management_system.config;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.stereotype.Service;
import javax.crypto.SecretKey;
import java.util.Collection;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

@Service
public class JwtProvider {
    private final SecretKey key = Keys.hmacShaKeyFor(JwtConstant.SECRET_KEY.getBytes());

    public String generateJwtToken(Authentication auth) {
        Collection<? extends GrantedAuthority> authorities = auth.getAuthorities();
        String userRole = populateAuthorities(authorities);

        String jwt = Jwts.builder().setIssuedAt(new Date())
            .setExpiration((new Date(new Date().getTime()+86400000)))
            .claim("username", auth.getName())
            .claim("authorities", userRole)
            .signWith(key)
            .compact();
        return jwt;
    }

    public String getUsernameFromJwtToken(String jwt) {
        jwt = jwt.substring(7);
        Claims claims = Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(jwt).getBody();
        String username = String.valueOf(claims.get("username"));
        return username;
    }

    private String populateAuthorities(Collection<? extends GrantedAuthority> authorities) {
        Set<String> auths = new HashSet<>();
        for (GrantedAuthority authority : authorities) {
            auths.add(authority.getAuthority());
        }

        return String.join(",", auths);
    }
}

```

JwtTokenValidator.java

```

package com.riabko.diploma.restaurant_management_system.config;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import jakarta.servlet.FilterChain;

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.apache.tomcat.util.codec.binary.Base64;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.crypto.SecretKey;
import java.io.IOException;
import java.util.List;

public class JwtTokenValidator extends OncePerRequestFilter {

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
    FilterChain filterChain) throws ServletException, IOException {
        String jwt = request.getHeader(JwtConstant.JWT_HEADER);
        if (jwt != null) {
            jwt = jwt.substring(7);
            try{
                SecretKey key = Keys.hmacShaKeyFor(JwtConstant.SECRET_KEY.getBytes());
                Claims claims =
                Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(jwt).getBody();
                String username = String.valueOf(claims.get("username"));
                String authorities = String.valueOf(claims.get("authorities"));
                List<GrantedAuthority> auth =
                AuthorityUtils.commaSeparatedStringToAuthorityList(authorities);
                Authentication authentication = new UsernamePasswordAuthenticationToken(username,
                null, auth);
                SecurityContextHolder.getContext().setAuthentication(authentication);
            }
            catch(Exception e){
                throw new BadCredentialsException("Недійсний JWT токен");
            }
        }
        filterChain.doFilter(request, response);
    }
}

```

WebSecurityConfiguration.java

```

package com.riabko.diploma.restaurant_management_system.config;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;

import java.util.Arrays;
import java.util.Collections;

@Configuration
@EnableWebSecurity
public class WebSecurityConfiguration {
    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http, CorsConfigurationSource
corsConfigurationSource) throws Exception {
        http.sessionManagement(management ->
management.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .csrf(csrf -> csrf.disable())
        .cors(cors -> cors.configurationSource(corsConfigurationSource))
        .formLogin(form -> form.disable())

        .httpBasic(basic -> basic.disable())
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/api/auth/").permitAll()
            .requestMatchers("/api/admin/").hasRole("ADMIN")
            .requestMatchers("/api/kitchen/").hasRole("KITCHEN")
            .requestMatchers("/api/waiter/").hasRole("WAITER")
            .requestMatchers("/api/manager/").hasRole("MANAGER")
            .requestMatchers("/uploads/").permitAll()
            .requestMatchers(HttpMethod.PATCH,
"/api/products/*/availability").hasRole("KITCHEN")
            .anyRequest().authenticated()
        ).addFilterBefore(new JwtTokenValidator(), BasicAuthenticationFilter.class);
        return http.build();
    }
    @Bean
    public CorsConfigurationSource corsConfigurationSource() {
        return new CorsConfigurationSource() {
            @Override
            public CorsConfiguration getCorsConfiguration(HttpServletRequest request) {
                CorsConfiguration config = new CorsConfiguration();
                config.setAllowedOrigins(Arrays.asList("http://localhost:3000"));
                config.setAllowedMethods(Collections.singletonList("*"));
                config.setAllowCredentials(true);
                config.setAllowedHeaders(Collections.singletonList("*"));
                config.setMaxAge(3600L);
                return config;
            }
        };
    }
}

```

		Рябоко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@Bean
PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
}

```

AdminController.java

```

package com.riabko.diploma.restaurant_management_system.controllers;
import com.riabko.diploma.restaurant_management_system.dto.request.RegisterRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateUserRequest;
import com.riabko.diploma.restaurant_management_system.dto.response.MessageResponse;
import com.riabko.diploma.restaurant_management_system.dto.response.RegisterResponse;
import com.riabko.diploma.restaurant_management_system.dto.response.UpdateUserResponse;
import com.riabko.diploma.restaurant_management_system.dto.response.UserDto;
import com.riabko.diploma.restaurant_management_system.models.User;
import com.riabko.diploma.restaurant_management_system.models.UserRole;
import com.riabko.diploma.restaurant_management_system.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

```

```

@RestController
@RequestMapping("/api/admin")
@CrossOrigin(origins = "http://localhost:3000")
public class AdminController {

```

```

    @Autowired
    private UserRepository userRepository;

```

```

    @Autowired
    private PasswordEncoder passwordEncoder;

