

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

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА КЛІЄНТ-СЕРВЕРНОЇ POS-СИСТЕМИ РЕСТОРАНУ
«AVERO» З ВИКОРИСТАННЯМ C# ТА ASP.NET CORE**

**DEVELOPMENT OF A CLIENT-SERVER POS SYSTEM FOR THE
RESTAURANT «AVERO» USING C# AND ASP.NET CORE**

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

Виконав: здобувач вищої освіти
групи ІПЗ-41
Ткачук Микола Анатолійович

Керівник:
Гулієва Наталія Михайлівна

Кваліфікаційну роботу
допущено до захисту
«__» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Ткачуку Миколі Анатолійовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка клієнт-серверної POS-системи ресторану «Avero» з використанням C# та ASP.NET Core».

Керівник роботи: к.т.н., доцент Гулієва Н. М.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: Visual Studio Code, Visual Studio, C#, ASP.NET Core, MySQL, методичні вказівки до виконання кваліфікаційної роботи бакалавра.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): розробка інтерфейсу POS-системи з каталогом меню і адміністративною панеллю керування; збереження замовлень та даних у локальній базі даних; створення функціоналу для відображення та редагування меню на сайті через POS-систему, адаптація інтерфейсу для різних екранів та платформ.

5. Перелік графічного матеріалу: 17 рисунків, 3 таблиці, 13 лістингів, 3 додатки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Гулієва Н. М.</i>		
Специфікація вимог до розробленої системи	<i>Гулієва Н. М.</i>		
Розробка об'єкта проектування	<i>Гулієва Н. М.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	<u> </u> %		
Академічна доброчесність	<i>Гулієва Н. М.</i>		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Микола ТКАЧУК

Керівник кваліфікаційної роботи _____

Наталія ГУЛІЄВА

АНОТАЦІЯ

Ткачук М. А. Розробка клієнт-серверної POS-системи ресторану «Avero» з використанням C# та ASP.NET Core. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

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

У першому розділі здійснено аналіз сучасних POS-систем для бізнесів маленького та середнього сегменту, виявлено їхні недоліки та визначено технічні проблеми при використанні таких застосунків. Сформульовано завдання на розробку власної POS-системи з вектором клієнт серверної архітектури. У другому розділі обґрунтовано вибір архітектури, засобів реалізації, технологій, створено UML-діаграми, визначено функціональні, технічні, нефункціональні вимоги та завдання до системи. У третьому розділі реалізовано практичну частину POS-системи: побудовано інтерфейс POS-системи, інтерфейс сайту меню, організовано взаємодію та бізнес-логіку між компонентами, створено базу даних, а також проведено тестування й формування інсталяційного пакета.

Ключові слова: C#, ASP.NET Core, JavaScript, WinForms, MySQL, інтерфейс, ресторан, сайт, меню, QR-код, адаптивність, ООП.

ABSTRACT

Tkachuk M. A. Development of a client-server POS system for the restaurant “Avero” using C# and ASP.NET Core. Manuscript. Bachelor’s qualification thesis of the educational program “Software Engineering” of the specialty “Software Engineering”. Lutsk National Technical University. Lutsk, 2026.

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

The first chapter analyzes modern POS systems for small and medium-sized businesses, identifies their shortcomings, and determines technical problems arising during the use of such applications. The task for developing a proprietary POS system based on a client-server architecture and formulated. The second chapter substantiates the choice of architecture, implementation tools, and technologies; UML diagrams are created; functional, technical, and non-functional requirements of the system are defined. The third chapter presents the practical implementation of the POS system: the POS system interface and menu website interface are developed, interaction and business logic between components are organized, a database is created, and testing and installation package generation are carried out.

Keywords: C#, ASP.NET Core, JavaScript, WinForms, MySQL, interface, restaurant, website, menu, QR code, adaptability, OOP.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ POS-СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ РЕСТОРАННОГО БІЗНЕСУ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ	10
1.1 Аналіз сучасного стану проблеми	10
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	13
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	15
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення	15
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання	18
РОЗДІЛ 3 РОЗРОБКА КЛІЄНТ-СЕРВЕРНОЇ POS-СИСТЕМИ РЕСТОРАНУ «AVERO» З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ C# ТА ASP.NET CORE	20
3.1 Практична реалізація об'єкта проєктування	20
3.2 Проєктування та реалізація бази даних	40
3.3 Тестування та налагодження інформаційно-комп'ютерної системи	44
3.4 Розробка інсталяційного пакета	47
ВИСНОВКИ	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53
ДОДАТКИ	55

ВСТУП

Актуальність теми. У сучасних умовах розвитку інформаційних технологій автоматизація бізнес-процесів стає важливою складовою ефективного функціонування підприємств різних сфер діяльності. Особливо актуальною зазначена проблема є для закладів ресторанного бізнесу, оскільки швидкість обслуговування, точність обробки і виконання замовлень, контроль продажів та якість взаємодії з клієнтами безпосередньо впливають на конкурентоспроможність закладу та рівень прибутку.

Сучасний ресторанний бізнес дедалі активніше впроваджує цифрові технології для автоматизації процесів прийому замовлень, ведення обліку, формування звітності та організації роботи персоналу. Одним із ключових інструментів автоматизації є POS-системи, які дозволяють оптимізувати роботу закладу, покращити взаємодію між персоналом та клієнтами, а також забезпечити централізоване керування бізнес-процесами.

Проте значна частина сучасних POS-систем орієнтована переважно на великі мережі ресторанів та закладів харчування, що супроводжується високою вартістю використання, складністю налаштування та надлишковим функціоналом. У результаті малий та середній ресторанний бізнес потребує більш доступних, гнучких та адаптивних рішень, які б поєднували необхідний функціонал із простотою використання.

У цьому контексті є дуже важливим актуальність розробки клієнт-серверної POS-системи ресторану «Avero», яка поєднуватиме десктоп-застосунок для персоналу, вебінтерфейс меню для клієнтів, систему QR-кодів та механізм онлайн замовлень. Завдяки використанню технологій C#, ASP.NET Core, WinForms, JavaScript та MySQL дозволяє реалізувати сучасну систему для автоматизації ресторанного бізнесу з гнучкою архітектурою, високою продуктивністю та можливістю масштабування.

Метою кваліфікаційної роботи бакалавра є розробка клієнт-серверної POS-системи ресторану «Avero» з використанням C# та ASP.NET Core, яка

забезпечить автоматизацію процесів обслуговування клієнтів, управління замовленнями, формування звітності та взаємодію між десктоп і вебкомпонентами системи.

Об'єктом дослідження є процес автоматизації ресторанного бізнесу із використанням сучасних технологій.

Предметом дослідження є особливості розробки клієнт-серверної POS-системи для ресторанного бізнесу із використанням технологій C#, ASP.NET Core, WinForms, JavaScript та MySQL.

Для досягнення поставленої мети в межах кваліфікаційної роботи бакалавра необхідно виконати наступні технічні та функціональні завдання:

- визначення функціональних та нефункціональних вимог до POS-системи;
- аналіз та вибір технологій і засобів для реалізації клієнт-серверної архітектури системи;
- розробка десктоп-застосунку для персоналу ресторану та вебінтерфейсу меню для клієнтів;
- створення системи QR-кодів і реалізація REST API для взаємодії між компонентами системи;
- проєктування та реалізація бази даних, механізму формування замовлень, чеків і звітності;
- проведення комплексного тестування функціоналу, продуктивності та стабільності роботи системи;
- формування інсталяційного пакета для подальшого використання та розгортання системи.

Практичне використання роботи полягає у розробці клієнт-серверної POS-системи ресторану «Avero» з використанням сучасних технологій програмної розробки, що дозволяє створити ефективне, гнучке та масштабоване рішення для автоматизації ресторанного бізнесу. У межах кваліфікаційної роботи бакалавра реалізовано повний цикл створення системи: від аналізу предметної області та

проєктування архітектури до реалізації, тестування та підготовки програмного продукту до використання.

РОЗДІЛ 1

АНАЛІЗ СУЧАСНИХ POS-СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ РЕСТОРАННОГО БІЗНЕСУ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

На сьогоднішній день інформаційні технології розвиваються надзвичайно швидкими темпами, що зумовлює потребу у створенні сучасних, гнучких та ефективних програмних рішень. Процеси цифровізації активно впроваджуються в різні сфери діяльності людини, зокрема й у сферу ресторанного бізнесу. Автоматизація процесів обслуговування, ведення обліку, формування замовлень та взаємодії з клієнтами стає важливим елементом забезпечення якісного сервісу та конкурентоспроможності закладів ресторанного господарства.

У сучасних умовах розвитку технологій зростають не лише вимоги до швидкості та якості обслуговування, а й очікування клієнтів щодо зручності користування сервісами. Саме тому використання POS-систем у ресторанному бізнесі стає невід'ємною частиною ефективної організації роботи закладів.

Особливо актуально дана проблема являється для ринку малого та середнього бізнесу, оскільки такі заклади потребують функціональних, доступних та зручних у використанні систем автоматизації. Більшість сучасних POS-систем орієнтовані переважно на великі мережі ресторанів та закладів харчування, що супроводжується високою вартістю впровадження, надлишковим функціоналом та складністю налаштування [13]. У результаті невеликі кафе, піцерії чи місцеві ресторани не завжди мають можливість використовувати дорогі комерційні рішення або ж не потребують складних систем із великою кількістю додаткових модулів [14].

Серед найбільш популярних систем автоматизації ресторанного бізнесу варто виділити такі рішення, як Poster POS (рис. 1.1), R-Keeper, Syrve(iiko), Toast POS (рис. 1.2) та інші [15].

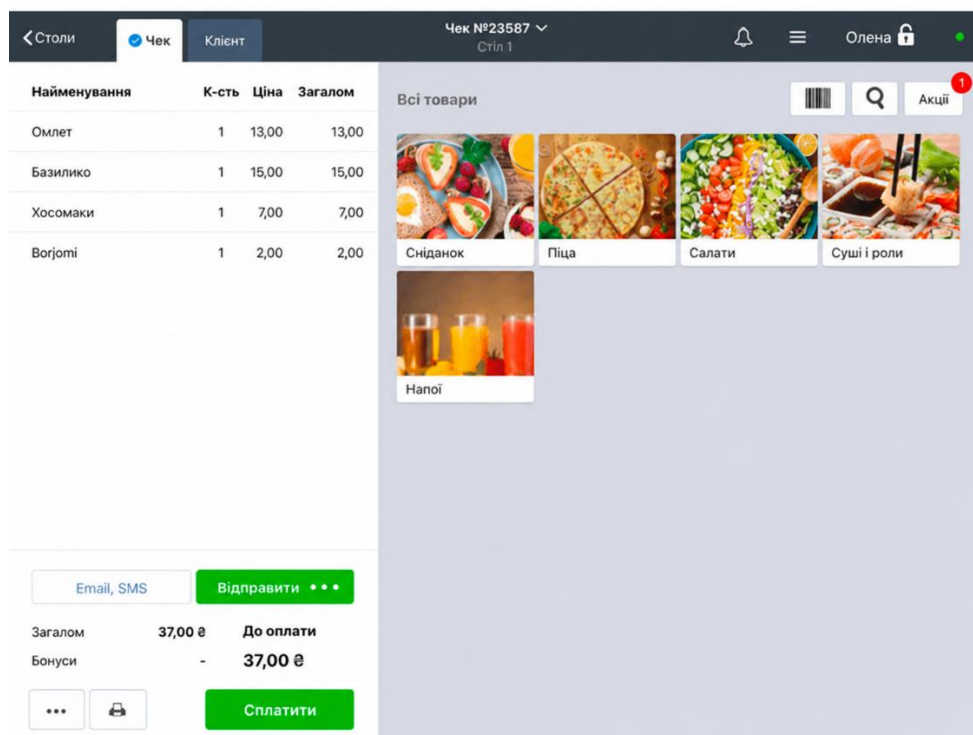


Рисунок 1.1 – Вигляд Poster POS [13]

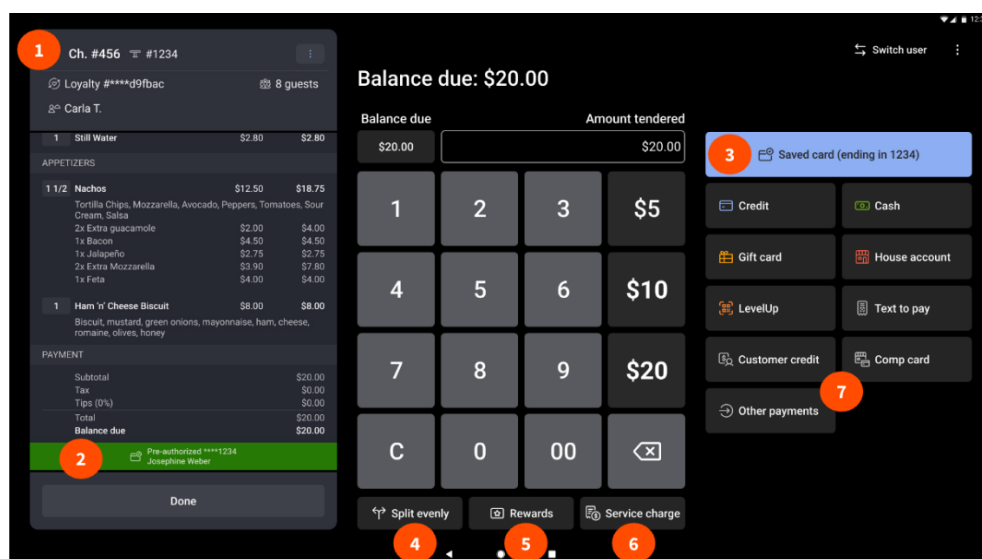


Рисунок 1.2 – Меню Toast POS [9]

Дані системи забезпечують широкий функціонал для ведення замовлень, роботи з персоналом, аналітики продажів та адміністрування закладу. Проте аналіз сучасних рішень показує, що частина з них має складний інтерфейс, потребує постійної підписки, значних витрат на підтримку або ж є недостатньо гнучкою для адаптації під потреби малого та середнього бізнесу.

У таблиці 1.1 наведено порівняння таких сучасних POS-систем як Poster POS, LiqPay POS, r-keeper.

Таблиця 1.1 – Порівняння сучасних POS-систем

Характеристика	Poster POS	LiqPay POS/ SmartTouch POS	R_keeper	POS-система «Avero»
Тип розгортання	Хмарна	Локальна/хмарна	Корпоративна	Клієнт-серверна
Десктоп-застосунок	Частково	Так	Так	Так
Вебменю	Так	Обмежено	Опціонально	Так
Онлайн-замовлення	Так	Частково	Так	Так
QR-коди для столів	Так	Частково	Так	Так
Інтеграція оплат	Так	Так	Так	Так
Звітність	Так	Так	Так	Так
Гнучке редагування меню	Так	Так	Так	Так
Вартість впровадження	Середня/підписка	Середня	Висока	Низька
Орієнтація на малий бізнес	Так	Частково	Частково	Так
Можливість модифікації коду	Ні	Ні	Ні	Так
Повний контроль над системою	Ні	Частково	Ні	Так

У таблиці 1.1 наведено порівняння найбільш поширених POS-систем, що використовуються в ресторанному бізнесі та розробленої системи «Avero».

