Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»
(повне найменування вищого навчального закладу)

Відділення телекомунікацій та електронних систем
(назва відділення)
Циклова комісія комп'ютерної інженерії
(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

бакалавра
(освітній ступінь)

на тему: **Розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»**

Виконав: студент VI курсу, групи КІб-602

Спеціальності: **123 Комп'ютерна інженерія**
(шифр і назва спеціальності)

_____ Сергій НАВРОЦЬКИЙ
(підпис) (ім'я та прізвище)

Керівник _____ Василь ПИЖ
(підпис) (ім'я та прізвище)

Рецензент _____
(підпис) (ім'я та прізвище)

Тернопіль – 2025

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
ІМЕНІ ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерної інженерії
Освітній ступінь бакалавр
Освітньо-професійна програма: Комп'ютерна інженерія
Спеціальність 123 Комп'ютерна інженерія
Галузь знань: 12 Інформаційні технології

ЗАТВЕРДЖУЮ
Голова циклової комісії
комп'ютерної інженерії
_____ Андрій ЮЗЬКІВ
"06" травня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Навроцький Сергій Зіновійович
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи: **Розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»**

керівник роботи: Пиж Василь Степанович

затверджені наказом Відокремленого структурного підрозділу «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя» від 05.05.2025 р. № 4/9-217.

2. Строк подання студентом кваліфікаційної роботи 20 червня 2025 року.

3. Вихідні дані до роботи: мова програмування C++, фреймворк Qt, система керування базами даних PostgreSQL.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Загальний розділ. Розробка технічного та робочого проєкту. Спеціальний розділ. Економічний розділ. Безпека життєдіяльності, основи охорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма варіантів використання. Схема структурна програмного забезпечення. ER-діаграма бази даних. UML-діаграма класів програмного забезпечення. Блок-схема алгоритму роботи функції. Таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Оксана РЕДЬКВА заст. директора з НВР		
Безпека життєдіяльності, основи охорони праці	Володимир ШТОКАЛО викладач		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	06.05	
2	Збір і узагальнення інформації по роботі	19.05	
3	Написання першого розділу	23.05	
4	Розробка технічного та робочого проекту	28.05	
5	Написання спеціального розділу	03.06	
6	Розрахунок економічної частини	05.06	
7	Написання розділу охорони праці	07.06	
8	Виконання графічної частини	12.06	
9	Оформлення проекту	16.06	
10	Проходження нормоконтролю	18.06	
11	Попередній захист роботи	20.06	
12	Захист роботи		

7. Дата видачі завдання 06 травня 2025 року

Студент _____ Сергій НАВРОЦЬКИЙ
(підпис) (ім'я та прізвище)

Керівник роботи _____ Василь ПИЖ
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Метою кваліфікаційної роботи бакалавра є розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин».

Розроблений програмний продукт призначений для автоматизації ключових операційних процесів у ресторані «Старий Млин». Ця система забезпечує ефективний прийом, обробку та відстеження замовлень напоїв, замінюючи традиційні ручні методи цифровою платформою.

Проєкт охоплює основні функціональні домени: управління замовленнями (створення, редагування, відстеження статусів), управління меню (каталог напоїв, ціноутворення, контроль доступності) та управління персоналом (автентифікація, авторизація, розмежування ролей). Архітектура системи побудована на базі C++ та фреймворку Qt, з використанням PostgreSQL як основної системи управління базами даних, що гарантує високу продуктивність, надійність та кросплатформенність.

Інтерфейс користувача розроблено з акцентом на інтуїтивність та простоту, забезпечуючи швидке освоєння та мінімізацію помилок. Проведений комплексний процес тестування підтвердив стабільність функціонування системи та її готовність до практичного застосування.

Актуальність проєкту полягає у його здатності значно підвищити ефективність обслуговування клієнтів, зменшити операційні витрати та оптимізувати адміністративні процеси в закладах громадського харчування. Система може бути успішно застосована не лише у ресторані «Старий Млин», але й у інших подібних закладах, що прагнуть цифрової трансформації своїх робочих процесів..

Ключові слова: система управління замовленнями, автоматизація замовлень, піцерії, піцерія, C++17, QT6, PostgreSQL, POS-система, управління меню.

ANNOTATION

The purpose of the bachelor's qualification paper is the development of the software "Flamingo Pizzeria Waiter Order Window."

The goal of the bachelor's qualification paper is to develop software for taking and processing bartender orders at the "Stary Mlyn" restaurant.

The developed software product is designed to automate key operational processes within the "Stary Mlyn" restaurant. This system ensures efficient reception, processing, and tracking of beverage orders, replacing traditional manual methods with a digital platform.

The project covers core functional domains: order management (creation, editing, status tracking), menu management (beverage catalog, pricing, availability control), and personnel management (authentication, authorization, role separation). The system's architecture is built on C++ and the Qt framework, utilizing PostgreSQL as the primary database management system, guaranteeing high performance, reliability, and cross-platform compatibility.

The user interface has been designed with an emphasis on intuitiveness and simplicity, ensuring quick adoption and minimizing errors. A comprehensive testing process confirmed the system's operational stability and its readiness for practical application.

The project's relevance lies in its ability to significantly enhance customer service efficiency, reduce operational costs, and optimize administrative processes in food service establishments. The system can be successfully implemented not only at "Stary Mlyn" restaurant but also in other similar venues aiming for the digital transformation of their workflows.

Keywords: order management system, order automation, pizzerias, pizzeria, C++17, Qt6, PostgreSQL, POS system, menu management.

ЗМІСТ

ВСТУП

У сучасному ресторанному бізнесі ефективність та швидкість обслуговування клієнтів — ключові фактори успіху. Ресторан «Старий Млин» прагне оптимізувати свої внутрішні процеси, підвищити якість сервісу та забезпечити безперебійну роботу персоналу. Для досягнення цих цілей було розроблено програмне забезпечення «Система управління замовленнями бармена».