```

```

    @PostMapping("/register")
    public ResponseEntity<RegisterResponse> register(@RequestBody RegisterRequest request) {
        String username = request.getUsername();
        String rawPassword = request.getPassword();
        UserRole role = request.getUserRole();
        RegisterResponse resp = new RegisterResponse();
        if (username == null || username.trim().isEmpty()) {
            resp.setMessage("Ім'я користувача не може бути порожнім");
            return new ResponseEntity<>(resp, HttpStatus.BAD_REQUEST);
        }
        if (rawPassword == null || rawPassword.length() < 5) {
            resp.setMessage("Пароль має містити щонайменше 5 символів");
            return new ResponseEntity<>(resp, HttpStatus.BAD_REQUEST);
        }
    }

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (role == null) {
    resp.setMessage("Роль користувача є обов'язковою");
    return new ResponseEntity<>(resp, HttpStatus.BAD_REQUEST);
}
if (userRepository.existsByUsername(username)) {
    resp.setMessage("Користувач з таким ім'ям вже існує");
    return new ResponseEntity<>(resp, HttpStatus.CONFLICT);
}
User user = new User();
user.setUsername(username.trim());
user.setPassword(passwordEncoder.encode(rawPassword));
user.setUserRole(role);
userRepository.save(user);
resp.setMessage("Користувача успішно зареєстровано");
resp.setUsername(user.getUsername());
resp.setUserRole(user.getUserRole());
return new ResponseEntity<>(resp, HttpStatus.CREATED);
}

@GetMapping("/users")
public ResponseEntity<List<UserDto>> getAllUsers() {
    List<User> users = userRepository.findAll();
    List<UserDto> dtos = users.stream()
        .map(this::toUserDto)
        .collect(Collectors.toList());

    return ResponseEntity.ok(dtos);
}

private UserDto toUserDto(User user) {
    UserDto dto = new UserDto();
    dto.setId(user.getId());
    dto.setUsername(user.getUsername());
    dto.setUserRole(user.getUserRole());
    return dto;
}

@PutMapping("/users/{id}")
public ResponseEntity<UpdateUserResponse> updateUser(@PathVariable Long id, @RequestBody
UpdateUserRequest request) {
    UpdateUserResponse resp = new UpdateUserResponse();
    Optional<User> optionalUser = userRepository.findById(id);
    if (optionalUser.isEmpty()) {
        resp.setMessage("Користувача не знайдено");
        return new ResponseEntity<>(resp, HttpStatus.NOT_FOUND);
    }
    User user = optionalUser.get();
    String newUsername = request.getUsername();
    UserRole newRole = request.getUserRole();

    String newPassword = request.getNewPassword();
    if (newUsername == null || newUsername.trim().isEmpty()) {
        resp.setMessage("Ім'я користувача не може бути порожнім");
        return new ResponseEntity<>(resp, HttpStatus.BAD_REQUEST);
    }

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    if (newRole == null) {
        resp.setMessage("Роль користувача є обов'язковою");
        return new ResponseEntity<>(resp, HttpStatus.BAD_REQUEST);
    }
    if (!user.getUsername().equals(newUsername.trim())
        && userRepository.existsByUsername(newUsername.trim())) {
        resp.setMessage("Користувач з таким ім'ям вже існує");
        return new ResponseEntity<>(resp, HttpStatus.CONFLICT);
    }
    if (newPassword != null && !newPassword.trim().isEmpty()) {
        if (newPassword.length() < 5) {
            resp.setMessage("Новий пароль має містити щонайменше 5 символів");
            return new ResponseEntity<>(resp, HttpStatus.BAD_REQUEST);
        }
        user.setPassword(passwordEncoder.encode(newPassword.trim()));
    }

    user.setUsername(newUsername.trim());
    user.setUserRole(newRole);
    userRepository.save(user);
    resp.setMessage("Користувача успішно оновлено");
    resp.setId(user.getId());
    resp.setUsername(user.getUsername());
    resp.setUserRole(user.getUserRole());
    return ResponseEntity.ok(resp);
}

@DeleteMapping("/users/{id}")
public ResponseEntity<MessageResponse> deleteUser(@PathVariable Long id) {
    MessageResponse resp = new MessageResponse();
    Optional<User> optionalUser = userRepository.findById(id);
    if (optionalUser.isEmpty()) {
        resp.setMessage("Користувача не знайдено");
        return new ResponseEntity<>(resp, HttpStatus.NOT_FOUND);
    }
    userRepository.deleteById(id);
    resp.setMessage("Користувача успішно видалено");
    return ResponseEntity.ok(resp);
}
}

```

AuthController.java

```

package com.riabko.diploma.restaurant_management_system.controllers;
import com.riabko.diploma.restaurant_management_system.config.JwtProvider;
import com.riabko.diploma.restaurant_management_system.models.User;
import com.riabko.diploma.restaurant_management_system.models.UserRole;
import com.riabko.diploma.restaurant_management_system.repositories.UserRepository;
import com.riabko.diploma.restaurant_management_system.dto.request.LoginRequest;
import com.riabko.diploma.restaurant_management_system.dto.response.AuthResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import java.util.List;
```

```
@Controller
@RequestMapping("/api/auth")
@CrossOrigin(origins = "http://localhost:3000")
public class AuthController {
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
    private JwtProvider jwtProvider;
    @PostMapping("/login")
    public ResponseEntity<AuthResponse>login(@RequestBody LoginRequest request){
        User user = userRepository.findByUsername(request.getUsername())
            .orElseThrow(() -> new BadCredentialsException("Невірний логін або пароль"));
        if (!passwordEncoder.matches(request.getPassword(), user.getPassword())) {
            throw new BadCredentialsException("Невірний логін або пароль");
        }
        var auth = new UsernamePasswordAuthenticationToken(
            user.getUsername(),
            null,
            List.of(new SimpleGrantedAuthority(user.getUserRole().name()))
        );
        String jwt = jwtProvider.generateJwtToken(auth);
        AuthResponse authResponse = new AuthResponse();
        authResponse.setJwt(jwt);
        authResponse.setMessage("Вхід успішний");
        authResponse.setUserRole(user.getUserRole());
        return new ResponseEntity<>(authResponse, HttpStatus.OK);
    }
}
```

ProductController.java

```
package com.riabko.diploma.restaurant_management_system.controllers;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateProductRequest;
import
com.riabko.diploma.restaurant_management_system.dto.request.UpdateProductAvailabilityRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateProductRequest;
import com.riabko.diploma.restaurant_management_system.models.Product;
import com.riabko.diploma.restaurant_management_system.repositories.ProductRepository;
```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

import com.riabko.diploma.restaurant_management_system.services.ImageStorageService;
import com.riabko.diploma.restaurant_management_system.services.ProductService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/api/products")
@CrossOrigin(origins = "http://localhost:3000")

public class ProductController {
    private final ProductService productService;
    private final ImageStorageService imageStorageService;
    public ProductController(ProductService productService, ImageStorageService
imageStorageService) {
        this.productService = productService;
        this.imageStorageService = imageStorageService;
    }
    @GetMapping
    public ResponseEntity<List<Product>> getAll() {
        return ResponseEntity.ok(productService.findAll());
    }
    @GetMapping("/{id}")
    public ResponseEntity<?> getById(@PathVariable Long id) {
        try {
            Product product = productService.findById(id);
            return ResponseEntity.ok(product);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
        }
    }
    @PostMapping
    public ResponseEntity<?> create(@RequestBody CreateProductRequest request) {
        try {
            Product created = productService.createProduct(request);
            return ResponseEntity.status(HttpStatus.CREATED).body(created);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }
    @PutMapping("/{id}")
    public ResponseEntity<?> update(@PathVariable Long id, @RequestBody UpdateProductRequest
request) {
        try {
            Product updated = productService.updateProduct(id, request);