Проведений аналіз показав, що сучасні комерційні рішення мають широкий функціонал, однак часто супроводжуються високою вартістю впровадження, щомісячною абонентською платою та обмеженими можливостями кастомізації. Більшість систем не надає доступу до вихідного коду, що унеможливорює адаптацію програмного забезпечення під специфічні потреби конкретного закладу.

У зв'язку з цим виникає потреба у створенні сучасної, зручної та гнучкої POS-системи, яка б поєднувала необхідний функціонал для автоматизації ресторанного бізнесу з доступністю та простотою використання.

POS-система «Avero» розробляється з урахуванням потреб закладів малого та середнього сегменту ресторанного бізнесу та пропонує комплексне рішення

для автоматизації роботи кафе, ресторанів, піцерій та інших закладів. Система забезпечує можливість зручного керування меню, редагування позицій, формування чеків, ведення статистики продажів, створення звітності по категоріях, персоналу та замовленнях.

Крім десктоп-частини, система містить вебінтерфейс меню із підтримкою онлайн замовлень та інтеграцією QR-кодів, що дозволяє клієнтам швидко отримати доступ до меню та формувати замовлення безпосередньо зі смартфона. Додатковою перевагою системи є редагування вмісту сайту без необхідності глибоких знань програмування, що робить систему зручною не лише для клієнтів ресторану чи кафе так і для персоналу закладу і їх власників.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Було проведено аналіз предметної області та сучасного стану автоматизації ресторанного бізнесу із використанням POS-систем. У результаті було визначено основні проблеми сучасних рішень, серед яких висока вартість використання, складність налаштування, надлишковий функціонал та недостатня адаптивність для потреб малого та середнього бізнесу.

Для реалізації поставленої мети в межах кваліфікаційної роботи бакалавра необхідно виконати наступні технічні та функціональні завдання:

- визначення функціональних та нефункціональних вимог до POS-системи;
- аналіз та вибір технологій і засобів для реалізації клієнт-серверної архітектури системи;
- розробка десктоп-застосунку для персоналу ресторану та вебінтерфейсу меню для клієнтів;
- створення системи QR-кодів і реалізація REST API для взаємодії між компонентами системи;
- проєктування та реалізація бази даних, механізму формування замовлень, чеків і звітності;

- проведення комплексного тестування функціоналу, продуктивності та стабільності роботи системи;
- формування інсталяційного пакета для подальшого використання та розгортання системи.

Таким чином, виконання поставлених завдань дозволить реалізувати сучасну клієнт-серверну POS-систему для автоматизації ресторанного бізнесу, яка забезпечить зручне керування замовленнями, меню, звітністю та взаємодією між персоналом і клієнтами. Розроблена система повинна відповідати сучасним вимогам до продуктивності, функціональності, масштабованості та зручності використання.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

У процесі розробки клієнт-серверної POS-системи для ресторану «Авего» виникає потреба у детальному проєктуванні архітектури програмного забезпечення та визначенні основних функціональних можливостей системи. Правильне проєктування програмного забезпечення дозволяє забезпечити стабільність роботи системи, масштабованість, зручність підтримки, а також ефективну взаємодію між десктоп та вебкомпонентами.

Розроблювана система повинна забезпечувати автоматизацію основних процесів ресторанного бізнесу, зокрема: керування та налаштування меню, формування замовлень, створення чеків, введення статистики продажів, взаємодію між персоналом закладу та клієнтів, а також підтримку онлайн замовлень через вебінтерфейс.

Також важливим етапом проєктування системи є визначення структури взаємодії між окремими компонентами програмного забезпечення та забезпечення цілісності даних. Для цього у системі використовується реляційна база даних MySQL, яка забезпечує збереження інформації про меню, замовлення, столи, чеки, користувачів та платежі. Взаємодія між клієнтською частиною, сервером та базою даних реалізується через REST API, що дозволяє забезпечити централізовану обробку даних, спростити масштабування системи та забезпечити стабільну роботу програмного забезпечення в умовах одночасної роботи декількох користувачів.

Особливістю системи є використання клієнт-серверної архітектури, де десктоп-застосунок виконує роль клієнтської частини для персоналу закладу, а ASP.NET Core Web API забезпечує обробку запитів, бізнес-логіку та взаємодію з базою даних. Крім цього, система містить вебінтерфейс меню із підтримкою QR-кодів для доступу клієнтів до функціоналу онлайн-сервісу (рис. 2.1).

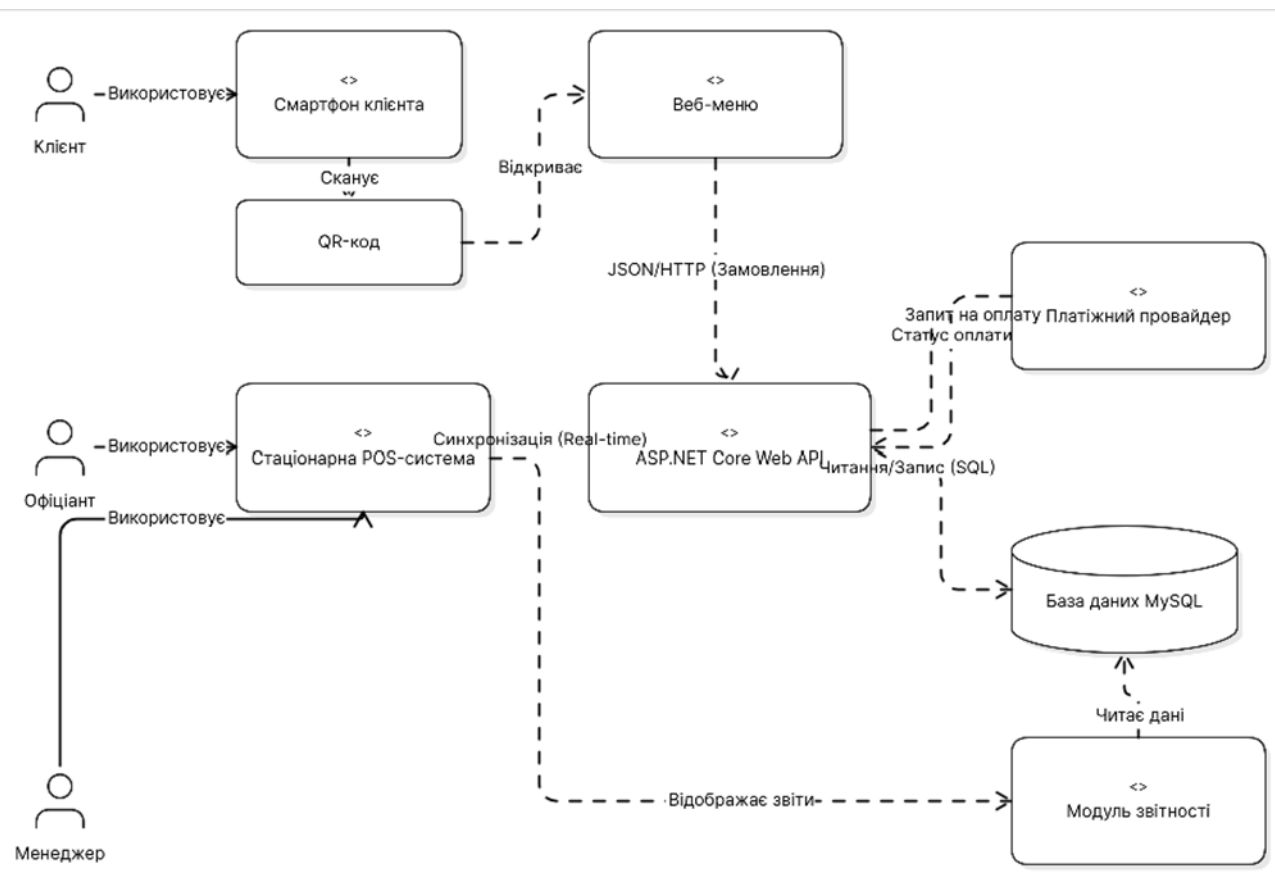


Рисунок 2.1 – UML діаграма зв'язків системи

На даній діаграмі відображено основні компоненти програмного забезпечення та взаємозв'язки між ними. До складу системи входять десктоп-застосунок для персоналу ресторану, вебінтерфейс меню для клієнтів, серверна частина, система керування базами даних, модуль генерації QR-кодів та зовнішній платіжний сервіс [1].

Клієнт взаємодіє із системою шляхом сканування QR-коду, після чого відкривається вебінтерфейс меню. Вебінтерфейс надсилає запити до серверної частини, яка обробляє їх та виконує операції читання і запису до бази даних. Десктоп-застосунок персоналу також взаємодіє з сервером через REST API, отримуючи інформацію про меню, замовлення, чеки та платежі.

У разі здійснення оплати серверна частина формує запит до зовнішнього платіжного провайдера та отримує інформацію про статус транзакції. Модуль

звітності використовує дані з бази даних для формування аналітичної інформації щодо продажів та роботи персоналу.

Таким чином, UML-діаграма компонентів наочно демонструє архітектуру системи та логіку взаємодії між її основними складовими.

У таблиці 2.1 наведено опис основних об'єктів діаграми використання системи.

Таблиця 2.1 – Опис об'єктів діаграми використання.

Об'єкт	Опис
Адміністратор	Користувач системи, який має доступ до керування меню, столами, персоналом, звітністю та налаштуванням системи
Офіціант	Користувач системи, який створює замовлення, працює із чеками та обслуговує клієнтів
Клієнт	Відвідувач закладу, який взаємодіє із вебменю через QR-код та може створювати онлайн-замовлення
Перегляд меню	Надає можливість переглядати список доступних страв та категорій
Формування замовлення	Дозволяє додавати позиції меню до замовлення, змінювати кількість та створювати чек
Керування столами	Забезпечує зміну статусів столів їх редагування та контроль зайнятості
Генерація QR-коду	Дозволяє створювати QR-коди для швидкого переходу до вебменю
Онлайн замовлення	Забезпечує створення замовлень через вебінтерфейс
Формування звітності	Дозволяє отримувати статистику продажів, чеків та роботи персоналу
Робота з базою даних	Забезпечує збереження, редагування та отримання інформації із системи

На основі проведеного аналізу предметної області було сформовано функціональні, нефункціональні та технічні вимоги до розроблюваної POS-системи. Серед основних функціональних вимог варто виділити підтримку роботи із замовленнями, чеками, меню, QR-кодами та вебзамовленнями. До нефункціональних вимог належать стабільність роботи системи, адаптивність інтерфейсу, продуктивність та зручність використання. Технічні вимоги передбачають використання сучасних технологій розробки програмного забезпечення, зокрема C#, ASP.NET Core, WinForms, JavaScript та MySQL.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для реалізації клієнт-серверної POS-системи ресторану «Avero» необхідно було обрати сучасні технології та засоби розробки, які б забезпечували стабільність роботи системи, високу продуктивність, гнучкість у масштабуванні та зручність для подальшої підтримки програмного забезпечення.

Під час аналізу існуючих технологій було розглянуто різні підходи до побудови десктоп та вебзастосунків, а також засоби для реалізації серверної частини системи та організації взаємодії з базою даних. Основна увага приділялась технологіям, які дозволяють ефективно реалізувати клієнт-серверну архітектуру, забезпечити стабільну взаємодію між компонентами системи та підтримувати подальше розширення функціональних можливостей програмного забезпечення.

Для реалізації десктоп-частини POS-системи було обрано мову програмування C# та технологію WinForms [11, 19]. Дане рішення обумовлене високою продуктивністю технології, зручною роботою з графічним інтерфейсом та підтримкою швидкої розробки застосунків для операційної системи Windows. Використання WinForms дозволяє створити інтуїтивно зрозумілий інтерфейс для персоналу ресторану, реалізувати роботу із замовленнями, меню, чеками та іншими елементами системи в режимі реального часу.

Серверна частина системи реалізована із використанням ASP.NET Core Web API [12]. Використання даної технології дозволяє організувати обробку HTTP-запитів, реалізувати бізнес-логіку системи та забезпечити взаємодію між десктоп-застосунком, вебінтерфейсом та базою даних. ASP.NET Core забезпечує високу швидкодію, підтримку масштабування та стабільну роботу серверної частини навіть при одночасній роботі декількох користувачів. Для організації взаємодії між компонентами системи використано REST API [17], що дозволяє централізовано обробляти запити та забезпечувати обмін даними між клієнтською та серверною частинами системи.

Для збереження та обробки інформації використано систему керування базами даних MySQL [10]. Дане рішення забезпечує підтримку реляційних зв'язків між таблицями, ефективну роботу із великими обсягами даних та стабільне збереження інформації. Використання MySQL також дозволяє організувати структуру бази даних із розділенням даних між окремими компонентами системи для підвищення рівня безпеки та зручності адміністрування.

Для реалізації вебінтерфейсу меню використано HTML, CSS та JavaScript [6, 3, 8]. Дані технології дозволяють створити адаптивний вебінтерфейс із підтримкою онлайн-замовлень, інтеграцією QR-кодів та можливістю швидкого доступу клієнтів до меню ресторану через мобільні пристрої. Використання JavaScript забезпечує динамічне оновлення елементів сторінки та взаємодію із серверною частиною системи без необхідності перезавантаження сторінки.

Для реалізації системи QR-кодів використано бібліотеку QRCode [5], що дозволяє автоматично генерувати QR-коди для столів ресторану та забезпечувати швидкий перехід клієнтів до вебменю системи. Використання QR-кодів значно спрощує взаємодію клієнтів із системою та дозволяє автоматизувати процес створення онлайн-замовлень [21].

У якості середовища розробки було використано Visual Studio [18], що забезпечує зручну роботу із проєктом, підтримку налагодження програмного забезпечення, інтеграцію із системою керування базами даних та засобами тестування. Для створення інсталяційного пакета використано Inno Setup [7], що дозволяє автоматизувати процес встановлення програмного забезпечення та спростити розгортання системи на робочих місцях персоналу ресторану.

Таким чином, обраний набір технологій та засобів розробки дозволяє реалізувати сучасну клієнт-серверну POS-систему із високою продуктивністю, зручним інтерфейсом, підтримкою масштабування та можливістю подальшого розширення функціональних можливостей системи.

РОЗДІЛ 3

РОЗРОБКА КЛІЄНТ-СЕРВЕРНОЇ POS-СИСТЕМИ РЕСТОРАНУ «AVERO» З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ C# ТА ASP.NET CORE

3.1 Практична реалізація об'єкта проєктування

Практична реалізація кваліфікаційної роботи бакалавра полягала у розробці клієнт-серверної POS-системи для ресторану «Avero», призначеної для автоматизації основних бізнес-процесів закладу ресторанного господарства. У процесі розробки було створено комплексне програмне рішення, що складається з десктоп-застосунку для персоналу, серверної частини у вигляді Web API, вебінтерфейсу меню для клієнтів та реляційної бази даних.