Ця система створена для автоматизації процесів прийому, обробки та виконання замовлень напоїв. Вона дозволить мінімізувати ручні операції, зменшити кількість помилок та значно прискорити обслуговування. Система формує єдине цифрове середовище для координації роботи офіціантів, барменів та адміністративного персоналу, забезпечуючи централізований облік замовлень і контроль за їх виконанням.

Інтуїтивно зрозумілий інтерфейс та надійна архітектура, розроблена на базі C++ і Qt, разом із використанням PostgreSQL, гарантують стабільність, високу продуктивність та легкість масштабування. Розробка цього програмного забезпечення є стратегічним кроком для «Старого Млина» на шляху до підвищення операційної ефективності та покращення загального клієнтського досвіду.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

1.2 Технічне завдання

Аналіз наявних програмних рішень для управління замовленнями в ресторанному бізнесі є важливим етапом перед розробкою власної системи. Це дозволяє ідентифікувати кращі практики, оцінити функціональні можливості та архітектурні підходи конкурентів, а також виявити потенційні ніші та унікальні переваги для власного продукту. Огляд зосереджується на платформах, що пропонують комплексну автоматизацію процесів у закладах громадського харчування:

1. POS-системи та Системи Управління Ресторанами (RMS).

Більшість сучасних рішень для ресторанів є інтегрованими POS-системами (Point of Sale) або комплексними системами управління ресторанами (Restaurant Management Systems – RMS), які охоплюють широкий спектр операцій.

Toast POS – ця система є однією з провідних на ринку США, орієнтована на широкий спектр закладів – від невеликих кафе до великих мереж. Toast пропонує інтегровані модулі для прийому замовлень (через термінали офіціантів, онлайн-замовлення), управління столами, кухнею (KDS – Kitchen Display System), звітності, управління персоналом та програмами лояльності. Система працює на хмарній основі та використовує Android-планшети для мобільних POS-терміналів [1]. Її особливістю є фокус на єдиній інтегрованій екосистемі, що спрощує взаємодію всіх підсистем.

Lightspeed Restaurant – цей продукт є ще одним відомим гравцем на ринку, який пропонує хмарне POS-рішення для ресторанів. Lightspeed акцентує увагу на інтуїтивно зрозумілому інтерфейсі, управлінні столиками, налаштовуваних меню, розширеній аналітиці продажів та інтеграції з платіжними системами. Підтримує роботу на iPad, що забезпечує мобільність офіціантів. Система дозволяє керувати

кількома закладами з централізованої панелі [2].

Poster POS – українське хмарне POS-рішення, популярне серед ресторанів та кафе в Україні та Східній Європі. Poster надає функціонал для управління замовленнями, складом, фінансами, клієнтами та персоналом. Його перевагами є відносно низька вартість підписки, простота налаштування та локалізована підтримка. Система підтримує роботу на планшетах (Android, iOS) та комп'ютерах [3].

2. Спеціалізовані Рішення для Барів та Кухні

Окрім комплексних RMS, існують також більш нішеві рішення, що фокусуються на конкретних аспектах роботи ресторану:

– Kitchen Display Systems (KDS).

Багато POS-систем включають KDS, але також існують окремі KDS-рішення. Наприклад, Expeditor або модулі в складі більших RMS, які перетворюють замовлення офіціантів на електронні черги для кухарів та барменів, відображаючи статус замовлень, час очікування та пріоритет. Це дозволяє підвищити ефективність роботи кухні/бару та зменшити кількість помилок [4].

3. Онлайн-Платформи для Замовлень та Доставки

Зростання популярності онлайн-замовлень та доставки стимулювало розвиток платформ, які інтегруються з ресторанними системами або працюють автономно.

DoorDash, Uber Eats, Glovo (для доставки) – ці платформи надають ресторани можливість приймати замовлення онлайн та організовувати доставку. Вони часто мають власні планшети або API для інтеграції з POS-системами ресторанів, дозволяючи барам отримувати замовлення безпосередньо в систему [5].

Аналіз існуючих рішень показує, що ринок програмного забезпечення для ресторанів є насиченим, але переважно орієнтований на комплексні та часто дорогі POS/RMS. Багато з них пропонують розширений функціонал, що може бути надлишковим для невеликих або середніх закладів, які шукають цілеспрямовану автоматизацію певних процесів, наприклад, лише роботи бару.

Розробка власної системи управління замовленнями бармена для «Старого Млина» з фокусом на C++/Qt та PostgreSQL дозволяє створити високоефективне, надійне та кросплатформенне рішення, яке, на відміну від багатьох комерційних аналогів, може бути точно адаптовано під конкретні потреби замовника, уникаючи надлишкового функціоналу та пов'язаних з ним витрат. Це дозволяє зосередитися на ключових бізнес-процесах бармена, таких як швидкий прийом, обробка та виконання замовлень, з можливістю подальшого масштабування та інтеграції.

1.2.1 Найменування та область застосування

Тема кваліфікаційної роботи бакалавра: «Розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»».

Найменування програмного забезпечення «Система управління замовленнями бармена ресторану «Старий Млин».

1.2.2 Призначення розробки

Програмне забезпечення розробляється для автоматизації низки ключових бізнес-процесів. Це включає управління замовленнями, яке охоплює прийом замовлень від офіціантів, відстеження статусу їх виконання, формування черги виконання для бармена та облік готових замовлень. Система також забезпечуватиме управління асортиментом, що передбачає ведення каталогу напоїв з цінами, їх категоризацію, контроль доступності позицій меню та оновлення цін і характеристик. Функціонал управління персоналом буде реалізований через реєстрацію та автентифікацію співробітників, розмежування прав доступу за ролями, ведення облікових записів персоналу та контроль активності користувачів. На додаток, система надаватиме засоби аналітики та звітності для формування звітів про продажі, аналізу ефективності роботи

персоналу, збору статистики популярності напоїв та контролю виконання замовлень.