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return ResponseEntity.ok(updated);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
    }
}

@DeleteMapping("/{id}")
public ResponseEntity<?> delete(@PathVariable Long id) {
    try {
        productService.deleteProduct(id);
        return ResponseEntity.ok("Продукт успішно видалено");
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
    }
}

@PostMapping("/upload-image")
public ResponseEntity<?> uploadImage(@RequestParam("file") MultipartFile file) {
    try {
        String url = imageStorageService.storeProductImage(file);
        Map<String, String> response = new HashMap<>();
        response.put("url", url);
        return ResponseEntity.ok(response);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
    }
}

@PatchMapping("/{id}/availability")
public ResponseEntity<?> updateAvailability(@PathVariable Long id, @RequestBody
UpdateProductAvailabilityRequest request
) {
    try {
        Product updated = productService.updateProductAvailability(id, request.getInStock());
        return ResponseEntity.ok(updated);
    } catch (Exception e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
}

```

CategoryController.java

```

package com.riabko.diploma.restaurant_management_system.controllers;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateCategoryRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateCategoryRequest;
import com.riabko.diploma.restaurant_management_system.models.Category;
import com.riabko.diploma.restaurant_management_system.services.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

```

```

@RestController
@RequestMapping("/api/categories")

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

@CrossOrigin(origins = "http://localhost:3000")
public class CategoryController {
    private final CategoryService categoryService;
    public CategoryController(CategoryService categoryService) {
        this.categoryService = categoryService;
    }

    @GetMapping
    public ResponseEntity<List<Category>> getAllCategories() {
        return ResponseEntity.ok(categoryService.findAll());
    }

    @GetMapping("/{id}")
    public ResponseEntity<?> getCategoryById(@PathVariable Long id) {
        try {
            Category category = categoryService.findCategoryById(id);
            return ResponseEntity.ok(category);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
        }
    }

    @PostMapping
    public ResponseEntity<?> createCategory(@RequestBody CreateCategoryRequest request) {
        try {
            Category created = categoryService.createCategory(request);
            return ResponseEntity.status(HttpStatus.CREATED).body(created);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }

    @PutMapping("/{id}")
    public ResponseEntity<?> updateCategory(
        @PathVariable Long id,
        @RequestBody UpdateCategoryRequest request
    ) {
        try {
            Category updated = categoryService.updateCategory(id, request);
            return ResponseEntity.ok(updated);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?> deleteCategory(@PathVariable Long id) {
        try {
            categoryService.deleteCategory(id);
            return ResponseEntity.ok("Категорію успішно видалено");
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```
}
}
```

OrderController.java

```
package com.riabko.diploma.restaurant_management_system.controllers;
import com.riabko.diploma.restaurant_management_system.dto.request.CashCheckoutRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateOrderRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateOrderStatusRequest;
import com.riabko.diploma.restaurant_management_system.models.Order;
import com.riabko.diploma.restaurant_management_system.models.OrderStatus;
import com.riabko.diploma.restaurant_management_system.models.OrderType;
import com.riabko.diploma.restaurant_management_system.services.OrderService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/api/orders")
```

```
@CrossOrigin(origins = "http://localhost:3000")
```

```
public class OrderController {
```

```
    private final OrderService orderService;
```

```
    public OrderController(OrderService orderService) {
```

```
        this.orderService = orderService;
```

```
    }
```

```
    @PostMapping
```

```
    public ResponseEntity<?> createOrder(@RequestBody CreateOrderRequest request) {
```

```
        try {
```

```
            Order created = orderService.createOrder(request);
```

```
            return ResponseEntity.status(HttpStatus.CREATED).body(created);
```

```
        } catch (Exception e) {
```

```
            return ResponseEntity.badRequest().body(e.getMessage());
```

```
        }
```

```
    }
```

```
    @GetMapping
```

```
    public ResponseEntity<List<Order>> getAll(@RequestParam(required = false) OrderType type,
```

```
@RequestParam(required = false) OrderStatus status) {
```

```
        if (type != null && status != null) {
```

```
            return ResponseEntity.ok(orderService.findByTypeAndStatus(type, status));
```

```
        }
```

```
        return ResponseEntity.ok(orderService.findAll());
```

```
    }
```

```
    @GetMapping("/{id}")
```

```
    public ResponseEntity<?> getById(@PathVariable Long id) {
```

```
        try {
```

```
            Order order = orderService.findById(id);
```

```
            return ResponseEntity.ok(order);
```

```
        } catch (Exception e) {
```

```
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
```

```
        }
```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    @PatchMapping("/{id}/status")
    public ResponseEntity<?> updateStatus(@PathVariable Long id, @RequestBody
UpdateOrderStatusRequest request) {
        try {
            Order updated = orderService.updateStatus(id, request.getStatus());
            return ResponseEntity.ok(updated);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
    @PostMapping("/{id}/cash-checkout")
    public ResponseEntity<?> cashCheckout(@PathVariable Long id, @RequestBody
CashCheckoutRequest request) {
        try {
            Order updated = orderService.cashCheckout(id, request);

            return ResponseEntity.ok(updated);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
    @PutMapping("/{id}")
    public ResponseEntity<?> updateOrder(@PathVariable Long id, @RequestBody
CreateOrderRequest request) {
        try {
            Order updated = orderService.updateOrder(id, request);
            return ResponseEntity.ok(updated);
        } catch (Exception e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
}

```

OrderRepository.java

```

package com.riabko.diploma.restaurant_management_system.repositories;
import com.riabko.diploma.restaurant_management_system.dto.response.DailySalesSummaryDto;
import com.riabko.diploma.restaurant_management_system.dto.response.SalesByCategoryDto;
import com.riabko.diploma.restaurant_management_system.dto.response.SalesByItemDto;
import com.riabko.diploma.restaurant_management_system.models.*;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import java.time.LocalDateTime;
import java.util.Collection;
import java.util.List;
import java.util.Optional;

public interface OrderRepository extends JpaRepository<Order, Long> {
    List<Order> findByOrderTypeAndStatus(OrderType orderType, OrderStatus status);
    // Перевірка, чи є хоч одне замовлення на конкретний стіл з певними статусами
    boolean existsByTableAndStatusIn(RestaurantTable table, Collection<OrderStatus> statuses);
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```
@Modifying
@Query("UPDATE Order o SET o.table = null WHERE o.table = :table")
void clearTableForTable(@Param("table") RestaurantTable table);
boolean existsByWaiterAndStatusIn(Staff waiter, Collection<OrderStatus> statuses);
```

```
@Modifying
@Query("UPDATE Order o SET o.waiter = null WHERE o.waiter = :waiter")
void clearWaiterForStaff(@Param("waiter") Staff waiter);
```