Архітектура системи побудована за клієнт-серверним принципом [20]. Де клієнтська частина представлена десктоп-застосунком, розробленим мовою C# із використанням технології WinForms [19]. Даний застосунок забезпечує роботу офіціантів та адміністраторів із меню, замовленнями, чеками, столами, звітністю та налаштуваннями системи.

Серверна частина реалізована із використанням ASP.NET Core Web API. Вона забезпечує централізовану обробку HTTP-запитів, реалізацію бізнес-логіки та взаємодію з базою даних. Через REST API здійснюється обмін даними між десктоп-застосунком, вебінтерфейсом та сервером.

Для збереження інформації використовується система керування базами даних MySQL [10]. У базі даних зберігаються відомості про меню, категорії та підкатегорії страв, столи, замовлення, чеки, платежі, користувачів, вебзамовлення та QR-коди.

Окремим компонентом системи є вебінтерфейс меню, розроблений із використанням HTML, CSS та JavaScript [6]. Вебкомпонент системи забезпечує можливість перегляду меню, формування онлайн-замовлень, перегляду поточного замовлення та переходу до сторінки оплати.

Для швидкого доступу клієнтів до вебменю реалізовано систему QR-кодів. Для кожного столу генерується унікальний QR-код, який містить посилання з

токемом доступу. Після сканування QR-коду клієнт автоматично переходить до меню закладу прив'язаного до відповідного столу.

У процесі реалізації системи було розроблено такі основні модулі десктоп-застосунку:

- MenuForm – модуль роботи з меню та додавання позицій до замовлення;
- TableForm – модуль керування столами та відображення їх поточного стану;
- OrdersForm – модуль перегляду активних замовлень;
- CheckForm – модуль формування та закриття чеків;
- WebOrderForm – модуль для обробки вебзамовлень;
- ReportForm – модуль формування аналітичної звітності;
- AddForm та EditTableForm – модулі редагування довідкової інформації.

Практична реалізація системи здійснювалася поетапно: від проектування архітектури та структури бази даних до створення користувацького інтерфейсу, реалізації бізнес-логіки, тестування та підготовки інсталяційного пакета.

Таким чином, у межах кваліфікаційної роботи бакалавра було реалізовано повноцінну клієнт-серверну POS-систему для ресторану «Avero», яка забезпечує автоматизацію основних процесів ресторанного бізнесу та може бути використана як готове програмне рішення для закладів малого та середнього бізнесу, нижче наведено вигляд POS-системи на рисунку 3.1 та рисунку 3.2.

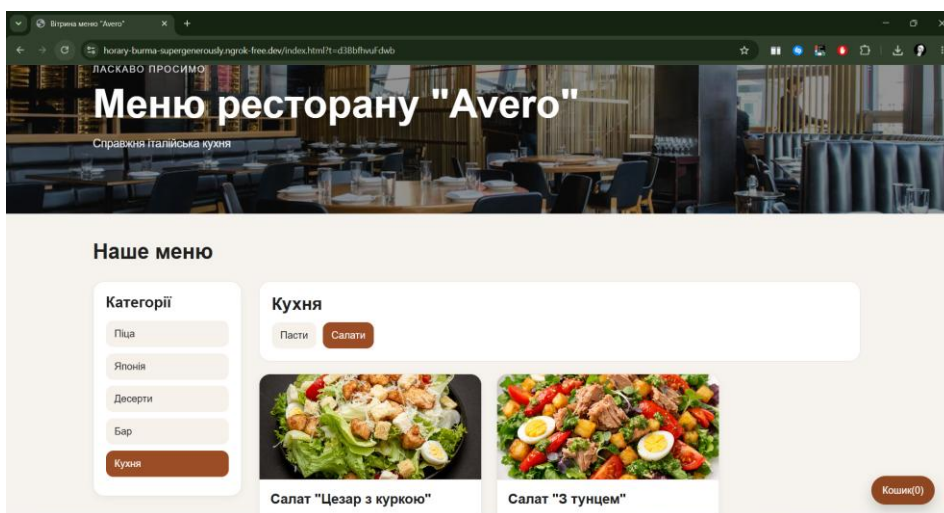


Рисунок 3.1 – Вигляд POS-системи на сайті

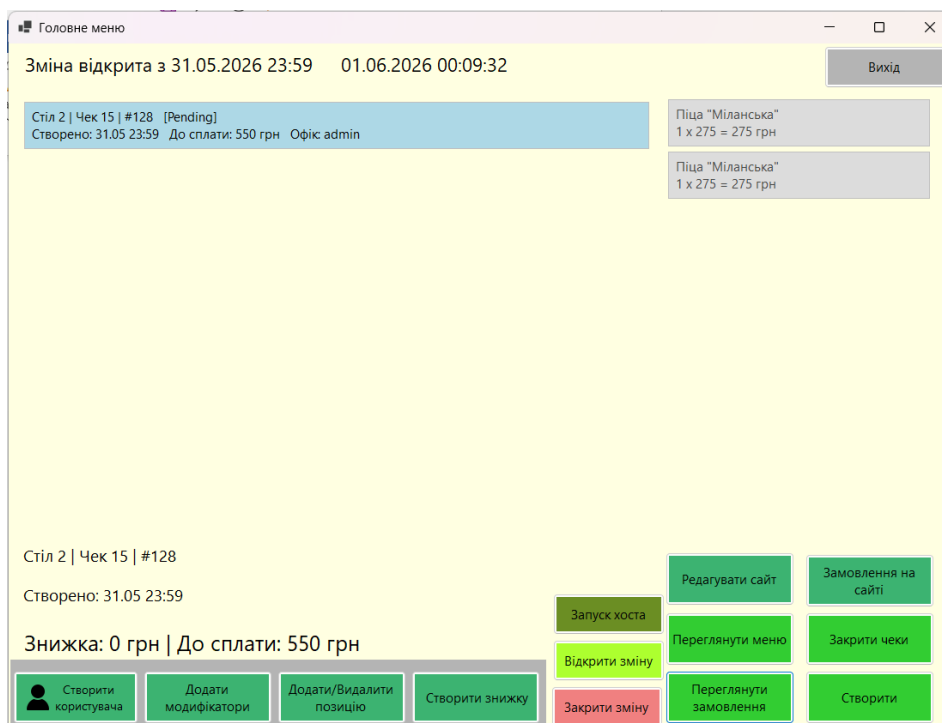


Рисунок 3.2 – AdminForm POS-системи

Серверна частина POS-системи ресторану «Avero» реалізована із використанням ASP.NET Core Web API [12]. Основним призначенням серверної частини є централізована обробка запитів від десктоп-застосунку та вебінтерфейсу, реалізація бізнес-логіки та забезпечення взаємодії з базою даних. Використання ASP.NET Core дозволяє забезпечити високу продуктивність, стабільність роботи системи та можливість подальшого масштабування програмного забезпечення.

Архітектура серверної частини побудована відповідно до принципів REST [17]. Кожен функціональний модуль системи представлений окремим контролером, який містить HTTP-методи для отримання, створення, оновлення та видалення даних. Такий підхід дозволяє логічно розділити функціонал системи, спростити підтримку програмного забезпечення та забезпечити зручну взаємодію між окремими компонентами системи.

У складі серверної частини реалізовано такі основні контролери:

- MenuController;
- OrdersController;

- ChecksController;
- WebOrderController;
- PaymentsController;
- TablesController;
- ReportController;
- QrController.

Контролер MenuController відповідає за отримання, створення та редагування елементів меню ресторану. OrdersController забезпечує формування та обробку замовлень, а також взаємодію між персоналом ресторану та клієнтською частиною системи. ChecksController реалізує механізми створення та закриття чеків, обробку оплат та формування підсумкових даних замовлення. WebOrderController відповідає за обробку онлайн-замовлень, отриманих через вебінтерфейс меню, код обробки вебзамовлень наведено в додатку А. PaymentsController реалізує взаємодію із механізмом оплати та обробку інформації про платежі. TablesController забезпечує роботу із столами ресторану та їх поточними статусами, а ReportController відповідає за формування звітності та статистики системи. Для роботи із QR-кодами використовується окремий QrController, який реалізує генерацію QR-кодів та механізм прив'язки клієнтів до столів ресторану.

Обмін даними між компонентами системи здійснюється у форматі JSON. Сервер забезпечує перевірку вхідних даних, виконання бізнес-правил та повернення результатів у стандартизованому форматі. Це дозволяє забезпечити стабільну взаємодію між серверною частиною, десктоп-застосунком та вебінтерфейсом системи. Нижче наведено фрагмент коду для обробки вебзамовлень (ліст. 3.1).

Лістинг 3.1 – Фрагмент контролера для обробки вебзамовлень

```
[HttpPost]
public IActionResult CreatewebOrder([FromBody] webOrderModel
model)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);
```

```

        using var connection = new MySqlConnection(connectionString);
        connection.Open();

        string query = @"INSERT INTO web_orders
                        (TableID, CustomerName, Comment, Status,
CreatedAt)
                        VALUES
                        (@TableID, @CustomerName, @Comment, 'New',
NOW())";

        using var command = new MySqlCommand(query, connection);

        command.Parameters.AddWithValue("@TableID", model.TableID);
        command.Parameters.AddWithValue("@CustomerName",
model.CustomerName);
        command.Parameters.AddWithValue("@Comment", model.Comment);

        command.ExecuteNonQuery();

        return Ok(new { message = "Замовлення створено успішно" });
}

```

кінець лістингу 3.1

Даний метод забезпечує створення нового вебзамовлення, перевірку коректності вхідних даних та запис інформації у базу даних. Після успішного виконання запиту сервер повертає відповідь клієнту у форматі JSON.

Для забезпечення взаємодії клієнтів із вебменю через QR-коди реалізовано механізм отримання інформації про стіл за унікальним токеном (ліст. 3.2). Завдяки такому механізму сервер передає актуальні дані про замовлення, статус, номер замовлення, номер столу, загальну суму, кількість позицій, та дозволяє взаємодіяти з цим замовленням.

Лістинг 3.2 – Отримання інформації про стіл через QR-код

```

[HttpGet("resolve-table")]
public IActionResult ResolveTable(string token)
{
    using var connection = new MySqlConnection(connectionString);
    connection.Open();

    string query = @"SELECT TableID, Name
                    FROM table_qr
                    WHERE Token = @Token";

    using var command = new MySqlCommand(query, connection);

    command.Parameters.AddWithValue("@Token", token);

```

```

using var reader = command.ExecuteReader();
if (reader.Read())
{
    return Ok(new
    {
        tableId = reader["TableID"],
        tableName = reader["Name"]
    });
}
return NotFound();
}

```

кінець лістингу 3.2

Реалізований серверний функціонал дозволяє забезпечити стабільну взаємодію між усіма компонентами системи, централізовано обробляти інформацію та підтримувати роботу як внутрішньої POS-системи, так і вебінтерфейсу для клієнтів ресторану.

Десктоп-застосунок POS-системи реалізований мовою програмування C# із використанням технології WinForms. Даний застосунок побудований за модульним принципом, відповідно до принципів ООП та SOLID [4]. Такий підхід дозволяє спростити підтримку програмного забезпечення, забезпечити зручність подальшого розширення функціоналу системи та підвищити стабільність роботи окремих компонентів. Використання WinForms дозволило реалізувати швидкий та зручний графічний інтерфейс для роботи персоналу ресторану, а також забезпечити ефективну взаємодію із серверною частиною системи.

Вхід та авторизація в застосунок відповідає LoginForm, де здійснюється авторизація користувача. В системі існують ролі такі як «staff» та «admin» і кожна з цих ролей мають різні рівні доступу та відповідні права, і при авторизації система визначає роль користувача і надає відповідні права. Це дозволяє обмежити доступ до окремих функціональних можливостей системи та забезпечити безпечну роботу із даними ресторану. Адміністратор має доступ до редагування меню, перегляду звітності, керування користувачами та

налаштування системи, тоді як персонал закладу працює лише із функціоналом створення та обробки замовлень.

Головне меню закріплене за AdminForm, тут знаходиться як швидкий доступ до перегляду замовлень, так і доступ до перегляду вебзамовлень, панель управління (рис. 3.3).



Рисунок 3.3 – Панель управління Averro POS

Панель управління складається з декількох функціональних блоків таких як додавання, перегляд та відкриття чи закриття зміни. Така структура дозволяє логічно розділити функціонал системи та забезпечити швидкий доступ до необхідних інструментів. Головне меню системи розроблене з урахуванням простоти використання та мінімізації часу на виконання основних операцій персоналом ресторану.

Блок додавання відповідає за створення і додавання елементів меню, знижок, модифікаторів та користувачів, і дозволяє швидко та легко перейти до відповідних форм та повернутись назад на головну сторінку. Даний функціонал забезпечує гнучке керування системою без необхідності внесення змін у програмний код, що дозволяє адаптувати систему під різні типи закладів ресторанного бізнесу.

За перегляд меню, замовлень та створення замовлення відповідає блок перегляду, тут наявний швидкий доступ до редагування меню сайту і перегляду замовлень на сайті, перегляду замовлень внутрішньої системи та переходу до створення самого замовлення. Крім цього, система дозволяє здійснювати

контроль активних замовлень, переглядати статуси чеків та працювати із поточними замовленнями в режимі реального часу.

Блок відкриття та закриття зміни має функцію відкриття або ж закриття доступу до замовлень, тобто відкриття зміни запускає систему в робочий стан та відкриває доступ до створення замовлення, а також запускає необхідний функціонал для доступу вебменю, переводить сайт в робочий стан і дозволяє також приймати замовлення від клієнтів. Закриття зміни, навпаки, завершує роботу системи для персоналу, обмежує створення нових замовлень та дозволяє сформувати підсумкову звітність за зміну.

Окремим важливим функціоналом є меню, яке знаходиться на MenuForm, тут відбуваються основні процеси роботи системи та реалізована основна логіка всіх замовлень. Саме дана форма є центральним елементом роботи POS-системи, оскільки забезпечує взаємодію персоналу із меню ресторану, кошиком замовлення, столами та механізмами формування чеків.

Особливістю цієї форми є наявність динамічного меню, створення замовлень, взаємодія з замовленнями, пошук по категоріях чи позиціях для зручного і швидкого користування, також форма має можливість переглянути вебзамовлення та додавати нові позиції, як до звичайного замовлення так і до вебзамовлень.

Нижче наведено фрагмент коду з відображенням динамічного меню POS-системи (ліст. 3.3).