Основними цілями створення даної системи є підвищення швидкості обслуговування клієнтів на 30-40%, зменшення кількості помилок при оформленні замовлень, покращення контролю за їх виконанням та автоматизація процесу формування звітності для управлінського персоналу. Крім того, система сприятиме стандартизації робочих процесів персоналу.

Система розроблена для застосування в ресторанах середнього класу з кількістю посадочних місць від 50 до 150. Вона також підходить для кафе, пивних закладів, барів, пабів та може бути масштабована для використання в мережах закладів громадського харчування.

1.2.3 Вимоги до програмного забезпечення

1.2.3.1 Вимоги до функціональних характеристик

Вимоги до функцій (задач), що мають виконуватись системою розглянемо в розрізі наступних підсистем:

– Підсистема управління замовленнями.

Ця підсистема повинна забезпечувати повний цикл управління замовленнями, від їх створення до завершення. Передбачається, що офіціанти матимуть змогу створювати замовлення для конкретних столів, вказуючи перелік напоїв, їх кількість та будь-які особливості приготування. Час створення такого замовлення не повинен перевищувати 30 секунд, при цьому введені дані мають проходити обов'язкову валідацію. Система повинна реалізовувати функціонал відстеження статусу замовлень, який змінюватиметься від "Нове" до "Видано" або "Скасовано". При кожній зміні статусу буде генеруватися автоматичне сповіщення, а всі зміни фіксуватимуться в журналі в реальному часі. Для бармена буде розроблена черга замовлень, пріоритизована за алгоритмом FIFO (First-In, First-Out), з врахуванням складності приготування та часу очікування. Передбачається можливість фільтрації черги за різними критеріями. Також

система повинна дозволяти редагування активних замовлень до моменту початку їх приготування, що включатиме додавання, видалення позицій або зміну їх кількості. Функціонал редагування потребуватиме контролю прав доступу та обов'язкового журналювання всіх внесених змін.

– Підсистема управління меню.

Підсистема управління меню буде включати централізований каталог напоїв, який слугуватиме реєстром усіх доступних позицій з такими атрибутами, як назва, ціна, категорія, опис та час приготування. Цей каталог має бути структурованим та забезпечувати швидкий пошук необхідних позицій. Для адміністрування вартості напоїв буде реалізовано функціонал управління цінами, що дозволятиме оновлювати ціни, вносити масові зміни та зберігати історію цін, з обов'язковою валідацією коректності введених даних. Також буде розроблено механізм контролю доступності, що дозволить управляти наявністю позицій у меню. Статуси доступності включатимуть "Доступний", "Тимчасово недоступний" та "Знятий з виробництва", і будь-які зміни повинні миттєво відображатися в інтерфейсі замовлень..

– Підсистема управління персоналом.

У рамках підсистеми управління персоналом буде розроблено механізм автентифікації користувачів, який передбачає вхід до системи за допомогою логіна та пароля, з потенційною можливістю розширення методів автентифікації. Паролі будуть хешуватися, а акаунти блокуватимуться після певної кількості невдалих спроб входу. Час входу користувача до системи має становити менше 5 секунд. Авторизація та ролі забезпечать розмежування доступу до функцій системи відповідно до ролей користувачів: офіціанти зможуть створювати та переглядати лише власні замовлення; бармени отримають доступ до управління чергою замовлень та редагування меню; адміністратори матимуть повний доступ до всіх функцій системи. Цей функціонал вимагає гранулярного контролю прав та механізмів аудиту. Також буде реалізовано управління обліковими записами, що включатиме функціонал створення, редагування та деактивації акаунтів користувачів з атрибутами: ПІБ, роль, контактні дані та статус активності, з

обов'язковою валідацією даних та збереженням історії змін.

– Підсистема звітності.

Підсистема звітності буде надавати можливість генерувати звіти про продажі за різними періодами (день, тиждень, місяць, довільний проміжок) та критеріями, з групуванням даних за напоями, категоріями або офіціантами. Передбачається можливість експорту звітів у формати PDF, Excel та CSV. Час генерації стандартних звітів не повинен перевищувати 10 секунд. Окрім того, буде розроблено функціонал аналізу ефективності, що охоплюватиме метрики продуктивності персоналу та процесів, такі як кількість замовлень, середній час їх виконання та популярність напоїв. Ці показники будуть візуалізовані у вигляді графіків та діаграм для забезпечення точності розрахунків та наочності представлення даних.

1.2.3.2 Вимоги до надійності

Розроблювана система управління замовленнями для бармена ресторану «Старий Млин» повинна відповідати високим стандартам надійності, що є критично важливим для забезпечення безперебійної роботи закладу. Ці вимоги охоплюють три ключові аспекти: надійність функціонування, стійкість до відмов та ефективний контроль помилок.

Майбутня система має демонструвати високий рівень доступності, який планується на рівні не менше 99.5% від загального робочого часу (що еквівалентно 8760 годинам на рік). Це означає мінімізацію простоїв та забезпечення постійного доступу до функціоналу. Середній час між відмовами (MTBF) системи повинен становити не менше 720 годин, що свідчить про її здатність працювати без збоїв протягом тривалого періоду. У випадку виникнення відмови, середній час на її відновлення (MTTR) не повинен перевищувати 30 хвилин. Це гарантуватиме швидке повернення системи до робочого стану. Крім того, ціль точки відновлення (RPO – Recovery Point Objective), що визначає максимальний обсяг втрати даних, встановлюється на рівні не більше 15 хвилин, а ціль часу відновлення (RTO – Recovery Time Objective) – не більше 5 хвилин. Ці показники забезпечують мінімальні втрати інформації та оперативне відновлення

функціональності після будь-яких непередбачених ситуацій.

Стійкість до відмов.