// Отримання статистики продажів по категоріях

```
@Query("SELECT new
com.riabko.diploma.restaurant_management_system.dto.response.SalesByCategoryDto(" +
    "c.name, SUM(oi.quantity), SUM(oi.totalPrice)) " +
    "FROM Order o " +
    "JOIN o.items oi " +
    "JOIN oi.product p " +
    "JOIN p.category c " +
    "WHERE o.status =
com.riabko.diploma.restaurant_management_system.models.OrderStatus.PAID " +
    "AND o.createdAt BETWEEN :start AND :end " +
    "GROUP BY c.name " +
    "ORDER BY SUM(oi.totalPrice) DESC")
List<SalesByCategoryDto> getSalesByCategory(
    @Param("start") LocalDateTime start,
    @Param("end") LocalDateTime end
);
```

// Отримання статистики продажів по конкретних меню-позиціях

```
@Query("SELECT new
com.riabko.diploma.restaurant_management_system.dto.response.SalesByItemDto(" +
    "oi.productName, oi.sizeLabel, SUM(oi.quantity), SUM(oi.totalPrice)) " +
    "FROM Order o " +
    "JOIN o.items oi " +
    "WHERE o.status =
com.riabko.diploma.restaurant_management_system.models.OrderStatus.PAID " +
    "AND o.createdAt BETWEEN :start AND :end " +
    "GROUP BY oi.productName, oi.sizeLabel " +
    "ORDER BY SUM(oi.totalPrice) DESC")
List<SalesByItemDto> getSalesByItem(
    @Param("start") LocalDateTime start,
    @Param("end") LocalDateTime end
);
```

//Денна статистика продажів

```
@Query("SELECT new
com.riabko.diploma.restaurant_management_system.dto.response.DailySalesSummaryDto(" +
    "DATE(o.createdAt), COUNT(o), SUM(o.totalAmount)) " +
    "FROM Order o " +
    "WHERE o.status =
com.riabko.diploma.restaurant_management_system.models.OrderStatus.PAID " +
    "AND o.createdAt BETWEEN :start AND :end " +
    "GROUP BY DATE(o.createdAt) " +
```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"ORDER BY DATE(o.createdAt)")
List<DailySalesSummaryDto> getDailySalesSummary(
    @Param("start") LocalDateTime start,
    @Param("end") LocalDateTime end
);
}

```

OrderItemRepository.java

```

package com.riabko.diploma.restaurant_management_system.repositories;
import com.riabko.diploma.restaurant_management_system.models.OrderItem;
import com.riabko.diploma.restaurant_management_system.models.OrderStatus;
import com.riabko.diploma.restaurant_management_system.models.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import java.util.Collection;
public interface OrderItemRepository extends JpaRepository<OrderItem, Long> {
    boolean existsByProductAndOrder_StatusIn(Product product, Collection<OrderStatus> statuses);
    // Обнулити size_option для всіх позицій цього продукту
    @Modifying(clearAutomatically = true, flushAutomatically = true)
    @Query("UPDATE OrderItem oi SET oi.sizeOption = null WHERE oi.product = :product")
    void clearSizeOptionForProduct(@Param("product") Product product);

    // Обнулити product
    @Modifying(clearAutomatically = true, flushAutomatically = true)
    @Query("UPDATE OrderItem oi SET oi.product = null WHERE oi.product = :product")
    void clearProductForProduct(@Param("product") Product product);
}

```

CategoryServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateCategoryRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateCategoryRequest;
import com.riabko.diploma.restaurant_management_system.models.Category;
import com.riabko.diploma.restaurant_management_system.repositories.CategoryRepository;
import com.riabko.diploma.restaurant_management_system.repositories.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.annotation.Secured;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;

@Service
public class CategoryServiceImpl implements CategoryService {

    private final CategoryRepository categoryRepository;
    private final ProductRepository productRepository;
    public CategoryServiceImpl(CategoryRepository categoryRepository, ProductRepository
productRepository) {
        this.categoryRepository = categoryRepository;
    }
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        this.productRepository = productRepository;
    }
    @Override
    public Category findByName(String name) throws Exception {
        Category category = categoryRepository.findByName(name);
        if (category == null) {
            throw new Exception("Категорія "" + name + "" не знайдена");
        }
        return category;
    }

    @Override
    public Category findCategoryById(Long id) throws Exception {
        Optional<Category> category = categoryRepository.findById(id);
        if (category.isEmpty()) {
            throw new Exception("Категорію з id="" + id + "" не знайдено");
        }
        return category.get();
    }

    @Override
    public Category createCategory(CreateCategoryRequest request) throws Exception {
        if (request.getName() == null || request.getName().isBlank()) {
            throw new Exception("Назва категорії не може бути порожньою");
        }

        if (categoryRepository.existsByName(request.getName())) {
            throw new Exception("Категорія з ім'ям "" + request.getName() + "" вже існує");
        }

        Category category = new Category();
        category.setName(request.getName().trim());
        return categoryRepository.save(category);
    }

    @Override
    public Category updateCategory(Long id, UpdateCategoryRequest request) throws Exception {
        Category category = categoryRepository.findById(id)
            .orElseThrow(() -> new Exception("Категорію з id="" + id + "" не знайдено"));
        if (request.getName() != null && !request.getName().isBlank()) {
            String newName = request.getName().trim();
            if (!newName.equals(category.getName())
                && categoryRepository.existsByName(newName)) {
                throw new Exception("Ім'я категорії "" + newName + "" вже зайнято");
            }
            category.setName(newName);
        }
        return categoryRepository.save(category);
    }

    @Override
    public void deleteCategory(Long id) throws Exception {

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

Category category = categoryRepository.findById(id)
    .orElseThrow(() -> new Exception("Категорію з id=" + id + " не знайдено"));
boolean hasProducts = productRepository.existsByCategory(category);
if (hasProducts) {
    throw new Exception("Неможливо видалити категорію! Вона містить продукти! " +
        "Спочатку видаліть або перенесіть ці продукти в іншу категорію!");
};
}
categoryRepository.delete(category);
}

@Override
public List<Category> findAll() {
    return categoryRepository.findAll();
}
}