Лістинг 3.3 – Завантаження вмісту динамічного меню

```
void LoadAllProducts()
{
    var table = MainDb.GetTable(@"
        SELECT
            mi.idmenu_items,
            mi.name,
            mi.price,
            mi.SubCategoryID,
            sc.CategoryID,
            c.Name AS CategoryName,
            sc.Name AS SubCategoryName
        FROM menu_items mi
        JOIN subcategories sc ON sc.SubCategoryID =
            mi.SubCategoryID
```

```

        JOIN categories c ON c.CategoryID = sc.CategoryID
        WHERE mi.is_available = 1
              AND sc.IsActive = 1
              AND c.IsActive = 1

        ORDER BY c.DisplayOrder, sc.DisplayOrder, mi.Name");
products = table.AsEnumerable()
    .Select(row => new Product
    {
        idmenu_items = row.Field<int>("idmenu_items"),
        Name = row.Field<string>("name"),
        Price = Convert.ToDecimal(row["price"]),
        CategoryID = Convert.ToInt32(row["CategoryID"]),
        SubCategoryID =
Convert.ToInt32(row["SubCategoryID"]),
        CategoryName = row["CategoryName"].ToString(),
        SubCategoryName = row["SubCategoryName"].ToString()

    })
    .ToList();
}

```

кінець лістингу 3.3

Даний метод бере дані із бази даних, де прив'язані дані до конкретних категорій та підкатегорій, чим дозволяє легко розміщати та керувати меню, як в десктоп-застосунку, так і на сайті. Метод динамічного меню набагато зручніший у використанні ніж статичний, особливо коли функціонал призначений на великий об'єм роботи, за допомогою такого функціоналу, можна легко створити зручний інтерфейс який легко можна редагувати без втручання в код та сам інтерфейс десктоп-застосунку без особливих навичок в програмуванні, так як при створенні позицій, є додаткова можливість як тимчасово прибрати непотрібні позиції так і видалити їх. Так само можна взаємодіяти з категоріями та підкатегоріями, і розміщати в зручному порядку для використання. Також нижче наведено фрагмент коду, який дозволяє пошук в меню (ліст 3.4).

Лістинг 3.4 – Метод пошуку в меню

```

private void ApplySearch()
{
    string searchText = textBoxSearch.Text.Trim();
    OrderPanel.Controls.Clear();

    try
    {
        DataTable dt = mainDb.GetTable(@"
            SELECT mi.idmenu_items, mi.name, mi.price,
                  sc.CategoryID, mi.SubCategoryID,

```

```

        c.Name AS CategoryName,
        sc.Name AS SubcategoryName
    FROM menu_items mi
    INNER JOIN subcategories sc ON sc.SubCategoryID =
mi.SubCategoryID
    INNER JOIN categories c ON c.CategoryID =
sc.CategoryID
    WHERE mi.is_available = 1
        AND sc.IsActive = 1
        AND c.IsActive = 1
        AND (@Search = '' OR mi.name LIKE CONCAT('%',
@Search, '%'))
    ORDER BY c.DisplayOrder, sc.DisplayOrder, mi.name",
    ("@Search", searchText));

    foreach (DataRow row in dt.Rows)
    {
        Product product = new Product
        {
            idmenu_items =
Convert.ToInt32(row["idmenu_items"]),
            Name = row["name"].ToString(),
            Price = Convert.ToDecimal(row["price"])
        };

        Button btn = CreateProductButton(product);
        OrderPanel.Controls.Add(btn);
    }

    if (dt.Rows.Count == 0)
        ShowEmptySearchResult();
}
catch (Exception ex)
{
    MessageBox.Show("Помилка пошуку: " + ex.Message);
}
}

```

кінець лістингу 3.4

У лістингу 3.4 наведено фрагмент методу пошуку позицій меню в десктоп-застосунку POS-системи. Метод отримує введений користувачем текст, виконує SQL-запит до бази даних із фільтрацією доступних позицій меню та динамічно формує кнопки для відображення знайдених страв. Якщо результат пошуку порожній, система виводить відповідне повідомлення для користувача. Такий підхід дозволяє швидко знаходити необхідні позиції меню та спрощує процес формування замовлення.

Також можна взаємодіяти із функціоналом позицій меню, коли при натисканні в панелі меню на будь-яку позицію, її можна додати в кошик, де можна взаємодіяти з таким функціоналом як видалити, редагувати, замовити,

перенести на інше замовлення, збільшити чи зменшити кількість. Основний функціонал MenuForm зображений на (рис 3.4).

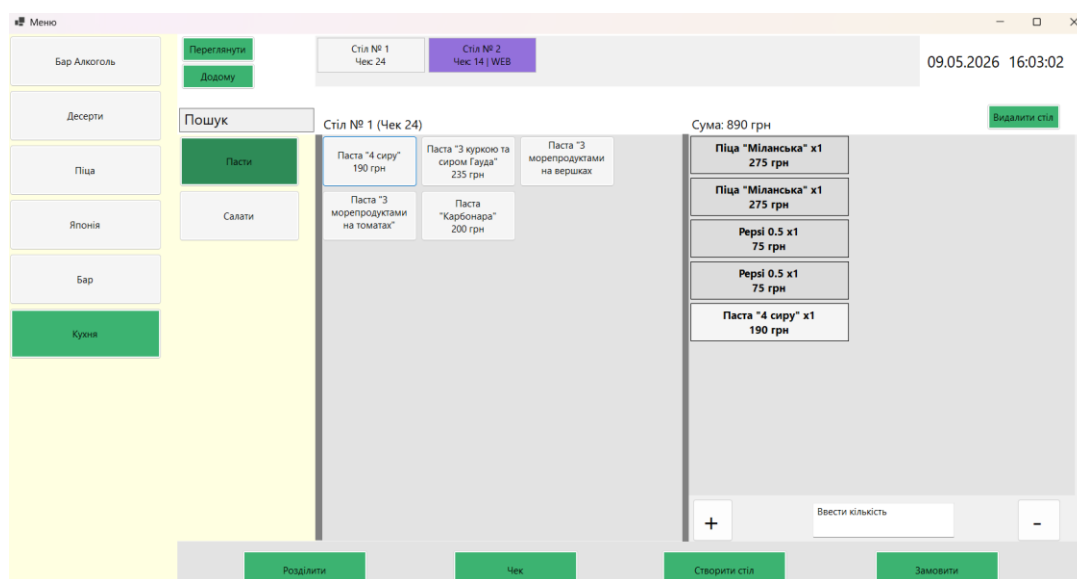


Рисунок 3.4 – Основний функціонал MenuForm

Саме тут реалізовано основну частину функціоналу системи, яка дозволяє створити замовлення, вибрати стіл, додати до поточного замовлення позиції меню, оплату, розділення замовлення, перегляд та дозамовлення у веб-замовлення, також нижче наведено фрагмент коду (ліст. 3.5), який демонструє створення нового чека.

Лістинг 3.5 – Створення нового чека

```
private void CreateNewCheck()
{
    CurrentOrderNumber = MainDb.GetScalar<int>(
        "SELECT IFNULL(MAX(OrderNumber), 0) + 1 FROM checks");

    CurrentTableOrderNumber = MainDb.GetScalar<int>(
        "SELECT IFNULL(MAX(TableOrderNumber), 0) + 1 FROM checks
WHERE TableID=@tid",
        ("@tid", CurrentTable));

    string sql = @"
        INSERT INTO checks
        (TableID, TableOrderNumber, OrderNumber, Status,
CreatedAt,
        CreatedByUserId, CreatedByUsername, CreatedByUserRole)
        VALUES
```

```

        (@TableID, @TableOrderNumber, @OrderNumber, 'Pending',
        @CreatedAt,
        @CreatedByUserId, @CreatedByUsername,
        @CreatedByUserRole)";

        MainDb.Execute(sql,
            ("@TableID", CurrentTable),
            ("@TableOrderNumber", CurrentTableOrderNumber),
            ("@OrderNumber", CurrentOrderNumber),
            ("@CreatedAt", DateTime.Now),
            ("@CreatedByUserId", CurrentUserId),
            ("@CreatedByUsername", CurrentUsername),
            ("@CreatedByUserRole", CurrentUserRole));

        CurrentCheckId = MainDb.GetScalar<int>("SELECT
        LAST_INSERT_ID()");

        MainDb.Execute(
            "UPDATE tables SET Status = 'Occupied' WHERE TableID =
        @TableID",
            ("@TableID", CurrentTable));

        SelectedCheck = new CheckKey(CurrentTable,
        CurrentTableOrderNumber, CurrentOrderNumber);
    }

```

кінець лістингу 3.5

Форма також має окремі блоки такі як: блок позицій меню, блок вибору замовлень, блок перегляду позицій, кошик, та панель керування. Дане рішення розроблено з урахуванням принципів простоти та зручності використання функціоналу системи. Завдяки цьому можна легко створити замовлення, знайти необхідні страви через пошук або панель меню яка знаходиться ліворуч, і обрати для замовлення, в кошику також присутня взаємодія з позиціями меню, має можливість як додати модифікатор до страви і видалити інгредієнт з страви, так видалити саму страву з замовлення, якщо вона ще не була замовлена, також можна легко обрати кількість страви чи напою, можна як зробити це через введення кількості, так і через додавання чи віднімання по одній позиції, це зручно для замовлень де потрібно вводити конкретну кількість страв чи напоїв.

Форма також має можливість обрати стіл і виводить дані по замовленню, такі як: сума, знижка, номер столу та чеку. Присутній швидкий доступ до перегляду всіх замовлень, де можна переглянути не лише замовлення, а й зробити звітність як по днях, так і за період, так і по елементах системи (позиції меню, категорії, офіціанти, тип оплати тощо) (рис. 3.5).

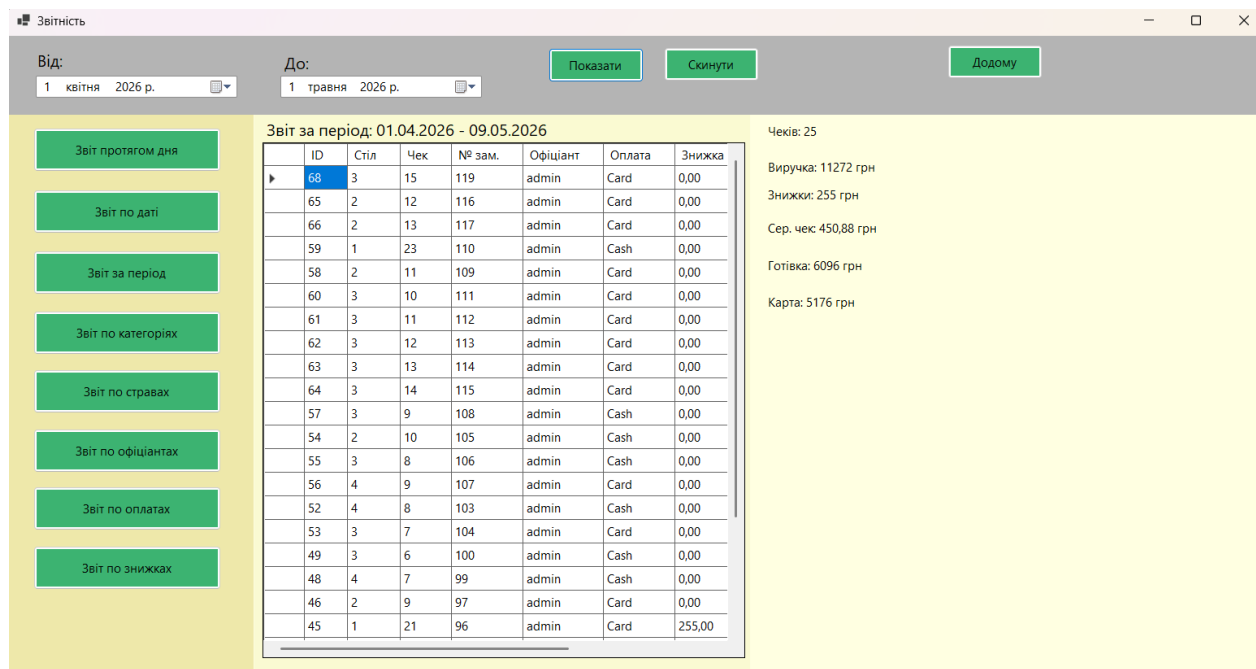


Рисунок 3.5 – Перегляд звітності

Система десктоп-застосунку має обширний обсяг функціоналу, де мають можливість редагувати меню та виводити потрібні дані на сайт, без втручання в код сайту, так і створення та змінення столів, розрахунок замовлень, закриття чеків через тип оплати, розділення, або переміщення замовлень по різних столиках. Таким чином система має потрібний функціонал для існування всієї системи. Інтуїтивно зрозумілий дизайн і логічну послідовність.

Вебінтерфейс меню розроблений із використанням HTML, CSS та JavaScript. Основним призначенням є надання клієнтам можливість переглядати меню закладу, формувати замовлення та здійснювати оплату безпосередньо із смартфона.

Після переходу за QR-кодом клієнт потрапляє на сторінку меню, де автоматично система визначає стіл, до якого прив'язане замовлення.

Функціональні можливості вебінтерфейсу:

- перегляд категорій та підкатегорій меню;
- додавання позицій до кошика;
- редагування кількості товарів;

- надсилення замовлення на сервер;
- перегляд поточного замовлення;
- перехід до сторінки оплати.

Вебінтерфейс адаптований для мобільних пристроїв та коректно відображається на різних розмірах екранів.

Розглядаючи систему зі сторони клієнта, вебменю має досить зрозумілий і стильний формат (рис. 3.6).

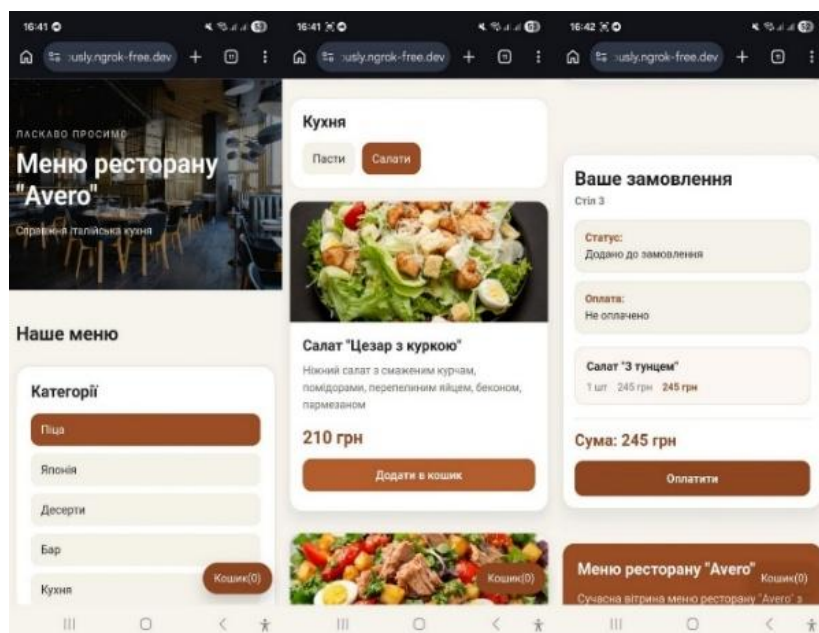


Рисунок 3.6 – Вигляд вебменю клієнтської частини

Дане зображення містить об'єднанні скріншоти сайту де можна переглянути вигляд сайту та зміст. Під час розробки такого інтерфейсу та функціоналу було застосовані методи динамічного виводу даних із бази даних через запити API та шифрування. Суть меню полягає не в статичному розширенні, а в динамічному, що значно робить легким роботу для власника системи, і саме тому користувач системи який працює із функціоналом має можливість редагувати меню, опис страв, додавання чи змінення фотографій, зміну ціни, без особливих знань в програмуванні, які навіть не потрібні для користувача, тому що функціонал має автоматизацію системи завдяки чому меню динамічно виводиться на сайті, код наведено в додатку Б.