Для забезпечення стійкості до відмов система повинна реалізовувати механізми автоматичного збереження даних кожні 30 секунд, мінімізуючи потенційні втрати інформації у випадку непередбаченого завершення роботи. Щоденно о 02:00 буде виконуватися повне резервне копіювання даних. У разі збою системи, передбачається автоматичне відновлення з останньої збереженої точки. При кожному запуску системи здійснюватиметься контроль цілісності бази даних, що дозволить виявляти та усувати можливі пошкодження. Всі критичні операції та системні події будуть детально журналюватися для подальшого аналізу та аудиту.

Система повинна включати комплексні механізми контролю помилок. Це передбачає обов'язкову валідацію вводу на всіх етапах взаємодії користувача з системою, що запобігатиме внесенню некоректних даних. Обробка виключень буде реалізована таким чином, щоб забезпечувати "graceful degradation" – система повинна коректно реагувати на помилки без втрати даних чи критичних збоїв. Користувачам будуть надаватися зрозумілі та інформативні повідомлення про помилки, що допоможе їм усувати незначні проблеми самостійно. Для розробників та адміністраторів буде вестися детальний лог помилок, який міститиме всю необхідну інформацію для швидкого виявлення причин збоїв та їх ефективного усунення.

1.2.3.3 Вимоги до складу і параметрів технічних засобів

Для ефективного функціонування системи управління замовленнями бармена необхідно забезпечити відповідність технічних засобів визначеним параметрам. Ці вимоги охоплюють склад апаратного забезпечення та його конфігурацію, що є критично важливим для стабільної та продуктивної роботи системи.

Система розроблена для розгортання на стаціонарних робочих станціях, які є основними точками взаємодії персоналу з програмним забезпеченням.

Програмне забезпечення передбачає використання двох робочих станцій. Одна з них буде розташована у барній зоні, забезпечуючи прямий доступ барменам для обробки замовлень. Друга станція розміщуватиметься в адміністративній зоні, призначена для адміністративного персоналу та управління каталогом меню і обліковими записами співробітників.

Для забезпечення оптимальної продуктивності та надійності системи, кожна робоча станція повинна відповідати наступним мінімальним та рекомендованим параметрам. Апаратне забезпечення:

– Операційна система: Система сумісна з Windows 10/11, Ubuntu 20.04+.

– Архітектура процесора: Підтримуються архітектури x86-64 та ARM64 (зокрема, Apple Silicon).

– Оперативна пам'ять (RAM):

  – Мінімально: 4 ГБ.

  – Рекомендовано: 8 ГБ для забезпечення вищої швидкодії та можливості запуску інших додатків без втрати продуктивності .

– Дисковий простір:

  – Мінімально: 10 ГБ вільного місця на накопичувачі для інсталяції системи та зберігання даних.

– Рекомендовано: Використання SSD-накопичувача значно покращить швидкість завантаження додатку та загальну чутливість системи. Обсяг даних для програми становить близько 20 МБ, а файл бази даних зростає приблизно на 1 МБ на кожні 1000 замовлень.

– Мережевий адаптер: Для локальної роботи системи мережеве з'єднання не є критично необхідним для функціоналу версії 1.0, оскільки вона функціонує в автономному режимі з локальною базою даних SQLite [6]. Проте, наявність Ethernet 100 Мбіт/с або Wi-Fi є базовою вимогою для підключення до інших ресурсів або для майбутніх оновлень системи.

Програмне забезпечення:

– Система управління базами даних (СУБД): Система розгортається для

використання PostgreSQL 14+ [7].

Для коректного відображення інтерфейсу та функціонування компонентів необхідно забезпечення сумісності з встановленими системними шрифтами та бібліотеками, що є частиною стандартних дистрибутивів підтримуваних операційних систем.

Дотримання цих вимог дозволить забезпечити стабільну, надійну та продуктивну роботу системи управління замовленнями, що відповідає поточним потребам ресторану "Старий Млин" та закладає основу для подальшого розвитку.

1.2.4 Вимоги до програмної документації

Програмна документація є невід'ємною частиною проєкту розробки програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин». Вона повинна забезпечувати повне та чітке розуміння архітектури, функціональності, принципів роботи та процедур розгортання і підтримки системи. Наявність якісної документації є критичною для ефективного супроводу проєкту, навчання нових користувачів та подальшого розвитку програмного продукту.

Згідно індивідуального завдання необхідно розробити наступну програмну документацію:

- інструкція з інсталяції програмного забезпечення;
- інструкція з використання тестових наборів;
- інструкція з експлуатації програмного забезпечення.

1.2.5 Техніко-економічні показники

Техніко-економічні показники (ТЕП) відображають ефективність інвестицій у розробку програмного забезпечення (ПЗ), співвідносячи витрати на створення системи з очікуваними вигодами. Для системи управління замовленнями бармена ресторану «Старий Млин» ці показники розглядаються у контексті поточної версії

1.0 та планів на подальший розвиток.

Економічні показники початкової стадії розробки включають вартість ресурсів та терміни виконання, що безпосередньо впливають на загальний бюджет проєкту.

Загальна тривалість розробки повинна становити 15 календарних днів.

Детальна вартість розробки не вказана у наданому технічному завданні. Однак, з огляду на використання відкритого фреймворку Qt, мови C++ та бази даних PostgreSQL, можна припустити оптимізацію ліцензійних витрат. Основними статтями витрат є оплата праці команди розробки ПЗ, що включає керівника проєкту, програміста та тестувальника.

Стек технологій, який повинен використовуватись при розробці:

Програмне забезпечення: C++17, Qt 6.5+, PostgreSQL 14+, що дозволяє уникнути значних ліцензійних відрахувань на етапі MVP.

Інструменти розробки: Qt Creator, qmake, Git – це стандартизовані та переважно безкоштовні або доступні інструменти, що також сприяє зниженню витрат.