```

FileSystemImageStorageService.java

```

package com.riabko.diploma.restaurant_management_system.services;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import org.springframework.web.multipart.MultipartFile;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

@Service
public class FileSystemImageStorageService implements ImageStorageService {
    @Value("${app.upload-dir:uploads}")
    private String uploadDir;
    @Override
    public String storeProductImage(MultipartFile file) throws Exception {
        if (file.isEmpty()) {
            throw new Exception("Файл порожній");
        }
        String originalFilename = StringUtils.cleanPath(file.getOriginalFilename());
        String extension = "";
        int dotIndex = originalFilename.lastIndexOf('.');
        if (dotIndex >= 0) {
            extension = originalFilename.substring(dotIndex);
        }
        String fileName = UUID.randomUUID().toString() + extension;

        // Створюю шлях до папки uploads/products
        Path uploadPath = Paths.get(uploadDir, "products");
        try {
            Files.createDirectories(uploadPath);
            Path destinationFile = uploadPath.resolve(fileName).normalize().toAbsolutePath();
            file.transferTo(destinationFile.toFile());

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

        // Повертаю URL, який буде працювати через WebConfig
        return "/uploads/products/" + fileName;
    } catch (IOException e) {
        throw new Exception("Помилка при збереженні файлу", e);
    }
}
}

```

UserServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.config.JwtProvider;
import com.riabko.diploma.restaurant_management_system.models.User;
import com.riabko.diploma.restaurant_management_system.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class UserServiceImpl implements UserService {

    private final UserRepository userRepository;
    public UserServiceImpl(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Autowired
    private JwtProvider jwtProvider;

    @Override
    public User findUserByJwtToken(String jwt) {
        // дістаємо username з JWT
        String username = jwtProvider.getUsernameFromJwtToken(jwt);
        // шукаємо користувача з таким username
        return userRepository.findByUsername(username).orElseThrow(() -> new
        UsernameNotFoundException("Користувача з username " + username + " не знайдено"));
    }

    @Override
    public User findUserByUsername(String username) {
        return userRepository.findByUsername(username).orElseThrow(() -> new
        UsernameNotFoundException("Користувача з username " + username + " не знайдено"));
    }
}

```

ProductServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateProductRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.ProductSizeDto;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateProductRequest;
import com.riabko.diploma.restaurant_management_system.models.Category;
import com.riabko.diploma.restaurant_management_system.models.OrderStatus;
import com.riabko.diploma.restaurant_management_system.models.Product;

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import com.riabko.diploma.restaurant_management_system.models.ProductSizeOption;
import com.riabko.diploma.restaurant_management_system.repositories.CategoryRepository;
import com.riabko.diploma.restaurant_management_system.repositories.OrderItemRepository;
import com.riabko.diploma.restaurant_management_system.repositories.ProductRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.ArrayList;
import java.util.List;
```

@Service

```
public class ProductServiceImpl implements ProductService {
```

```
    private final ProductRepository productRepository;
    private final CategoryRepository categoryRepository;
    private final OrderItemRepository orderItemRepository;
    public ProductServiceImpl(ProductRepository productRepository, CategoryRepository
categoryRepository, OrderItemRepository orderItemRepository) {
        this.productRepository = productRepository;
        this.categoryRepository = categoryRepository;
        this.orderItemRepository = orderItemRepository;
    }
```

@Override

```
public List<Product> findAll() {
    return productRepository.findAll();
}
```

@Override

```
public Product findById(Long id) throws Exception {
    return productRepository.findById(id)
        .orElseThrow(() -> new Exception("Продукт з id=" + id + " не знайдено"));
}
```

@Override

```
public Product createProduct(CreateProductRequest request) throws Exception {
    if (request.getName() == null || request.getName().isBlank()) {
        throw new Exception("Назва продукту не може бути порожньою");
    }
    if (request.getCategoryId() == null) {
        throw new Exception("Категорія продукту є обов'язковою");
    }
    String trimmedName = request.getName().trim();
    if (productRepository.existsByNameIgnoreCaseAndCategory_Id(trimmedName,
request.getCategoryId())) {
        throw new Exception("Продукт з такою назвою вже існує в цій категорії");
    }
}
```

```
    Category category = categoryRepository.findById(request.getCategoryId())
        .orElseThrow(() -> new Exception("Категорію не знайдено"));
    Product product = new Product();
    product.setName(request.getName().trim());
    product.setCategory(category);
```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

product.setInStock(request.getInStock() == null ? request.getInStock());
product.setImageUrl(request.getImageUrl());

List<ProductSizeOption> sizeOptions = new ArrayList<>();
if (request.getSizes() != null) {
    for (ProductSizeDto dto : request.getSizes()) {
        if (dto.getSizeLabel() == null || dto.getSizeLabel().isBlank()) {
            continue;
        }
        ProductSizeOption option = new ProductSizeOption();
        option.setSizeLabel(dto.getSizeLabel().trim());
        option.setPrice(dto.getPrice());
        option.setProduct(product);
        sizeOptions.add(option);
    }
}
if (sizeOptions.isEmpty()) {
    throw new Exception("Потрібно додати хоча б один розмір та ціну для продукту");
}
product.setSizes(sizeOptions);
return productRepository.save(product);
}

@Override
@Transactional
public Product updateProduct(Long id, UpdateProductRequest request) throws Exception {
    Product product = productRepository.findById(id)
        .orElseThrow(() -> new Exception("Продукт з id=" + id + " не знайдено"));
    String newName = product.getName();
    Long newCategoryId = product.getCategory() != null ? product.getCategory().getId() : null;

    if (request.getName() != null && !request.getName().isBlank()) {
        newName = request.getName().trim();
    }

    if (request.getCategoryId() != null) {
        newCategoryId = request.getCategoryId();
    }

    if (newName == null || newName.isBlank()) {
        throw new Exception("Назва продукту не може бути порожньою");
    }
    if (newCategoryId == null) {
        throw new Exception("Категорія продукту є обов'язковою");
    }

    if (productRepository.existsByNameIgnoreCaseAndCategory_IdAndIdNot(newName,
        newCategoryId, id)) {
        throw new Exception("Інший продукт з такою назвою вже існує в цій категорії");
    }
    product.setName(newName);
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (!newCategoryId.equals(product.getCategory().getId())) {
    Category category = categoryRepository.findById(newCategoryId)
        .orElseThrow(() -> new Exception("Категорію не знайдено"));
    product.setCategory(category);
}

if (request.getInStock() != null) {
    product.setInStock(request.getInStock());
}

if (request.getImageUrl() != null) {
    product.setImageUrl(request.getImageUrl());
}

if (request.getSizes() != null) {

    List<OrderStatus> activeStatuses = List.of(
        OrderStatus.PREPARING,
        OrderStatus.DONE
    );

    boolean hasActiveItems =
        orderItemRepository.existsByProductAndOrder_StatusIn(product, activeStatuses);

    if (hasActiveItems) {
        throw new Exception("Неможливо змінювати розміри цього продукту: є активні
замовлення з ним (PREPARING або DONE).");
    }