Забезпечує таку реалізацію три компоненти, десктоп-частина яка лише редагує і вносить зміни в налаштування, там відбувається процес надання потрібних даних, база даних де зберігаються дані нашої системи, і сайт, сама головна частина вебменю, сайт робить запит через API до бази даних і успішно повертає вміст, але цього недостатньо для виведення динамічного меню, такий метод працює із статистичним меню, щоб повністю забезпечити динамічне меню потрібне автоматизований вивід даних, і це завдання виконує в основному JavaScript, де прописано великий обсяг функціоналу який виводить автоматично потрібні дані в правильному для користувача порядку, нижче представлено фрагмент коду (ліст. 3.6), який демонструє створення рендеру вмісту меню.

Лістинг 3.6 – Створення картки вмісту

```
function renderItem() {
  const grid = document.getElementById("menuGrid");
  const items = getFilteredItems();

  if (!items.length) {
    grid.innerHTML = `<div class="empty-state">У цій
    підкатегорії ще немає позицій меню.</div>`;
    return;
  }

  grid.innerHTML = items.map(item => `
    <article class="menu-card">
      ${createImageBlock(item)}
      <div class="menu-body">
        <div>
          <h4 class="menu-name">${escapeHtml(item.name ||
"" )}</h4>
          <p class="menu-
description">${escapeHtml(item.description || "" )}</p>
          </div>
          <div class="menu-footer">
            <div class="menu-
price">${formatPrice(item.price)} грн</div>
            <div class="menu-
onclick="addToCart(${item.id})">
              <button class="add-to-cart-btn"
                Додати в кошик
              </button>
            </div>
          </article>
        `).join("");
  }
}
```

кінець лістингу 3.6

Основний принцип сайту налаштований саме на рендері потрібних позицій для виведення інформації та її прийнятті.

Також система має можливість приймати замовлення, так як кожен QR-код прив'язаний до конкретного стола і має власний унікальний токен який зберігається в базі даних, це спрощує визначення столу при обробці замовлення, і коли клієнт додає в кошик замовлення і відправляє його в обробку, система визначає за токеном конкретний стіл, таким чином не потрібно шукати чи призначати стіл замовлення вручну. Аналогічно при виведенні замовлення, якщо воно створене не через сайт.

Клієнт також може оплачувати замовлення через сайт, що є досить зручним функціоналом як для персоналу так і гостей, процес оплати відбувається через шифрування та API, є декілька різних способів створення оплати.

Перший із яких налаштування оплати через підключення сертифікованого провайдера таких як LiqPay, Altegio і т.д [2]. Цей метод полягає у договорі між закладом та провайдером, власник системи і закладу в цілому реєструє мережу закладу та подає запит на прийняття оплати через Google Pay, Apple Pay, сервіс надає власні API та код для переходу на сторінку провайдера при оплаті, тим самим бере відповідальність за збереження особистих даних клієнтів та безпеку оплати на своїй стороні, це є досить зручним і простим методом, так як всю важливу роль в оплаті грає провайдер, і за відповідним договором власник оплачує даний функціонал системи і взамін отримує просте і готове рішення для системи.

Другий метод полягає в підключенні оплати через власне API та створення власного функціоналу для оплати через сайт, де демонструється метод успішної оплати, невдалої оплати, формування електронного чеку, даний метод використовується в POS-системі для більш масштабнішого розуміння роботи з замовленнями, оплатою, та безпекою даних. Приклад формування тестової оплати зображений на рисунку 3.7.

Тестова оплата

Перевірка логіки оплати перед підключенням платіжного провайдера

Order reference: CHK-77-1780262424

Сума: 550,0000 грн

Статус: Paid

Імітувати успішну оплату

Імітувати невдалу оплату

Повернутися до меню

Переглянути чек

Статус оновлено: Paid

Рисунок 3.7 – Перегляд тестової оплати

Цей метод містить декілька проблем, такі як безпека даних, отримання дозволів, перевірка системи на захищеність від атак, і т.д. Щодо безпеки даних то тут доволі просто, тому що відповідальність за збереження інформації клієнтів і розрахунок лежить на стороні власника, якщо в першому випадку цю інформацію зберігає провайдер, і відповідальність лежить на стороні провайдера, то в даному випадку відповідальність зростає в рази. Отримання дозволів та сертифікатів, це потрібно пройти перевірку і отримати сертифікат на власний метод оплати в Google Pay чи Apple Pay для їх використання, і якщо не пройти перевірку на вимоги цих компаній, не буде надано доступ до користування таким функціоналом. Щодо перевірки системи на безпеку, система потребує доволі сильний захист від кібератак, які можуть отримати дані клієнтів, нижче наведено код (ліст. 3.7), який демонструє метод проведення оплати.

Лістинг 3.7 – Метод проведення оплати

```

async function completeTestPayment(status) {
  if(!orderReference){
    showStatus("Order reference не знайдено.");
    return;
  }

```

```

    try{
        const response = await
        fetch(`${window.location.origin}/api/payment/test-complete`,{
            method : "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({
                orderReference: orderReference,
                status: status,
                paymentMethod : "GooglePay"
            })
        });

        const text = await response.text();

        if(!response.ok){
            throw new Error(text || "Не вдалося оновити тестову
оплату");
        }

        const data = JSON.parse(text);

        paymentStatusText.textContent = data.status;
        showStatus(`Статус оновлено: ${data.status}`);

        if(data.status === "Paid" && data.checkId){
            paidCheckId = data.checkId;
            receiptBtn?.classList.remove("hidden");
        }

    }catch(error){
        console.error(error);
        showStatus("Помилка: "+ error.message);
    }
}

```

кінець лістингу 3.7

Тож дана система має можливість працювати з двома методами, і перевага в першому методі набагато більша, але було реалізовано тестову оплату, для розуміння роботи системи оплати, потрібні ключі, шифрування, та методів оплати, тому при установці системи, надається безкоштовна послуга на установку сертифікованого провайдера, і можливість отримати доступ до оплати через сайт, також було ознайомлено з вимогами провайдерів, і вивчено потрібний матеріал для отримання послуг провайдерів.

Для спрощення доступу клієнтів до електронного меню реалізовано систему QR-кодів [21].

Кожному столу відповідає власний унікальний токен, який зберігається в таблиці «table_qr». На основі цього токена формується посилання на вебменю.

Після сканування QR-коду вебінтерфейс надсилає запит до сервера для визначення номера столу та активного чека [5]. Після чого повертає необхідні дані такі як номер столу та активний чек, а також вміст замовлення, якщо воно було присутнє, що дозволяє швидко оплатити чи дозамовити до рахунку.

Такий підхід дозволяє автоматично прив'язувати вебзамовлення до конкретного столу та поточного чека.

Створення QR-кодів можна реалізувати за допомогою бібліотеки Visual Studio «QrCoder» достатньо інсталиувати її через NuGet Пакет, а також невелика кількість коду (ліст. 3.8), який буде формувати зображення завдяки бібліотеці, важливим є, щоб посилання та токени мали правильну адресу, тоді генерація QR-коду буде успішною.

Лістинг 3.8 – Генерація QR-Коду

```
private void GenerateAndShowQrImage(string url)
{
    if(string.IsNullOrEmpty(url))
    {
        pictureBoxQr.Image?.Dispose();
        pictureBoxQr.Image = null;
        return;
    }

    using QRCodeGenerator qrGenerator = new QRCodeGenerator();
    using QRCodeData qrCodeData = qrGenerator.CreateQrCode(url,
        QRCodeGenerator.ECCLLevel.Q);
    using QRCode qrCode = new QRCode(qrCodeData);
    using Bitmap qrBitmap = qrCode.GetGraphic(6);

    pictureBoxQr.Image?.Dispose();
    pictureBoxQr.Image = new Bitmap(qrBitmap);
}
```

кінець лістингу 3.8

Функція вибирає посилання створене раніше до конкретного стола з прив'язаним токеном, і відштовхуючись від цього генерує QR-код, який вже дозволяє перейти за посиланням, також можна налаштувати посилання і метод

створення посилання, тим самим посилання може бути тимчасовим так і постійним, нижче наведено вигляд форми з редагування даних столів де застосовуються методи формування токенів та QR-кодів (рис. 3.8).

Рисунок 3.8 – Форма редагування даних столу

Для генератора токена можна використовувати метод (ліст. 3.9), який створює загальний токен.

Лістинг 3.9 – Метод створення загального токена

```
private string GenerateQrToken(int length = 10)
{
    const string chars =
    "ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz23456789";
    var random = new Random();
    return new string(Enumerable.Repeat(chars, length)
        .Select(s => s[random.Next(s.Length)]).ToArray());
}
```

кінець лістингу 3.9

Також для створення унікального токена використовується метод (ліст. 3.10), який дозволяє згенерувати унікальний токен і записати його в базу даних, звідки можна використовувати його, як в посиланнях, так і генерації QR-кодів.

Лістинг 3.10 – Метод для створення унікального токена

```
private string GenerateUniqueQrToken()
```

```

{
    string token;
    int exists;

    do
    {
        token = GenerateQrToken(12);

        exists = MainDb.GetScalar<int>(@"
            SELECT COUNT(*)
            FROM table_qr
            WHERE QrToken = @QrToken
            ", ("@QrToken", token));
    }
    while (exists > 0);
    return token;
}

```

кінець лістингу 3.10

Даний метод використовує попередній метод, де генеруються загальні токени, і після цього при виборі столу та виконанні даного методу, токен прив'язується до столу та реєструється в базі даних під конкретним столом, після чого його можна буде використати в створенні посилань, та генерації QR-кодів, методом вказаним вище (ліст. 3.8).

Отже враховуючи, важливість системи QR-кодів. Наявність такого функціоналу значно спрощує керування та користування системою, що дозволяє зменшити навантаження на персонал, так і ефективно використовувати відвідувачам які можуть швидко замовити, оплатити чи переглянути замовлення.

3.2 Проєктування та реалізація бази даних

Для зберігання даних у системі використовується MySQL. Також для зручності роботи з базою даних робота була проведена в сервісі MySQL Workbench.

Для бази даних використовується дві поточні бази «MainDb» та «AuthDb». Розділення вимог та вмісту цих баз даних пов'язане з безпекою даних, хоча сайт та система працюють з базами даних в основному через API, але розділення

потоків інформації і доступу до неї дозволяє посилити захист, і легко орієнтуватись в базах даних при розробці.

Головна база даних «MainDb» містить таблиці як пов'язані з меню та замовленнями:

- Menu_items;
- Categories;
- Subcategories;
- Tables;
- Orders (рис. 3.9);
- Checks
- Payments;
- Web_orders;
- Table_qr.

Головна база даних містить в собі дані про замовлення такі як, назва, кількість, ціна, хто створив замовлення і т.д.

OrderID	TableID	ProductID	ProductName	Category	Quantity	Modifiers	OrderTime	PriceAtMoment	Status	CreatedAt	CheckID	Cr
210	3	46	Салат "З тунцем"	Кухня	1.00		2026-05-09 16:41:50	245.00	Completed	2026-05-09 16:41:51	70	1
211	1	36	Піца "Міланська"	Піца	1.00		2026-05-10 17:23:28	275.00	Paid	2026-05-10 17:23:29	71	1
212	3	48	Паста "З норепо..."	Кухня	2.00		2026-05-20 22:21:20	255.00	Paid	2026-05-20 22:21:21	72	1
213	1	36	Піца "Міланська"	Піца	1.00		2026-05-20 22:25:49	275.00	Paid	2026-05-20 22:25:49	73	1
214	3	35	Салат "Цезар з ку..."	Кухня	1.00		2026-05-20 22:33:34	210.00	Paid	2026-05-20 22:33:34	74	1
215	3	43	Чокелік "Солона к..."	Десерти	2.00		2026-05-29 10:05:07	140.00	Open	2026-05-29 10:05:08	75	1
216	1	36	Піца "Міланська"	Піца	1.00		2026-05-29 10:06:04	275.00	Open	2026-05-29 10:06:04	76	1
217	2	36	Піца "Міланська"	Піца	1.00		2026-05-31 23:59:00	275.00	Paid	2026-05-31 23:59:00	77	1
218	2	36	Піца "Міланська"	Піца	1.00		2026-05-31 23:59:00	275.00	Paid	2026-05-31 23:59:00	77	1

Рисунок 3.9 – Дані з таблиці Orders

Також можна взаємодіяти із базою даних напряму в POS-системі створюючи нові позиції для меню, змінюючи вихідні дані на сайт, створюючи замовлення, додавати нових користувачів, модифікатори, і також бачити в форматі таблиці та функціоналу, продемонстровано взаємодію із таблицею Menu_items на рисунку 3.10.

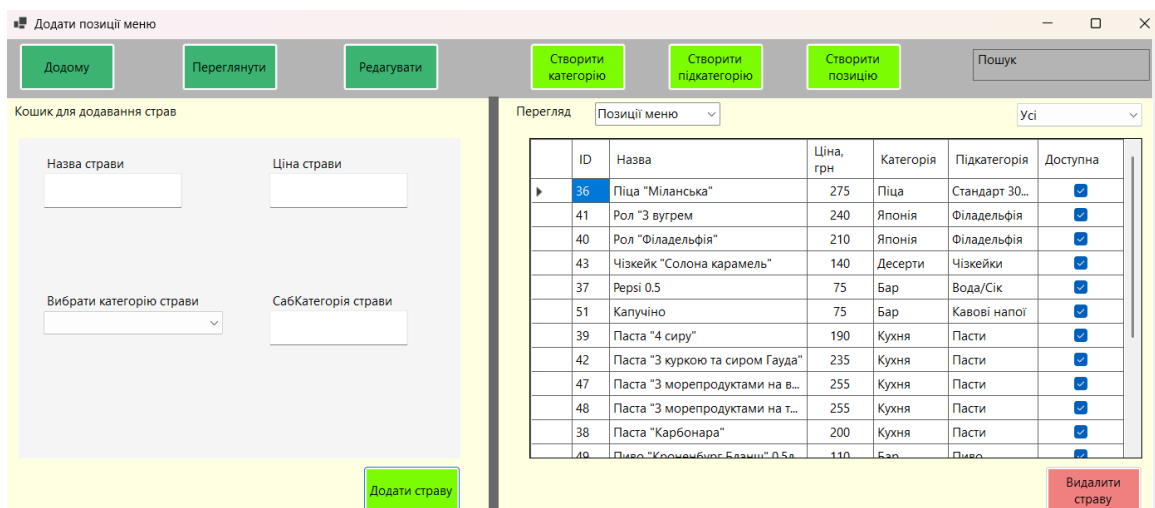


Рисунок 3.10 – Демонстрація взаємодії з таблицею Menu_items

База даних «AuthDb» містить інформацію про користувачів системи, їх ролі та дані, необхідні для аутентифікації й авторизації. База даних спроектована таким чином, що доступ до бази даних виконується з урахуванням реляційних зв'язків та забезпечує цілісність даних. Тому вибір розділити бази даних на ролі та обов'язки було необхідним рішенням, адже в разі витоку даних і отримання доступу до головної бази даних, не буде чутливим, адже інформація яка міститься там, лише містить замовлення і меню, та час оплати замовлень, тоді як оплата і платежі в ніякому випадку не взаємодіють з головною базою даних, а інша база даних має доступ до платежів таким чином, що вона не фігурує напряду із сайтом, і так само через API, тим самим ускладнюючи завдання для злоумисників, і при захисті для підвищення рівня захисту доступ до даних здійснюється через серверну частину системи та механізми авторизації користувачів та містить лише дату оплати, суму, і кінець номера карти для ідентифікації замовлень, саме тому безпека даних реалізована на достатньо високому рівні.