1.2.6 Стадії та етапи розробки

Розробка програмного забезпечення «Система управління замовленнями бармена» для ресторану «Старий Млин» здійснюється відповідно до класичних стадій життєвого циклу розробки ПЗ, що забезпечує структурований підхід, контроль якості та своєчасне досягнення поставлених цілей. Проєкт охоплює кілька ключових стадій, кожна з яких поділяється на послідовні етапи:

1. Стадія. аналіз та проектування.

Ця початкова стадія зосереджена на глибокому розумінні потреб замовника та розробці архітектурних рішень для майбутньої системи.

Етап 1.1. Збір та аналіз вимог.

Детальне вивчення бізнес-процесів ресторану, інтерв'ю з ключовими зацікавленими сторонами, такими як офіціанти, бармени, адміністратор та

менеджер. Систематизація функціональних та нефункціональних вимог до системи.

Результат: Формування початкової версії Технічного Завдання, що включає призначення, цілі, сферу застосування, функціональні та нефункціональні вимоги.

Етап 1.2. Проектування архітектури системи.

Розробка загальної архітектури програмного забезпечення, визначення основних компонентів та їх взаємодії. Вибір технологічного стеку, що включає мову програмування, фреймворки та систему управління базами даних.

Результат: Документ з архітектурою ПЗ (наприклад, патерн MVVM, структура шарів Presentation, Business Logic, Data Access, Infrastructure) та обґрунтування вибору технологій (C++17, Qt 6.5+, SQLite/PostgreSQL).

Етап 1.3. Проектування бази даних.

Розробка логічної та фізичної моделі даних, визначення таблиць, полів, зв'язків (Foreign Keys) та індексів.

Результат: Схема бази даних (для employees, drinks, orders, order_items) та правила її нормалізації (наприклад, 3NF).

Етап 1.4 Проектування інтерфейсу користувача (UI/UX).

Розробка макетів та прототипів інтерфейсу, визначення навігації, візуального стилю, принципів ергономіки та зручності використання.

Результат: Дизайн-документ з описом загальних принципів UI/UX, структури та візуального дизайну інтерфейсу.

2. Стадія. Розробка (імплементація).

Ця стадія включає безпосереднє написання коду та створення програмних модулів відповідно до розроблених специфікацій.

Етап 2.1. Розробка модуля автентифікації та авторизації.

Імплементація механізмів входу користувачів, перевірки облікових даних та розмежування прав доступу за ролями (офіціант, бармен, адміністратор).

Результат: Робочий модуль входу та менеджер авторизації.

Етап 2.2. Розробка підсистеми управління меню.

Створення функціоналу для ведення каталогу напоїв (CRUD-операції),

управління цінами та контролю доступності позицій у меню.

Результат: Модуль управління меню та репозиторій для роботи з даними напоїв.

Етап 2.3. Розробка підсистеми управління замовленнями.

Реалізація функцій створення, перегляду, фільтрації, редагування та відстеження статусу замовлень. Імплементація логіки зміни статусів та черги замовлень для бармена.

Результат: Модуль управління замовленнями та репозиторій для даних замовлень.

Етап 2.4. Розробка підсистеми управління персоналом.

Створення функціоналу для адміністраторів з додавання, редагування та деактивації облікових записів співробітників.

Результат: Модуль управління персоналом та репозиторій для даних співробітників.

Етап 2.5. Інтеграція та збірка.

З'єднання розроблених модулів у єдину систему. Налаштування взаємодії між шарами (Presentation, Business Logic, Data Access). Збірка виконуваних файлів для цільових операційних систем.

Результат: Єдиний виконуваний файл та налаштування конфігурації.

3. Стадія. Тестування

Ця стадія присвячена перевірці функціональності, надійності, продуктивності та безпеки розробленого ПЗ.

Етап 3.1. Функціональне тестування.

Перевірка відповідності всіх реалізованих функцій вимогам Технічного Завдання. Тестування сценаріїв використання для офіціантів, барменів та адміністраторів.

Результат: Виявлення та документування функціональних дефектів, їх усунення.

Етап 3.2. Тестування інтеграції.

Перевірка коректності взаємодії між різними модулями системи та з базою

даних. Тестування перемикання між SQLite та PostgreSQL.

Результат: Підтвердження безшовної роботи всіх компонентів системи.

Етап 3.3. Тестування надійності та продуктивності.

Перевірка стабільності роботи системи протягом тривалого часу, вимірювання часу відгуку на дії користувача, часу запуску та споживання ресурсів. Тестування поведінки при некоректному вводі даних.

Результат: Підтвердження відповідності системи вимогам до надійності та продуктивності, оптимізація коду за потреби.

Етап 3.4. Тестування безпеки (базове).

Перевірка коректності роботи механізмів автентифікації та авторизації, базовий аудит доступу.

Результат: Підтвердження базового рівня безпеки даних.

4. Стадія. Впровадження та експлуатація

Фінальна стадія, яка включає розгортання системи в реальному середовищі та її подальше використання.

Етап 4.1. Розгортання системи.

Інсталяція програмного забезпечення на робочих станціях ресторану, налаштування бази даних та початкова ініціалізація даних.

Результат: Повністю розгорнута та готова до використання система.

Етап 4.2. Навчання персоналу.

Проведення тренінгів для офіціантів, барменів та адміністраторів щодо роботи з новою системою.

Результат: Персонал, що володіє необхідними навичками для ефективного використання ПЗ.

Етап 4.3. Дослідна експлуатація та моніторинг.

Запуск системи у тестовому або обмеженому режимі експлуатації, збір зворотного зв'язку від користувачів, моніторинг продуктивності та виявлення потенційних проблем.

Результат: Функціонуюча система в реальному середовищі, зібрані дані для подальших покращень.

Етап 4.4. Підтримка та обслуговування.