    List<ProductSizeOption> newSizes = new ArrayList<>();
    for (ProductSizeDto dto : request.getSizes()) {
        if (dto.getSizeLabel() == null || dto.getSizeLabel().isBlank()) {
            continue;
        }
        ProductSizeOption option = new ProductSizeOption();
        option.setSizeLabel(dto.getSizeLabel().trim());
        option.setPrice(dto.getPrice());
        option.setProduct(product);
        newSizes.add(option);
    }
    if (newSizes.isEmpty()) {
        throw new Exception("Потрібно додати хоча б один розмір та ціну для продукту");
    }

    orderItemRepository.clearSizeOptionForProduct(product);
    product.getSizes().clear();
    product.getSizes().addAll(newSizes);
}

return productRepository.save(product);
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@Override
@Transactional
public void deleteProduct(Long id) throws Exception {
    Product product = productRepository.findById(id)
        .orElseThrow(() -> new Exception("Продукт з id=" + id + " не знайдено"));
    List<OrderStatus> activeStatuses = List.of(
        OrderStatus.PREPARING,
        OrderStatus.DONE
    );

    boolean hasActiveItems =
        orderItemRepository.existsByProductAndOrder_StatusIn(product, activeStatuses);

    if (hasActiveItems) {
        throw new Exception("Неможливо видалити продукт: він використовується в активних замовленнях (PREPARING або DONE).");
    }
    orderItemRepository.clearSizeOptionForProduct(product);
    orderItemRepository.clearProductForProduct(product);
    productRepository.delete(product);
}

@Override
public Product updateProductAvailability(Long id, Boolean inStock) throws Exception {
    Product product = productRepository.findById(id)
        .orElseThrow(() -> new Exception("Продукт з id=" + id + " не знайдено"));
    if (inStock != null) {
        product.setInStock(inStock);
    }
    return productRepository.save(product);
}
}

```

StaffServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateStaffRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.UpdateStaffRequest;
import com.riabko.diploma.restaurant_management_system.models.OrderStatus;
import com.riabko.diploma.restaurant_management_system.models.Staff;
import com.riabko.diploma.restaurant_management_system.repositories.OrderRepository;
import com.riabko.diploma.restaurant_management_system.repositories.StaffRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;

```

```

@Service
public class StaffServiceImpl implements StaffService {
    private final StaffRepository staffRepository;
    private final OrderRepository orderRepository;
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

public StaffServiceImpl(StaffRepository staffRepository, OrderRepository orderRepository) {
    this.staffRepository = staffRepository;
    this.orderRepository = orderRepository;
}

```

```

@Override
public List<Staff> findAll() {
    return staffRepository.findAll();
}

```

```

@Override
public Staff findById(Long id) throws Exception {
    return staffRepository.findById(id)
        .orElseThrow(() -> new Exception("Працівника з id=" + id + " не знайдено"));
}

```

```

@Override
public Staff createStaff(CreateStaffRequest request) throws Exception {
    if (request.getName() == null || request.getName().isBlank()) {
        throw new Exception("Ім'я не може бути порожнім");
    }
    if (request.getPosition() == null || request.getPosition().isBlank()) {
        throw new Exception("Позиція не може бути порожньою");
    }
    Staff staff = new Staff();
    staff.setName(request.getName().trim());
    staff.setPosition(request.getPosition().trim());
    staff.setPhone(request.getPhone() != null ? request.getPhone().trim() : null);
    staff.setActive(request.getActive() == null || request.getActive());
    return staffRepository.save(staff);
}

```

```

@Override
public Staff updateStaff(Long id, UpdateStaffRequest request) throws Exception {
    Staff staff = staffRepository.findById(id)
        .orElseThrow(() -> new Exception("Працівника з id=" + id + " не знайдено"));

    if (request.getName() != null && !request.getName().isBlank()) {
        staff.setName(request.getName().trim());
    }

    if (request.getPosition() != null && !request.getPosition().isBlank()) {
        staff.setPosition(request.getPosition().trim());
    }

    if (request.getPhone() != null) {
        staff.setPhone(request.getPhone().trim());
    }

    if (request.getActive() != null) {
        staff.setActive(request.getActive());
    }
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return staffRepository.save(staff);
    }

    @Override
    @Transactional
    public void deleteStaff(Long id) throws Exception {
        Staff staff = staffRepository.findById(id)
            .orElseThrow(() -> new Exception("Працівника з id=" + id + " не знайдено"));

        List<OrderStatus> activeStatuses = List.of(
            OrderStatus.PREPARING,
            OrderStatus.DONE
        );
        boolean hasActiveOrders = orderRepository.existsByWaiterAndStatusIn(staff, activeStatuses);
        if (hasActiveOrders) {
            throw new Exception("Неможливо видалити працівника: він прив'язаний до активних замовлень (PREPARING або DONE).");
        }
        orderRepository.clearWaiterForStaff(staff);
        staffRepository.delete(staff);
    }
}

```

TableServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateTableRequest;
import com.riabko.diploma.restaurant_management_system.models.OrderStatus;
import com.riabko.diploma.restaurant_management_system.models.RestaurantTable;
import com.riabko.diploma.restaurant_management_system.repositories.OrderRepository;
import com.riabko.diploma.restaurant_management_system.repositories.RestaurantTableRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;

@Service
public class TableServiceImpl implements TableService {
    private final RestaurantTableRepository tableRepository;
    private final OrderRepository orderRepository;
    public TableServiceImpl(RestaurantTableRepository tableRepository, OrderRepository
orderRepository) {
        this.tableRepository = tableRepository;
        this.orderRepository = orderRepository;
    }