Для розділення і підключення різних баз даних було створено такі файли як: «DatabaseHelper.cs» та «DB.cs». В файлі «DatabaseHelper.cs» містяться додаткові функції та підключення до баз даних необхідних для взаємодії системи з базою даних, наведених в лістингу 3.11.

Лістинг 3.11 – Підключення до бази даних «AuthDb»

```

public static bool CanConnectToAuthDb()
{
    try
    {
        using (var connection = new
        MySqlConnection(AuthConnectionString))
        {
            connection.Open();
            return true;
        }
    }
    catch
    {
        return false;
    }
}

```

кінець лістингу 3.11

Дана функція використовує підключення до бази даних для аутентифікації користувачів, але для повноцінного функціонування файлу потрібно мати «Dbconfig.cs» в якому міститься підключення до порту, серверу, містить також логін та пароль, і записує це в файл, в лістингу 3.12 наведено приклад головного підключення.

Лістинг 3.12 – Головне підключення до бази даних

```

public string GetMainConnectionString()
{
    return
    $"server={Server};port={Port};database={MainDatabase};uid={User};pwd={Password};charset=utf8mb4;";
}

public static DbConfig Load()
{
    try
    {
        if (!File.Exists(ConfigPath))
        {
            return new DbConfig();
        }

        string json = File.ReadAllText(ConfigPath);
        return JsonConvert.DeserializeObject<DbConfig>(json) ?? new DbConfig();
    }
}

```

```
    }  
    catch  
    {  
        return new DbConfig();  
    }  
}
```

кінець лістингу 3.12

Таким чином, у процесі розробки було спроектовано та реалізовано структуру бази даних POS-системи ресторану «Авего», яка забезпечує надійне зберігання та обробку інформації про меню, замовлення, чеки, платежі та користувачів системи. Використання двох окремих баз даних дозволило розмежувати функціональні обов'язки та підвищити рівень безпеки інформації. Реалізовані механізми підключення до баз даних забезпечують стабільну взаємодію між серверною частиною, вебінтерфейсом та десктоп-застосунком, що є необхідною умовою для коректної роботи всієї POS-системи.

3.3 Тестування та налагодження інформаційно-комп'ютерної системи

У процесі розробки POS-системи було проведено комплексне тестування функціональності всіх компонентів системи.

Було перевірено:

- створення та редагування меню;
- формування замовлень (рис. 3.11);
- створення та закриття чеків;
- роботу QR- кодів;
- формування онлайн-замовлень (рис. 3.12);
- обробку платежів;
- генерацію звітності;
- роботу інсталяційного пакета.

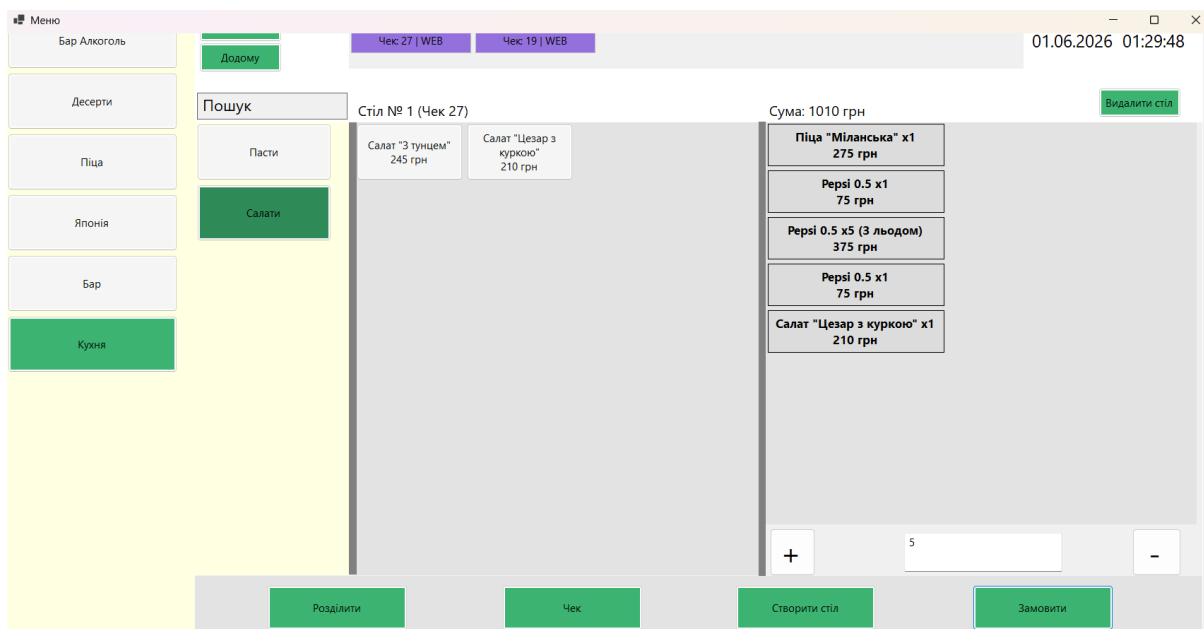


Рисунок 3.11 – Тестування формувань замовлень в MenuForm

Для перевірки коректності роботи механізму створення замовлень було проведено тестування додавання позицій меню до кошика, зміни кількості товарів, додавання модифікаторів та формування нового замовлення. У результаті тестування встановлено, що система коректно зберігає інформацію про замовлення, автоматично виконує розрахунок вартості та формує запис у базі даних.

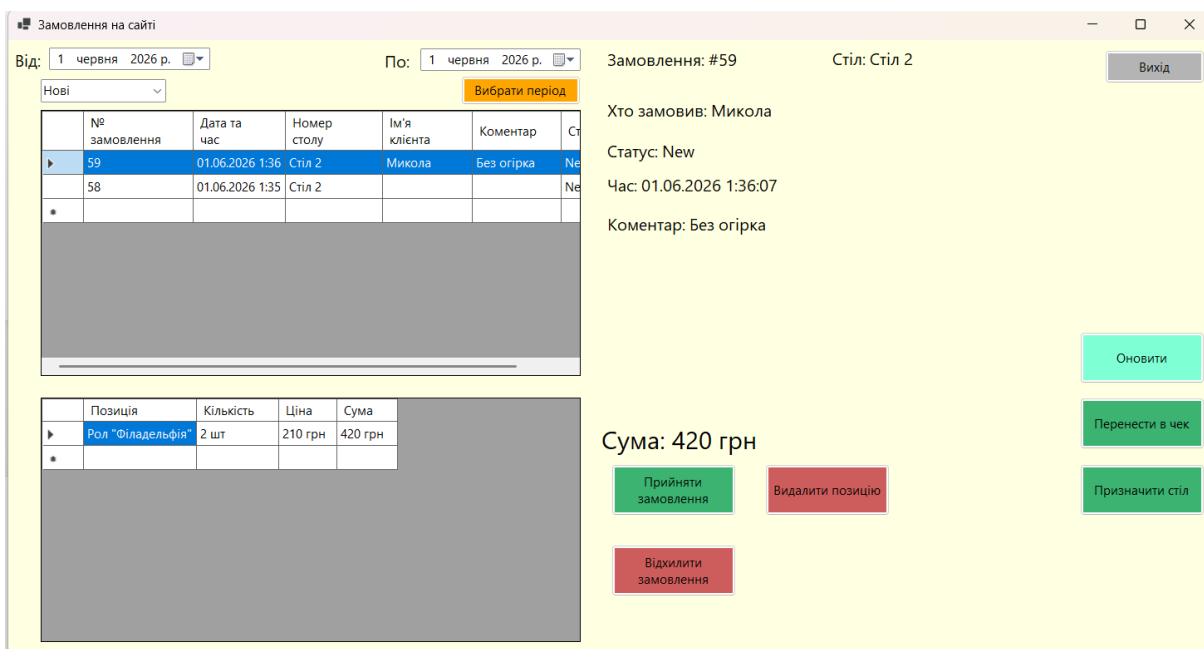


Рисунок 3.12 – Перевірка формування замовлень на сайті

Окремо було проведено тестування вебінтерфейсу та механізму онлайн-замовлень [16]. Перевірено передачу даних між вебінтерфейсом та серверною частиною системи через REST API. Усі створені замовлення успішно надходили до POS-системи та відображались у списку вебзамовлень без втрати даних.

У процесі тестування було виявленні та усунені помилки пов'язані з обробкою порожніх значень, SQL-запитами, зв'язками між таблицями та оновленням інтерфейсу, також наведено результати тестування основних компонентів в таблиці 3.1.

Таблиця 3.1 – Результати тестування основного функціоналу

Функціонал	Очікуваний результат	Результат
Створення замовлення	Замовлення додається до БД	Успішно
Закриття чека	Статус змінюється на Completed	Успішно
Вебзамовлення	Дані надходять у POS	Успішно
QR-код	Відкривається потрібне меню, та актуальні дані столу	Успішно
Формування звіту	Відображаються актуальні дані	Успішно

Під час розробки проєкту, тестування проходило як і під час розробки так і було проведено загальне кінцеве тестування кожного компонента та модулів. Під час розробки тестування спрощувало роботу в майбутньому через те, що можна було правильно та чітко виконати роботу над важливими елементами, одним із важливих елементів які тестувались і часто видавали помилки це була логіка замовлень, тому, що там присутні значення, як і столів, замовлень, так і нумерація замовлень по категоріях, інколи проблема полягала в неправильному рахунку, що дуже важливим було протестувати цей функціонал, через його велику значимість, код готового результату наведено в додатку В.

В результаті було знайдено правильне рішення і функціонал поширився також на вебінтерфейс. Перенесення такого досвіду на вебінтерфейс дозволило реалізувати повноцінний механізм замовлень та обмін даними між вебзастосунком та десктоп-застосунком. Було вирішено проблему з правильним запуском серверу, що дозволило виводити локальні дані такі як зображення на сайт, без особливих проблем (рис. 3.13).

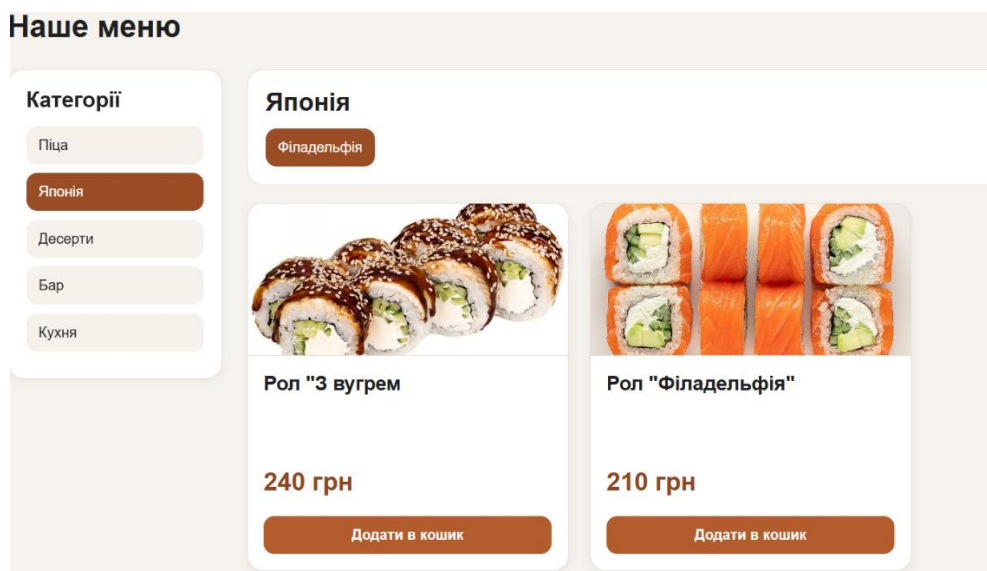


Рисунок 3.13 – Вивід зображень на сайті

Завдяки такому функціоналу, вдалось правильно виводити зображення, а також було протестовано як правильний вивід зображень, так і зміну зображень в реальному часі, що дозволило динамічно змінювати меню, без коду.

Загалом було проведено тестування важливих компонентів системи таких як десктоп-застосунок, вебзастосунок, серверну частину, базу даних, було визначено недоліки і виправлено їх, також було приділено увагу до проблем з отриманням та виведенням даних, після тестування та виправлень, система наближалась до завершення створення повноцінної системи.

3.4 Розробка інсталяційного пакета

Для спрощення процесу встановлення POS-системи було створено інсталяційний пакет за допомогою сервісу Inno Setup [7].

Інсталяційний пакет автоматично:

- копіює виконувані файли програми;
- додає необхідні бібліотеки;
- копіює конфігураційні файли;
- створює ярлики на робочому столі та в меню Пуск;

- запускає програму після завершення встановлення.

Основні секції скрипта Inno Setup:

- [Setup];
- [Files];
- [Icons];
- [Run].

Під час запуску інсталятора користувач обирає каталог встановлення, після чого система автоматично копіює необхідні файли та формує структуру директорій, нижче наведено фрагмент коду скрипту файлу (ліст. 3.13).