Забезпечення технічної підтримки користувачів, регулярне резервне копіювання даних, виправлення виявлених помилок та впровадження дрібних покращень.

Результат: Стабільна та безперебійна робота системи, оперативне реагування на проблеми.

Ці стадії та етапи забезпечують комплексний підхід до розробки ПЗ, гарантуючи якість та відповідність кінцевого продукту вимогам замовника, а також створюючи основу для його подальшого розвитку.

1.2.7 Порядок контролю та прийому

Якість програмного забезпечення є пріоритетом і контролюється на всіх етапах життєвого циклу розробки.

Контроль якості здійснюється на всіх стадіях життєвого циклу розробки програмного забезпечення, включаючи:

– Контроль на етапі аналізу та проєктування.

На цьому етапі перевіряється повнота та коректність зібраних вимог, архітектурних рішень, проєктів бази даних та інтерфейсів користувача. Важливим кроком є затвердження всієї документації, включаючи Технічне Завдання та архітектурний опис, замовником.

– Контроль на етапі реалізації (кодування).

Для забезпечення якості коду, його відповідності стандартам, ефективності та безпеки проводяться регулярні огляди коду (code reviews). Для ефективного відстеження всіх змін використовується системи контролю версій.

– Контроль на етапі тестування.

Цей етап передбачає виконання всіх видів тестування, які визначені в Технічному Завданні, таких як модульне, інтеграційне, системне та приймальне тестування. Важливою частиною є фіксація всіх виявлених дефектів та їхнє оперативне усунення.

Процедура Приймання Програмного Забезпечення

Приймання програмного забезпечення здійснюється поетапно, згідно з такою процедурою:

– Попереднє тестування розробником.

Команда розробки проводить повний цикл внутрішнього тестування. Метою є підтвердження відповідності системи всім функціональним та нефункціональним вимогам. Результати цього тестування обов'язково документуються.

– Демонстрація функціоналу.

Розробник демонструє ключовий функціонал програмного забезпечення представникам замовника. Основний акцент під час демонстрації робиться на тому, як реалізовані можливості відповідають вимогам, зазначеним у Технічному Завданні.

– Приймальне тестування (User Acceptance Testing – UAT).

Замовник, самостійно або із залученням своїх представників, проводить приймальне тестування системи. Це відбувається в умовах, максимально наближених до реальної експлуатації. Під час UAT перевіряється, наскільки програмне забезпечення відповідає бізнес-процесам піцерії та наскільки воно зручне для кінцевих користувачів. Замовник надає розробнику перелік усіх виявлених зауважень та дефектів.

– Усунення зауважень та повторне тестування.

Розробник усуває всі виявлені зауваження та дефекти. Після цього обов'язково проводиться повторне тестування виправлених компонентів, щоб переконатися у їх коректній роботі.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

2.1 Розробка загальної структури і варіантів використання програми

Програмне забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин» має чітко визначену структуру та набір варіантів використання, що забезпечують ефективну автоматизацію процесів та координацію роботи персоналу.

Система управління замовленнями призначена для автоматизації процесів прийому, обробки та відстеження замовлень, забезпечуючи координацію між офіціантами, барменами та адміністративним персоналом. Її основні функціональні домени охоплюють управління замовленнями (створення, редагування, відстеження статусу), управління меню (каталог напоїв, ціни, категорії, доступність), управління персоналом (автентифікація, авторизація, ролі користувачів) та систему безпеки (контроль доступу, аудит дій користувачів).

Архітектура системи базується на принципах модульності, що забезпечує незалежність функціональних блоків; розширюваності, що дозволяє легко додавати нові функції; переносимості, яка підтримує різні системи управління базами даних та операційні системи; а також зручності використання, що виражається в інтуїтивно зрозумілому інтерфейсі.

Система побудована за шаровою архітектурою, яка включає:

– Шар Представлення (Presentation Layer).

Відповідає за взаємодію з користувачем, відображення даних та обробку подій інтерфейсу. До нього входять головне вікно, вікна автентифікації, а також модулі для управління замовленнями, меню та персоналом.

– Шар Бізнес-Логіки (Business Logic Layer).

Реалізує бізнес-правила, валідує дані та координує операції. Включає менеджери автентифікації та конфігурації, бізнес-сутності та валідатори.

– Шар Доступу до Даних (Data Access Layer).

Забезпечує абстракцію доступу до бази даних, виконує CRUD-операції

(створення, читання, оновлення, видалення) та маппінг даних. Містить менеджери бази даних та репозиторії для замовлень, напоїв та співробітників.

– Шар Даних (Data Layer).

Відповідає за постійне збереження та отримання даних, а також за підтримку транзакцій та цілісності даних. Використовує PostgreSQL як базу даних за замовчуванням та SQLite як опціональну альтернативу.

Система розрізняє три основні ролі користувачів — Офіціант, Бармен та Адміністратор, кожна з яких має свій набір дозволів та сценаріїв використання.

Офіціант є основним користувачем системи, що працює у високому темпі та потребує швидкого доступу до функцій для ефективного обслуговування клієнтів. Його цілі включають швидке створення замовлень, відстеження їх статусу та мінімізацію помилок.

– Автентифікація в системі. Офіціант входить у систему, обираючи своє ім'я та вводячи пароль, отримуючи доступ до дозволених модулів.

– Створення замовлення. Офіціант ініціює створення нового замовлення, обирає офіціанта, вказує номер столу, додає позиції напоїв з каталогу та примітки, після чого система зберігає замовлення.

– Перегляд та фільтрація замовлень. Офіціант може переглядати список замовлень, застосовуючи фільтри за статусом, офіціантом або датою, але бачить лише власні замовлення.

– Редагування замовлення. Офіціант може коригувати власні замовлення, додаючи, видаляючи позиції або змінюючи їх кількість та примітки, за умови, що замовлення не має статусу "Видано" або "Скасовано".