    @Override
    public RestaurantTable findById(Long id) throws Exception {
        return tableRepository.findById(id)
            .orElseThrow(() -> new Exception("Стіл з id=" + id + " не знайдено"));
    }

    @Override
    public RestaurantTable createTable(CreateTableRequest request) throws Exception {

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (request.getName() == null || request.getName().isBlank()) {
        throw new Exception("Назва столу не може бути порожньою");
    }
    String name = request.getName().trim();
    if (tableRepository.existsByName(name)) {
        throw new Exception("Стіл з назвою '" + name + "' вже існує");
    }

    RestaurantTable table = new RestaurantTable();
    table.setName(name);
    return tableRepository.save(table);
}

@Override
@Transactional
public void deleteTable(Long id) throws Exception {
    RestaurantTable table = tableRepository.findById(id)
        .orElseThrow(() -> new Exception("Стіл з id=" + id + " не знайдено"));

    List<OrderStatus> activeStatuses = List.of(
        OrderStatus.PREPARING,
        OrderStatus.DONE);

    boolean hasActiveOrders = orderRepository.existsByTableAndStatusIn(table, activeStatuses);
    if (hasActiveOrders) {
        throw new Exception("Неможливо видалити стіл: є активні замовлення (PREPARING або DONE).");
    }
    orderRepository.clearTableForTable(table);
    tableRepository.delete(table);
}

@Override
public List<RestaurantTable> findAll() {
    return tableRepository.findAll();
}
}

```

ReportServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.dto.response.DailySalesSummaryDto;
import com.riabko.diploma.restaurant_management_system.dto.response.SalesByCategoryDto;
import com.riabko.diploma.restaurant_management_system.dto.response.SalesByItemDto;
import com.riabko.diploma.restaurant_management_system.repositories.OrderRepository;
import org.springframework.stereotype.Service;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.List;

```

```

@Service
public class ReportServiceImpl implements ReportService {

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

private final OrderRepository orderRepository;
public ReportServiceImpl(OrderRepository orderRepository) {
    this.orderRepository = orderRepository;
}

private LocalDateTime startOfDay(LocalDate date) {
    return date.atStartOfDay();
}

private LocalDateTime endOfDay(LocalDate date) {
    return date.atTime(LocalTime.MAX);
}

@Override
public List<SalesByCategoryDto> getSalesByCategory(LocalDate startDate, LocalDate endDate) {

    return orderRepository.getSalesByCategory(startOfDay(startDate), endOfDay(endDate));
}

@Override
public List<SalesByItemDto> getSalesByItem(LocalDate startDate, LocalDate endDate) {
    return orderRepository.getSalesByItem(startOfDay(startDate), endOfDay(endDate));
}

@Override
public List<DailySalesSummaryDto> getDailySalesSummary(LocalDate startDate, LocalDate
endDate) {
    return orderRepository.getDailySalesSummary(startOfDay(startDate), endOfDay(endDate));
}
}

```

OrderServiceImpl.java

```

package com.riabko.diploma.restaurant_management_system.services;
import com.riabko.diploma.restaurant_management_system.dto.request.CashCheckoutRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.CreateOrderRequest;
import com.riabko.diploma.restaurant_management_system.dto.request.OrderItemRequest;
import com.riabko.diploma.restaurant_management_system.models.*;
import com.riabko.diploma.restaurant_management_system.repositories.OrderRepository;
import com.riabko.diploma.restaurant_management_system.repositories.ProductRepository;
import com.riabko.diploma.restaurant_management_system.repositories.RestaurantTableRepository;
import com.riabko.diploma.restaurant_management_system.repositories.StaffRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Service
@Transactional

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

public class OrderServiceImpl implements OrderService {
    private final OrderRepository orderRepository;
    private final ProductRepository productRepository;
    private final RestaurantTableRepository restaurantTableRepository;
    private final StaffRepository staffRepository;
    public OrderServiceImpl(OrderRepository orderRepository, ProductRepository productRepository,
        RestaurantTableRepository restaurantTableRepository, StaffRepository
staffRepository) {
        this.orderRepository = orderRepository;
        this.productRepository = productRepository;
        this.restaurantTableRepository = restaurantTableRepository;
        this.staffRepository = staffRepository;
    }

    // Створення нового замовлення
    @Override
    public Order createOrder(CreateOrderRequest request) throws Exception {
        if (request.getOrderType() == null) {
            throw new Exception("Тип замовлення обов'язковий");
        }
        if (request.getItems() == null || request.getItems().isEmpty()) {
            throw new Exception("Список позицій замовлення порожній");
        }

        Order order = new Order();
        order.setOrderType(request.getOrderType());
        order.setStatus(OrderStatus.PREPARING);
        order.setOrderNumber(generateOrderNumber());

        // Стіл для замовлення "На місці"
        if (request.getOrderType() == OrderType.DINE_IN) {
            if (request.getTableId() == null) {
                throw new Exception("Для замовлення \"На місці\" потрібно вибрати стіл");
            }

            RestaurantTable table = restaurantTableRepository.findById(request.getTableId())
                .orElseThrow(() -> new Exception("Стіл з id=" + request.getTableId() + " не
знайдено"));

            // перевірка, чи вже є незакрите замовлення для цього столу
            var activeStatuses = List.of(OrderStatus.PREPARING, OrderStatus.DONE);
            boolean hasActive = orderRepository.existsByTableAndStatusIn(table, activeStatuses);
            if (hasActive) {
                throw new Exception("Для цього столу вже є незакрите замовлення. Відредагуйте
існуюче через \"Список чеків\".");
            }
            order.setTable(table);
        }
        if (request.getOrderType() == OrderType.TAKEAWAY) {
            order.setCustomerName(request.getCustomerName());
        }
    }
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (request.getWaiterId() != null) {
    Staff waiter = staffRepository.findById(request.getWaiterId())
        .orElseThrow(() -> new Exception("Працівника з id=" + request.getWaiterId() + " не
знайдено"));

    if (!waiter.isActive()) {
        throw new Exception("Обраний офіціант не активний");
    }
    if (!"Офіціант".equalsIgnoreCase(waiter.getPosition().trim())) {
        throw new Exception("Обраний працівник не є офіціантом");
    }
    order.setWaiter(waiter);
}

List<OrderItem> orderItems = new ArrayList<>();
BigDecimal totalAmount = BigDecimal.ZERO;

for (OrderItemRequest itemRequest : request.getItems()) {
    if (itemRequest.getProductId() == null) {
        throw new Exception("productId не може бути null");
    }
    if (itemRequest.getQuantity() <= 0) {
        throw new Exception("Кількість повинна бути більшою за 0");
    }

    Product product = productRepository.findById(itemRequest.getProductId())
        .orElseThrow(() -> new Exception("Продукт з id=" + itemRequest.getProductId() + " не
знайдено"));

    ProductSizeOption sizeOption = null;
    String sizeLabel = null;
    BigDecimal unitPrice;
    if (itemRequest.getSizeOptionId() != null) {
        sizeOption = findSizeOptionById(product, itemRequest.getSizeOptionId())
            .orElseThrow(() -> new Exception(
                "Розмір з id=" + itemRequest.getSizeOptionId()
                + " не знайдено для продукту " + product.getName()
            ));
        sizeLabel = sizeOption.getSizeLabel();
        unitPrice = sizeOption.getPrice();
    } else {
        throw new Exception("Для продукту " + product.getName() + " потрібно вибрати
розмір");
    }

    BigDecimal lineTotal = unitPrice.multiply(BigDecimal.valueOf(itemRequest.getQuantity()));
    OrderItem orderItem = new OrderItem();
    orderItem.setOrder(order);
    orderItem.setProduct(product);
    orderItem.setSizeOption(sizeOption);
    orderItem.setProductName(product.getName());
    orderItem.setSizeLabel(sizeLabel);

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        orderItem.setUnitPrice(unitPrice);
        orderItem.setQuantity(itemRequest.getQuantity());
        orderItem.setTotalPrice(lineTotal);
        orderItems.add(orderItem);
        totalAmount = totalAmount.add(lineTotal);
    }
    order.setItems(orderItems);
    order.setTotalAmount(totalAmount);
    return orderRepository.save(order);
}