Лістинг 3.13 – Фрагмент скрипту коду інсталяційного пакету

```
[Files]
; WinForms POS
Source: "POS\*"; DestDir: "{app}\POS"; Flags: ignoreversion
recursesubdirs createallsubdirs

; ASP.NET Core API (разом з wwwroot)
Source: "API\*"; DestDir: "{app}\API"; Flags: ignoreversion
recursesubdirs createallsubdirs

; SQL-дамп бази даних
Source: "Database\cafesym_db.sql"; DestDir: "{app}\Database";
Flags: ignoreversion
Source: "Database\auth_db.sql"; DestDir: "{app}\Database"; Flags:
ignoreversion

[Dirs]
Name: "{app}\Logs"

[Icons]
; Меню Пуск
Name: "{group}\Avero POS"; Filename: "{app}\POS\CafeSystem.exe"
Name: "{group}\Відкрити папку програми"; Filename: "{app}"

; Робочий стіл
Name: "{autodesktop}\Avero POS"; Filename:
"{app}\POS\CafeSystem.exe"; Tasks: desktopicon

[Run]
; Запуск програми після встановлення
Filename: "{app}\POS\CafeSystem.exe"; \
Description: "Запустити Avero POS"; \
Flags: nowait postinstall skipifsilent

[UninstallDelete]
Type: filesandordirs; Name: "{app}"
```

кінець лістингу 3.13

До складу інсталяційного пакета входять виконувані файли десктоп-застосунку, конфігураційні файли, бібліотеки, а також допоміжні ресурси.

У результаті компіляції .iss-скрипта формується виконуваний файл «Avero Setup.exe», який дозволяє швидко та зручно встановити POS-систему на комп'ютер користувача (рис. 3.14).

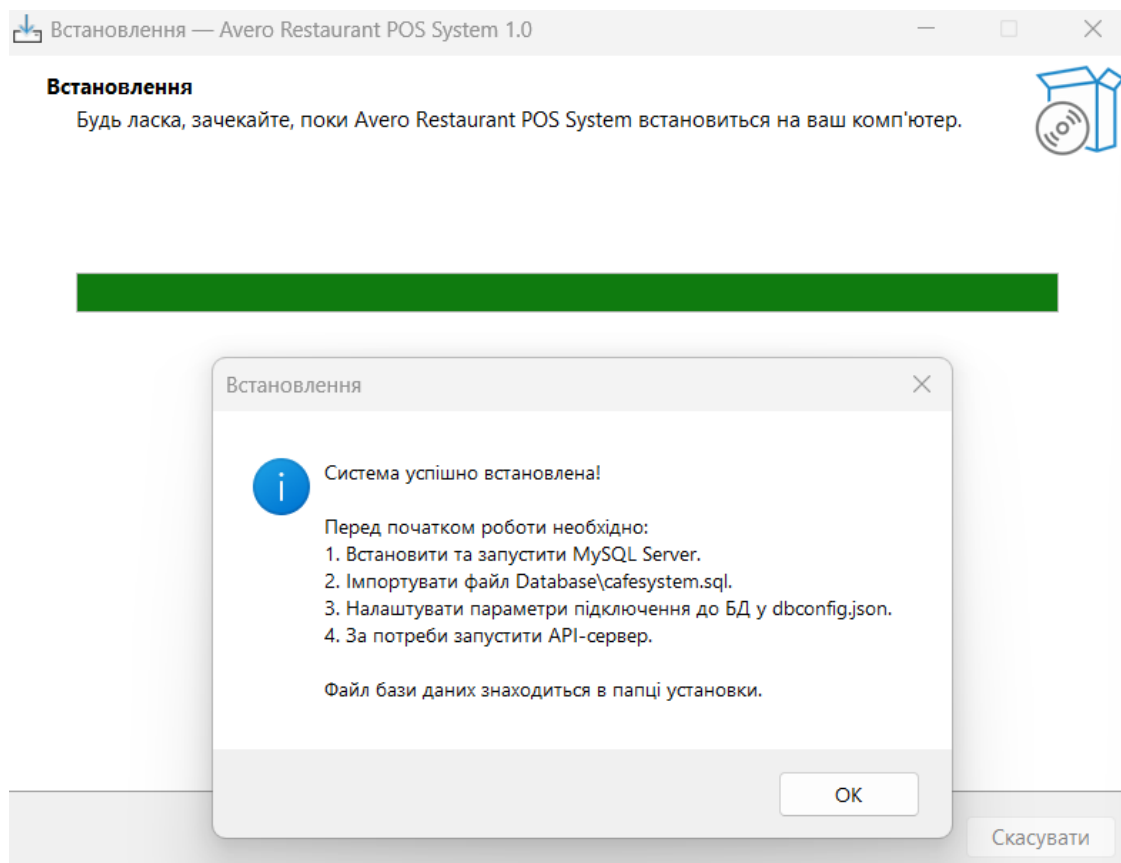


Рисунок 3.14 – Встановлення системи

Після завершення інсталяції система готова до запуску та подальшого налаштування параметрів підключення до сервера та бази даних. Особливістю розробленого інсталяційного пакета є підтримка двох режимів встановлення:

Головне встановлення – використовується для налаштування основного комп'ютера, на якому розміщується серверна частина системи, база даних та інші ключові компоненти;

Клієнтське встановлення – використовується для підключення додаткових робочих місць до головного комп'ютера за допомогою вказаної мережевої адреси.

Такий підхід дозволяє легко масштабувати систему та використовувати її як на одному комп'ютері. Так і в локальній мережі з декількома робочими місцями.

Таким чином, створення інсталяційного пакета дозволило значно спростити процес розгортання програмного забезпечення та забезпечити зручність його встановлення на робочих місцях персоналу ресторану.

ВИСНОВКИ

У межах даної кваліфікаційної роботи бакалавра було успішно реалізовано повний цикл розробки клієнт-серверної POS-системи для ресторану «Avero» з використанням технологій C#, ASP.NET Core, HTML, CSS та JavaScript. Виконана робота охопила всі основні етапи створення програмного забезпечення: від аналізу предметної області та формування вимог до системи до практичної реалізації, тестування та підготовки інсталяційного пакета.

У ході виконання роботи було проведено аналіз сучасних POS-систем, що використовуються у сфері ресторанного бізнесу, зокрема Toast POS, Poster POS та Restera. Проведений аналіз показав, що дані рішення забезпечують широкий функціонал для автоматизації діяльності закладів громадського харчування, включаючи керування меню, замовленнями, персоналом та платежами. Разом із цим було виявлено ряд недоліків, серед яких висока вартість використання окремих сервісів, залежність від сторонніх платформ, складність налаштування окремих модулів та наявність надлишкового функціоналу для закладів малого та середнього бізнесу. Отримані результати підтвердили актуальність розробки власної POS-системи, орієнтованої на потреби ресторанів, кафе та інших закладів громадського харчування.

На основі проведеного аналізу було сформовано функціональні та нефункціональні вимоги до POS-системи, а також визначено клієнт-серверну архітектуру програмного забезпечення. До функціональних вимог віднесено можливість керування меню, створення та обробки замовлень, формування чеків, облік оплат, ведення звітності, роботу із вебзамовленнями та підтримку системи QR-кодів для швидкого доступу клієнтів до меню ресторану. Нефункціональні вимоги передбачали забезпечення стабільності роботи системи, безпеки даних, зручності користування, масштабованості та можливості подальшого розширення функціоналу.

Було обґрунтовано вибір технологій та засобів розробки. Для реалізації десктоп-застосунку використано мову програмування C# та технологію

WinForms, для серверної частини ASP.NET Core Web API, для вебінтерфейсу HTML, CSS та JavaScript, а для збереження даних використано систему керування базами даних MySQL. Для створення інсталяційного пакета застосовано Inno Setup.

У процесі практичної реалізації розроблено повноцінну POS-систему, яка забезпечує керування меню, столами, замовленнями, чеками, платежами та звітністю. Також реалізовано систему QR-кодів, вебменю з підтримкою онлайн-замовлень та REST API для взаємодії між компонентами системи.

Особливу увагу було приділено проєктуванню та реалізації бази даних, а також забезпеченню безпеки інформації. Для підвищення рівня захисту даних використано розділення інформації між двома базами даних: MainDb та AuthDb, що дозволило ізолювати користувацькі дані та платіжну інформацію від основної операційної бази.

Проведено комплексне тестування функціоналу, продуктивності та стабільності роботи системи. Перевірено коректність роботи десктоп-застосунку, серверної частини, вебінтерфейсу, QR-кодів, механізму онлайн-замовлень, звітності та інсталяційного пакета. Усі виявлені помилки були усунені, що забезпечило стабільну та коректну роботу програмного забезпечення.

На завершальному етапі роботи було сформовано інсталяційний пакет, який автоматизує процес встановлення та дозволяє швидко розгорнути систему на робочому комп'ютері користувача.

Таким чином, у результаті виконання кваліфікаційної роботи бакалавра було розроблено сучасну клієнт-серверну POS-систему для ресторану «Авего», яка відповідає поставленим вимогам, забезпечує автоматизацію основних бізнес-процесів ресторанного закладу та демонструє високий рівень практичної реалізації програмного забезпечення. Розроблену систему можна використовувати як готове рішення для закладів малого та середнього бізнесу, а також як основу для подальшого розширення функціональних можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About the Unified Modeling Language Specification Version 2.5.1 Object Management Group. URL: <https://www.omg.org/spec/UML/> (дата звернення: 14.02.2026).
2. Business Management & Online Appointment Scheduling Software Altego. URL: <https://altego.io> (дата звернення: 03.03.2026).
3. CSS: Cascading Style Sheets MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернення: 14.03.2026).
4. The SOLID Principles of Object-Oriented Programming Explained in Plain English freeCodeCamp. URL: <https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/> (дата звернення: 14.02.2026).
5. GitHub – Shane32 QRCode: A Pure C# Open Source QR Code Implementation GitHub. URL: <https://github.com/Shane32/QRCode> (дата звернення: 14.04.2026).
6. HTML: HyperText Markup Language MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (дата звернення: 14.03.2026).
7. Inno Setup Jordan Russell's Software. URL: <https://jrsoftware.org/isinfo.php> (дата звернення: 10.05.2026).
8. JavaScript MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата звернення: 15.03.2026).
9. Manage Payments With Toast POS // Toast Support. URL: <https://support.toasttab.com/en/article/New-POS-Managing-Payments> (дата звернення: 18.05.2026).
10. MySQL HeatWave Service Guide Oracle Help Center. URL: <https://docs.oracle.com/en-us/iaas/mysql-database/doc/heatwave.html> (дата звернення: 14.04.2026).
11. Overview – A Tour of C# Microsoft Learn. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview> (дата звернення: 14.03.2026).

12. Overview of ASP.NET Core Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/core/overview?view=aspnetcore-10.0> (дата звернення: 14.03.2026).

13. Poster POS – Програма автоматизації закладів громадського харчування Poster POS. URL: <https://joinposter.com/ua> (дата звернення: 04.03.2026).

14. Restaurant POS System and Software Square. URL: <https://squareup.com/us/en/point-of-sale/restaurants> (дата звернення: 03.03.2026).

15. Rester Automation of Restaurants Rester. URL: <https://rester.com/> (дата звернення: 03.03.2026).

16. Software Testing Fundamentals Software Testing Fundamentals. URL: <https://softwaretestingfundamentals.com> (дата звернення: 04.05.2026).

17. What is REST? REST API Tutorial. URL: <https://restfulapi.net/> (дата звернення: 14.02.2026).

18. What Is Visual Studio? Microsoft Learn. URL: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=visualstudio> (дата звернення: 13.03.2026).

19. What is Windows Forms Microsoft Learn. URL: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/> (дата звернення: 14.03.2026).

20. Writing a Secure RPC Client or Server Microsoft Learn. URL: <https://learn.microsoft.com/en-us/windows/win32/rpc/writing-a-secure-rpc-client-or-server> (дата звернення: 14.02.2026).

21. Гулієва Н. М., Ткачук М. А. Використання QR-кодів у клієнт-серверних POS-системах ресторанного бізнесу Сучасні інформаційні технології: від теорії до практики (СІнТех-2026): тези доповідей I Міжнародної науково-практичної конференції, 22-23 травня 2026 р. Луцьк: ЛНТУ, 2026. С. 454-456 (дата звернення: 18.05.2026).

ДОДАТКИ

Додаток А – Код обробки вебзамовлення через API

```

[ApiController]
[Route("api/[controller]")]
public class WebOrderController : ControllerBase
{
    private readonly IConfiguration _configuration;

    public WebOrderController(IConfiguration configuration)
    {
        _configuration = configuration;
    }

    [HttpPost]
    public IActionResult CreateWebOrder([FromBody]
WebOrderRequest request)
    {
        if (request == null || request.Items == null ||
request.Items.Count == 0)
            return BadRequest("Замовлення порожнє.");

        string connectionString =
_configuration.GetConnectionString("MainDb");

        using MySqlConnection conn = new
MySqlConnection(connectionString);
        conn.Open();

        using var transaction = conn.BeginTransaction();

        try
        {
            string insertOrderQuery = @"
INSERT INTO web_orders
(TableID, CustomerName, Comment, Status, Source,
CreatedAt)
VALUES
(@TableID, @CustomerName, @Comment, 'New', 'Website',
NOW())";

            using var orderCmd = new
MySqlCommand(insertOrderQuery, conn, transaction);

            orderCmd.Parameters.AddWithValue("@TableID",
request.TableId.HasValue ? request.TableId.Value : DBNull.Value);
            orderCmd.Parameters.AddWithValue("@CustomerName",
string.IsNullOrEmpty(request.CustomerName) ? DBNull.Value :
request.CustomerName);
            orderCmd.Parameters.AddWithValue("@Comment",
string.IsNullOrEmpty(request.Comment) ? DBNull.Value :
request.Comment);

            orderCmd.ExecuteNonQuery();

            int webOrderId;

```

```

        using (var lastIdCmd = new MySqlCommand("SELECT
LAST_INSERT_ID()", conn, transaction))
        {
            webOrderId =
Convert.ToInt32(lastIdCmd.ExecuteScalar());
        }

        foreach (var item in request.Items)
        {
            string itemQuery = @"
SELECT idmenu_items, name, price
FROM menu_items
WHERE idmenu_items = @MenuItemID
LIMIT 1";

            using var itemCmd = new
MySqlCommand(itemQuery, conn, transaction);
            itemCmd.Parameters.AddWithValue("@MenuItemID",
item.MenuItemId);

            using var reader = itemCmd.ExecuteReader();

            if (!reader.Read())
            {
                reader.Close();
                throw new Exception($"Позицію меню не
знайдено. ID={item.MenuItemId}");
            }

            int menuItemId =
Convert.ToInt32(reader["idmenu_items"]);
            string productName =
reader["name"]?.ToString() ?? "";
            decimal priceAtMoment = reader["price"] ==
DBNull.Value ? 0 : Convert.ToDecimal(reader["price"]);

            reader.Close();

            string insertItemQuery = @"
INSERT INTO web_order_items
(WebOrderID, MenuItemID, ProductName,
PriceAtMoment, Quantity)
VALUES
(@WebOrderID, @MenuItemID, @ProductName,
@PriceAtMoment, @Quantity)";

            using var insertItemCmd = new
MySqlCommand(insertItemQuery, conn, transaction);

            insertItemCmd.Parameters.AddWithValue("@WebOrderID", webOrderId);
            insertItemCmd.Parameters.AddWithValue("@MenuItemID", menuItemId);
            insertItemCmd.Parameters.AddWithValue("@ProductName",
productName);
            insertItemCmd.Parameters.AddWithValue("@PriceAtMoment",
priceAtMoment);