Варіанти Використання для Бармена

Бармен є виконавцем замовлень, що потребує оперативної інформації та відповідає за якість продукції. Його цілі полягають в ефективній обробці черги замовлень, контролі доступності напоїв та оптимізації процесу приготування:

– Автентифікація в системі. Бармен входить у систему, отримуючи доступ до дозволених модулів.

– Створення замовлення. Бармен може створювати замовлення, ідентично офіціанту.

– Перегляд та фільтрація замовлень. Бармен переглядає всі замовлення в системі, застосовуючи фільтри та сортування.

– Редагування замовлення. Бармен може редагувати будь-які замовлення, які не мають статусу "Видано" або "Скасовано".

– Управління чергою замовлень. Бармен керує чергою активних замовлень, послідовно змінюючи їх статуси ("Нове" → "В процесі" → "Готово" → "Видано"), а також має можливість скасувати замовлення.

– Управління меню напоїв. Бармен може переглядати каталог напоїв, додавати нові позиції, редагувати існуючі та змінювати статус доступності напоїв.

– Контроль якості обслуговування. Бармен може аналізувати активні замовлення, переглядати метрики кількості замовлень в обробці, середній час виконання та завантаженість.

Адміністратор є керівником або менеджером закладу, що відповідає за налаштування системи, контроль роботи персоналу та управління каталогом продукції, а також забезпечення безпеки системи:

– Автентифікація в системі. Адміністратор входить у систему, отримуючи повний доступ до всіх модулів.

– Створення замовлення. Адміністратор може створювати замовлення, ідентично офіціанту.

– Перегляд та фільтрація замовлень. Адміністратор бачить всі замовлення в системі, використовуючи розширені можливості фільтрації.

– Редагування замовлення. Адміністратор може редагувати будь-які замовлення, які не мають статусу "Видано" або "Скасовано".

– Управління чергою замовлень. Адміністратор може керувати чергою активних замовлень та змінювати їх статуси.

– Управління меню напоїв. Адміністратор має повний доступ до каталогу напоїв, включаючи додавання, редагування та зміну статусу доступності.

– Контроль якості обслуговування. Адміністратор аналізує активні замовлення та метрики для оптимізації роботи.

– Управління персоналом. Адміністратор може додавати нових співробітників, редагувати їх інформацію, ролі та статус активності.

– Налаштування системи. Адміністратор може змінювати системні налаштування, включаючи тип бази даних та параметри підключення, з можливістю перезапуску програми для застосування змін.

2.2 Розробка системи класів

Вибір технологічного стеку, що складається з мови програмування C++ та фреймворку Qt, для розробки системи управління замовленнями бармена ресторану "Старий Млин" є стратегічно обґрунтованим рішенням, що відповідає вимогам проєкту щодо продуктивності, надійності, кросплатформенності та довгострокової підтримки.

1. Висока продуктивність та ефективність ресурсів

C++ відомий своєю високою продуктивністю та низьким споживанням системних ресурсів. Це є критично важливим для програмного забезпечення, що функціонуватиме в умовах інтенсивного використання в ресторані. Операції створення, редагування та фільтрації замовлень, а також швидкий доступ до каталогу напоїв, вимагають миттєвого відгуку системи. C++ дозволяє максимально оптимізувати код, що забезпечує швидкий запуск додатку (до 5 секунд) та відгук на дії користувача (до 1 секунди для більшості UI операцій) . Ефективне використання пам'яті (близько 50-100 МБ) та низьке завантаження CPU (1-2% в режимі очікування) є значними перевагами, особливо при роботі на обмежених апаратних ресурсах.[8]

2. Кросплатформенність

Qt — це потужний кросплатформенний фреймворк, що дозволяє розробляти додатки, які без змін вихідного коду можуть бути скомпільовані та працювати на різних операційних системах, включаючи Windows, Ubuntu (Linux) та macOS . Це

надає замовнику гнучкість у виборі обладнання та операційного середовища, мінімізуючи витрати на адаптацію ПЗ під різні платформи та спрощуючи розгортання в майбутньому, якщо знадобиться розширення або міграція на інші робочі станції.[9]

3. Надійність та стабільність

C++ у поєднанні з Qt забезпечує високий рівень контролю над системними ресурсами та пам'яттю, що сприяє створенню стабільних та надійних додатків. Qt, як зрілий та широко використовуваний фреймворк, має велику спільноту та ретельне тестування, що гарантує високу якість його компонентів. Це безпосередньо впливає на надійність функціонування системи, забезпечуючи її стабільну роботу без перезапуску протягом тривалого часу (тестовано до 8 годин безперервної роботи) та коректну обробку помилок без втрати даних.

4. Модульна архітектура та розширюваність

Використання C++ та Qt дозволяє легко реалізувати сучасні архітектурні патерни, такі як MVVM (Model-View-ViewModel) [10]. Це забезпечує чітке розділення відповідальності між компонентами системи, що робить код більш зрозумілим, легким для підтримки та подальшого розширення. Модульна організація проекту, реалізована за функціональністю, значно спрощує додавання нових функцій (наприклад, модуля звітності, розширеної аналітики, інтеграції з касовими системами) у майбутніх версіях, мінімізуючи ризики та вартість модифікацій.

5. Контроль та Гнучкість

C++ надає розробникам глибокий контроль над системними ресурсами та можливість оптимізувати продуктивність на низькому рівні. Qt доповнює це потужними інструментами для розробки графічного інтерфейсу та роботи з базами даних (зокрема, з PostgreSQL, яка є рекомендованою СУБД для виробничого середовища). Цей контроль і гнучкість дозволяють створювати високооптимізовані рішення, адаптовані до конкретних потреб замовника, а не покладатися на абстракції, які можуть призвести до надмірності або неефективності.