```

```

@Override
public Order findById(Long id) throws Exception {
    return orderRepository.findById(id)
        .orElseThrow(() -> new Exception("Замовлення з id=" + id + " не знайдено"));
}

```

```

@Override
public List<Order> findAll() {
    return orderRepository.findAll();
}

```

```

@Override
public List<Order> findByTypeAndStatus(OrderType type, OrderStatus status) {
    return orderRepository.findByOrderTypeAndStatus(type, status);
}

```

```

@Override
public Order updateStatus(Long id, OrderStatus newStatus) throws Exception {
    Order order = findById(id);
    // якщо вже сплачено нічого не міняємо
    if (order.getStatus() == OrderStatus.PAID) {
        throw new Exception("Не можна змінювати статус вже сплаченого замовлення");
    }
    // щоб поставити PAID, замовлення має бути DONE
    if (newStatus == OrderStatus.PAID && order.getStatus() != OrderStatus.DONE) {
        throw new Exception("Спочатку кухня має позначити замовлення як \"Виконано\"");
    }
    order.setStatus(newStatus);
    return orderRepository.save(order);
}

```

```

@Override
public Order cashCheckout(Long id, CashCheckoutRequest request) throws Exception {
    Order order = findById(id);
    if (order.getStatus() == OrderStatus.PAID) {
        throw new Exception("Замовлення вже сплачене");
    }
    if (order.getStatus() == OrderStatus.PREPARING) {
        throw new Exception("Замовлення ще готується. Спочатку кухня має позначити \"Виконано\"");
    }
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```

if (request.getPaidAmount() != null && order.getTotalAmount() != null) {
    if (request.getPaidAmount().compareTo(order.getTotalAmount()) < 0) {
        throw new Exception("Оплачена сума менша за суму замовлення");
    }
}
order.setStatus(OrderStatus.PAID);
return orderRepository.save(order);
}

@Override
public Order updateOrder(Long id, CreateOrderRequest request) throws Exception {
    Order order = findById(id);

    // Редагувати вже сплачене замовлення заборонено
    if (order.getStatus() == OrderStatus.PAID) {
        throw new Exception("Неможливо редагувати вже сплачене замовлення");
    }
    if (request.getItems() == null || request.getItems().isEmpty()) {
        throw new Exception("Список позицій замовлення порожній");
    }
    //Повністю перезбираємо позиції замовлення
    List<OrderItem> orderItems = new ArrayList<>();
    BigDecimal totalAmount = BigDecimal.ZERO;

    for (OrderItemRequest itemRequest : request.getItems()) {
        Product product = productRepository.findById(itemRequest.getProductId())
            .orElseThrow(() -> new Exception("Продукт з id=" + itemRequest.getProductId() + " не знайдено"));

        ProductSizeOption sizeOption = null;
        String sizeLabel = null;
        BigDecimal unitPrice;

        if (itemRequest.getSizeOptionId() != null) {
            var opt = product.getSizes().stream()
                .filter(s -> s.getId().equals(itemRequest.getSizeOptionId()))
                .findFirst()
                .orElseThrow(() -> new Exception("Розмір не знайдено для продукту " +
product.getName()));
            sizeOption = opt;
            sizeLabel = opt.getSizeLabel();
            unitPrice = opt.getPrice();
        } else {
            throw new Exception("Для продукту " + product.getName() + " потрібно вибрати розмір");
        }

        BigDecimal lineTotal = unitPrice.multiply(BigDecimal.valueOf(itemRequest.getQuantity()));
        OrderItem orderItem = new OrderItem();
        orderItem.setOrder(order);
        orderItem.setProduct(product);
        orderItem.setSizeOption(sizeOption);
    }
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк, Т. А.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        orderItem.setProductName(product.getName());
        orderItem.setSizeLabel(sizeLabel);
        orderItem.setUnitPrice(unitPrice);
        orderItem.setQuantity(itemRequest.getQuantity());
        orderItem.setTotalPrice(lineTotal);
        orderItems.add(orderItem);
        totalAmount = totalAmount.add(lineTotal);
    }
    order.getItems().clear();
    order.getItems().addAll(orderItems);
    order.setTotalAmount(totalAmount);

    return orderRepository.save(order);
}

private Optional<ProductSizeOption> findSizeOptionById(Product product, Long sizeOptionId) {
    if (product.getSizes() == null) {
        return Optional.empty();
    }
    return product.getSizes()
        .stream()
        .filter(s -> sizeOptionId.equals(s.getId()))
        .findFirst();
}

private String generateOrderNumber() {
    return UUID.randomUUID().toString().substring(0, 8).toUpperCase();
}
}

```

		Рябко О. Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		101