```

```

insertItemCmd.Parameters.AddWithValue("@Quantity", item.Quantity
<= 0 ? 1 : item.Quantity);

        insertItemCmd.ExecuteNonQuery();
    }

    transaction.Commit();

    return Ok(new
    {
        message = "Замовлення успішно створено",
        webOrderId = webOrderId
    });
}
catch (Exception ex)
{
    transaction.Rollback();
    return BadRequest("Помилка створення замовлення: "
+ ex.Message);
}

[HttpGet("resolve-table")]
public IActionResult ResolveTableByToken([FromQuery]
string token)
{
    if (string.IsNullOrEmpty(token))
        return BadRequest("Токен не вказано");

    string connctionString =
_configuration.GetConnectionString("MainDb");

    using MySqlConnection conn = new
MySqlConnection(connctionString);
    conn.Open();

    string query = @"
        SELECT TableID
        FROM table_qr
        WHERE QRToken =@QrToken
        LIMIT 1";

    using var cmd = new MySqlCommand(query, conn);
    cmd.Parameters.AddWithValue("@QrToken", token);

    object result = cmd.ExecuteScalar();

    if (result == null || result == DBNull.Value)
        return NotFound("Стіл по токену не знайдено.");

    int tableId = Convert.ToInt32(result);

    return Ok(new
    {
        tableId = tableId
    });
}

```

```

[HttpGet("current-by-token")]
public IActionResult GetCurrentOrderByToken([FromQuery]
string token)
{
    if (string.IsNullOrEmpty(token))
        return BadRequest("Токен не вказано.");

    string connectionString =
_configuration.GetConnectionString("MainDb");

    using MySqlConnection conn = new
MySqlConnection(connectionString);
    conn.Open();

    try
    {
        int tableId;
        string tableName = "";

        string tableQuery = @"
            SELECT t.TableID, t.Name
            FROM table_qr qr
            INNER JOIN tables t ON t.TableID = qr.TableID
            WHERE qr.QrToken = @QrToken
            LIMIT 1";

        using (var tableCmd = new MySqlCommand(tableQuery,
conn))
        {
            tableCmd.Parameters.AddWithValue("@QrToken",
token);

            using var reader = tableCmd.ExecuteReader();

            if (!reader.Read())
                return NotFound("Стіл по токену не
знайдено.");

            tableId = Convert.ToInt32(reader["TableID"]);

            // якщо буде дубляж стіл: стіл 1 типу просто
            tableName = reader["Name"]?.ToString() ??
$"Стіл {tableId}";
        }

        int checkId = 0;
        int orderNumber = 0;
        int tableOrderNumber = 0;
        string checkStatus = "";
        string paymentStatus = "Unpaid";

        string checkQuery = @"
            SELECT CheckID, OrderNumber, TableOrderNumber,
Status

```

```

        FROM checks
        WHERE TableID = @TableID
            AND Status IN ('Open', 'Pending',
'PrecheckPrinted')
            ORDER BY CheckID DESC
        LIMIT 1";

        using (var checkCmd = new
MySQLCommand(checkQuery, conn))
        {
            checkCmd.Parameters.AddWithValue("@TableID",
tableId);

            using var reader = checkCmd.ExecuteReader();
            if(!reader.Read())
            {
                return Ok(new
                {
                    tableId = tableId,
                    tableName = tableName,
                    checkId = 0,
                    checkStatus = "NoActiveCheck",
                    paymentStatus = "Unpaid",
                    lastWebOrderStatis = "",
                    items = new List<object>(),
                    total = 0m
                });
            }

            checkId = Convert.ToInt32(reader["CheckID"]);
            orderNumber =
Convert.ToInt32(reader["OrderNumber"]);
            tableOrderNumber =
Convert.ToInt32(reader["TableOrderNumber"]);
            checkStatus = reader["Status"]?.ToString() ??
"";

        }

        if (checkStatus == "Paid" || checkStatus ==
"Completed")
            paymentStatus = "Paid";

        string lastWebOrderStatus = "";

        string webOrderStatusQuery = @"
        SELECT Status
        FROM web_orders
        WHERE TableID = @TableID
        ORDER BY WebOrderID DESC
        LIMIT 1";

        using (var statusCmd = new
MySQLCommand(webOrderStatusQuery, conn))
        {

```

```

        statusCmd.Parameters.AddWithValue("@TableID",
tableId);

        object statusObj = statusCmd.ExecuteScalar();
        lastWebOrderStatus = statusObj == null ||
statusObj == DBNull.Value
        ? ""
        : statusObj.ToString();
    }

    var items = new List<object>();

    decimal total = 0m;

    string itemsQuery = @"
SELECT ProductName,
       SUM(Quantity) AS Quantity,
       PriceAtMoment,
       SUM(Quantity * PriceAtMoment) AS ItemTotal
FROM orders
WHERE TableID = @TableID
   AND TableOrderNumber = @TableOrderNumber
   AND OrderNumber = @OrderNumber
   AND Status IN ('Open', 'Pending', 'PrecheckPrinted')
GROUP BY ProductName, PriceAtMoment
ORDER BY ProductName";

    using (var itemsCmd = new MySqlCommand(itemsQuery,
conn))
    {
        itemsCmd.Parameters.AddWithValue("@TableID",
tableId);

        itemsCmd.Parameters.AddWithValue("@TableOrderNumber",
tableOrderNumber);

        itemsCmd.Parameters.AddWithValue("@OrderNumber", orderNumber);

        using var reader = itemsCmd.ExecuteReader();

        while (reader.Read())
        {
            string productName =
reader["ProductName"]?.ToString() ?? "";
            decimal quantity = reader["Quantity"] ==
DBNull.Value ? 0 : Convert.ToDecimal(reader["Quantity"]);
            decimal priceAtMoment =
reader["PriceAtMoment"] == DBNull.Value ? 0 :
Convert.ToDecimal(reader["PriceAtMoment"]);
            decimal itemTotal = reader["ItemTotal"] ==
DBNull.Value ? 0 : Convert.ToDecimal(reader["ItemTotal"]);

            total += itemTotal;

            items.Add(new
            {
                productName = productName,

```

```

        quantity = quantity,
        priceAtMoment = priceAtMoment,
        sum = itemTotal
    });
}

return Ok(new
{
    tableId = tableId,
    tableName = tableName,
    checkId = checkId,
    checkStatus = checkStatus,
    paymentStatus = paymentStatus,
    lastWebOrderStatus = lastWebOrderStatus,
    items = items,
    total = total
});
}
catch (Exception ex)
{
    return BadRequest("Помилка отримання поточного
замовлення: " + ex.Message);
}
}

```

Додаток Б – Код завантаження меню на сайті

```

async function loadMenu() {
  try {
    const response = await fetch(apiUrl);

    if (!response.ok) {
      throw new Error("Не вдалося отримати дані з API");
    }

    allItems = await response.json();

    if (!allItems || allItems.length === 0) {
      menuContainer.innerHTML = `<div class="empty-state">Меню поки порожнє.</div>`;
      return;
    }

    const categories = getCategories(allItems);
    activeCategory = categories[0] || null;

    const subCategories = getSubCategories(allItems,
activeCategory);
    activeSubCategory = subCategories[0] || null;

    renderLayout();
  }
  catch (error) {
    console.error(error);
    menuContainer.innerHTML = `
      <div class="error-state">
        Помилка завантаження меню. Перевір API та
підключення: ${error.message}
      </div>
    `;
  }
}

function getCategories(items) {
  return [...new Set(items.map(x => x.category || "Інше"))];
}

function getSubCategories(items, category) {
  return [
    ...new Set(
      items
        .filter(x => (x.category || "Інше") === category)
        .map(x => x.subCategory || "Без підкатегорії")
    )
  ];
}

function getFilteredItems() {
  return allItems.filter(item => {
    const category = item.category || "Інше";
    const subCategory = item.subCategory || "Без
підкатегорії";
  });
}

```

```

        return category === activeCategory && subCategory ===
activeSubCategory;
    });
}

function renderLayout() {
    menuContainer.innerHTML = `
        <section class="menu-layout">
            <aside id="categoryNav" class="category-nav"></aside>
            <div class="menu-main">
                <div id="subcategoryNav" class="subcategory-
nav"></div>
                <div id="menuGrid" class="menu-grid"></div>
            </div>
        </section>
    `;

    renderCategories();
    renderSubCategories();
    renderItem();
}

function renderCategories() {
    const container = document.getElementById("categoryNav");
    const categories = getCategories(allItems);

    container.innerHTML = `
        <h2 class="nav-title">Категории</h2>
        ${categories.map(category =>
"active" : ""}
            <button class="nav-btn ${category === activeCategory ?
'active" : ""}"
            onclick="selectCategory('${escapeJs(category)}
'")>
                ${escapeHtml(category)}
            </button>
        `).join("")}
    `;
}

function renderSubCategories() {
    const container = document.getElementById("subcategoryNav");
    const subCategories = getSubCategories(allItems,
activeCategory);

    if (!subCategories.includes(activeSubCategory)) {
        activeSubCategory = subCategories[0] || null;
    }

    container.innerHTML = `
        <h3 class="subnav-title">${escapeHtml(activeCategory ||
"" )}</h3>
        <div class="subnav-buttons">
            ${subCategories.map(sub => `
                <button class="subnav-btn ${sub ===
activeSubCategory ? "active" : ""}"
                onclick="selectSubCategory('${escapeJs(sub
)})">
                    ${escapeHtml(sub)}
                </button>
            `).join("")}
        </div>
    `;
}

```

```

    `;
  }
  </div>
}

function renderItem() {
  const grid = document.getElementById("menuGrid");
  const items = getFilteredItems();

  if (!items.length) {
    grid.innerHTML = `<div class="empty-state">у цій
    підкатегорії ще немає позицій меню.</div>`;
    return;
  }

  grid.innerHTML = items.map(item => `
    <article class="menu-card">
      ${createImageBlock(item)}
      <div class="menu-body">
        <div>
          <h4 class="menu-name">${escapeHtml(item.name
|| "")}</h4>
          <p class="menu-
description">${escapeHtml(item.description || "")}</p>
        </div>
        <div class="menu-footer">
          <div class="menu-
price">${formatPrice(item.price)} грн</div>
        </div>
        <button class="add-to-cart-btn"
onclick="addToCart(${item.id})">
          Додати в кошик
        </button>
        </div>
      </article>
    `).join("");
}

function selectCategory(category) {
  activeCategory = category;
  const subCategories = getSubCategories(allItems,
activeCategory);
  activeSubCategory = subCategories[0] || null;

  renderCategories();
  renderSubCategories();
  renderItem();
}

function selectSubCategory(subCategory) {
  activeSubCategory = subCategory;
  renderSubCategories();
  renderItem();
}

function createImageBlock(item) {
  if (item.imagePath) {
    const imageUrl =
`${window.location.origin}${item.imagePath}`;
    return
    <div class="menu-image-wrap">

```

```

        Немає фото</div>';"
        />
    </div>
    `;
}

return `
    <div class="menu-image-wrap">
        <div class="menu-image-placeholder">Немає фото</div>
    </div>
    `;
}

function formatPrice(value) {
    if (value === null || value === undefined || value === "") {
        return "0";
    }

    return Number(value).toLocaleString("uk-UA");
}

function addToCart(itemId) {

    const menuItem = allItems.find(x => x.id === itemId);

    if (!menuItem) {
        console.log("menuItem не знайдено");
        return;
    }

    const existing = cart.find(x => x.id === itemId);

    if (existing) {
        existing.quantity += 1;
    } else {
        cart.push({
            id: menuItem.id,
            name: menuItem.name,
            price: Number(menuItem.price || 0),
            quantity: 1
        });
    }

    console.log("cart =", cart);
    renderCart();
}

```

Додаток В – Код завантаження замовлень столу

```

private void LoadTableOrders()
{
    BasketPanel.Controls.Clear();
    decimal sum = 0;

    if (CurrentTable == 0 || CurrentOrderNumber == 0 ||
CurrentTableOrderNumber == 0)
        return;

    var orders = MainDb.GetList<Order>(@"
SELECT * FROM orders
WHERE TableID=@tableId
AND OrderNumber=@orderNumber
AND TableOrderNumber=@tableOrderNumber
AND Status IN ('Pending', 'PrecheckPrinted', 'Open')
ORDER BY OrderTime
",
        ("@tableId", CurrentTable),
        ("@orderNumber", CurrentOrderNumber),
        ("@tableOrderNumber", CurrentTableOrderNumber)
    );

    foreach (var o in orders)
    {
        Color itemColor = Color.Gainsboro;

        if(o.Status == "PrecheckPrinted")
        {
            itemColor = Color.Khaki;
        }
        else if(o.Status == "Open" && o.OrderSource ==
"website")
        {
            itemColor = Color.MediumPurple;
        }

        else if(o.Status == "Open")
        {
            itemColor = Color.Gainsboro;
        }

        Button item = new Button
        {
            width = 250,
            Height = 60,
            Text = $"{o.ProductName}
x{o.Quantity:0.##}" +
            (string.IsNullOrEmpty(o.Modifiers) ? "" : $"
({o.Modifiers})") +
            $"\\n{o.PriceAtMoment * o.Quantity:0.##} грн",
            FlatStyle = FlatStyle.Flat,
            FlatAppearance = { BorderSize = 1 },
            BackColor = itemColor,

```

```

        ForeColor = Color.Black,
        Font = new Font("Segoe UI", 10,
FontStyle.Bold),
        Enabled = true,
        Tag = o.Id// можна зберегти Id замовлення
для подальших операцій (зміна, видалення)

    };
    item.Click += (s, e) =>
    {
        if(SelectedCartButton != null)
        {
            if(SelectedCartButton.Tag is int
selectedOrderId)
            {
                var selectedOrder =
orders.FirstOrDefault(x => x.Id == selectedOrderId);
                if(selectedOrder != null)
                {
                    if (selectedOrder.Status ==
"PrecheckPrinted")
                        SelectedCartButton.BackColor =
Color.Khaki;
                    else if (selectedOrder.Status ==
"Open" && selectedOrder.OrderSource == "Website")
                        SelectedCartButton.BackColor =
Color.MediumPurple;
                    else SelectedCartButton.BackColor
= Color.Gainsboro;
                }
            }
            else
            {
                SelectedCartButton.BackColor =
Color.WhiteSmoke;
            }
        }
        else
        {
            SelectedCartButton.BackColor =
Color.WhiteSmoke;
        }
    }
    SelectedCartButton = item;
    SelectedCartButton.BackColor =
Color.LightSkyBlue;
};

    BasketPanel.Controls.Add(item);
    sum += o.PriceAtMoment * o.Quantity;
}
UpdateTotalWithDiscount(sum);
}

```