Загалом, вибір C++/Qt є оптимальним балансом між продуктивністю, гнучкістю розробки, можливістю масштабування та довгостроковою підтримкою, що відповідає як поточним потребам ресторану "Старий Млин", так і його майбутнім стратегічним планам розвитку.

2.3 Розробка методів

2.4 Розробка структури бази даних

База даних системи розроблена з фокусом на PostgreSQL як основну систему управління базами даних (СУБД), що забезпечує надійність, продуктивність та масштабованість. Кодування даних здійснюється у форматі UTF-8, що гарантує коректне відображення символів.

Структура бази даних включає чотири основні таблиці, спроектовані відповідно до третьої нормальної форми (3NF) з допустимими практичними денормалізаціями для оптимізації продуктивності. Цілісність даних підтримується за допомогою обмежень зовнішнього ключа (Foreign Key constraints) та перевіркових обмежень (Check constraints). Оптимізація запитів забезпечується автоматичними та додатковими індексами. [11]

Вибір PostgreSQL як основної СУБД для даного програмного забезпечення є обґрунтованим з кількох ключових причин. PostgreSQL є потужною, об'єктно-реляційною системою з відкритим вихідним кодом, яка відома своєю високою надійністю, стійкістю до відмов та суворим дотриманням стандартів SQL. Вона надає повну підтримку властивостей ACID (Atomicity, Consistency, Isolation, Durability), що є критично важливим для транзакційних систем, таких як управління замовленнями. На відміну від SQLite, яка є легкою файловою базою даних, що ідеально підходить для автономних, однокористувацьких додатків або як початкове рішення, PostgreSQL забезпечує нативну підтримку багатокористувацького доступу та мережевих підключень. Це дозволяє легко масштабувати систему для одночасної роботи кількох офіціантів, барменів та

адміністраторів, а також інтегрувати її з іншими сервісами у майбутньому. Розширена система індексації, оптимізатор запитів та можливість використання складних функцій та тригерів, написаних на PL/pgSQL, надають більшу гнучкість для реалізації складної бізнес-логіки та аналітичних запитів. Хоча SQLite використано як опція для спрощеного розгортання на початковому етапі розробки, стратегічний вибір PostgreSQL відповідає довгостроковим вимогам щодо масштабованості та надійності системи у ресторанному бізнесі.

Концептуальна модель даних представлена чотирма ключовими сутностями: Співробітники, Замовлення, Позиції Замовлень та Напої. Співробітники створюють Замовлення, які, у свою чергу, складаються з Позицій Замовлень, що посилаються на Напої.

1) Таблиця employees (Співробітники).

Таблиця employees призначена для зберігання інформації про персонал ресторану та їх облікових записів. Вона містить унікальний ідентифікатор id (первинний ключ, SERIAL PRIMARY KEY), ім'я first_name та прізвище last_name співробітника (обов'язкові поля типу VARCHAR(50)). Роль співробітника role (наприклад, 'waiter', 'bartender', 'admin') обмежується перевірковим обмеженням (CHECK). Поле is_active (BOOLEAN, за замовчуванням TRUE) вказує, чи є обліковий запис активним. created_at (TIMESTAMP, за замовчуванням CURRENT_TIMESTAMP) фіксує дату створення запису. Для цієї таблиці створюються індекси за роллю (idx_employees_role) та статусом активності (idx_employees_active) для оптимізації пошуку. Ім'я та прізвище є обов'язковими, роль може бути лише з допустимого списку, неактивні співробітники не можуть входити в систему, а при видаленні перевіряється наявність активних замовлень.

2) Таблиця drinks (Напої).

Таблиця drinks є каталогом всіх напоїв з їх характеристиками та цінами. Вона містить id (первинний ключ, SERIAL PRIMARY KEY), унікальну назву напою name (VARCHAR(100), UNIQUE, NOT NULL), ціну price (DECIMAL(8,2), NOT NULL, з обмеженням CHECK >= 0), категорію category (VARCHAR(50), NOT NULL) та статус активності is_active (BOOLEAN, за замовчуванням TRUE).

created_at (TIMESTAMP, за замовчуванням CURRENT_TIMESTAMP) вказує дату додавання напою. Індекси idx_drinks_category, idx_drinks_active та idx_drinks_name забезпечують швидкий пошук. Назва напою має бути унікальною, ціна не може бути від'ємною, неактивні напої не відображаються при створенні замовлень, а при редагуванні цін історія зберігається в позиціях замовлень.

3) Таблиця orders (Замовлення).

Таблиця orders зберігає основну інформацію про кожне замовлення та його поточний статус. Вона включає id (первинний ключ, SERIAL PRIMARY KEY), employee_id (INTEGER, NOT NULL, зовнішній ключ до employees), ім'я офіціанта employee_name (VARCHAR(100), NOT NULL – денормалізація для історичності), час створення created_at (TIMESTAMP, за замовчуванням CURRENT_TIMESTAMP), час завершення completed_at (TIMESTAMP, може бути NULL), статус замовлення status (VARCHAR(20), NOT NULL, за замовчуванням 'new', з перевірковим обмеженням на допустимі значення: 'new', 'in_progress', 'ready', 'served', 'cancelled'), номер столу table_number (INTEGER, за замовчуванням 0), загальну суму total_amount (DECIMAL(10,2), за замовчуванням 0) та примітки notes (VARCHAR(500)). Індекси idx_orders_status, idx_orders_employee_id, idx_orders_created_at та idx_orders_table оптимізують пошук. Кожне замовлення прив'язане до офіціанта, статус змінюється лише по визначеному ланцюжку, при завершенні автоматично встановлюється completed_at, а загальна сума розраховується з позицій.

4) Таблиця order_items (Позиції замовлень).

Таблиця order_items містить детальний склад кожного замовлення. Її поля: id (первинний ключ, SERIAL PRIMARY KEY), order_id (INTEGER, NOT NULL, зовнішній ключ до orders з ON DELETE CASCADE), drink_id (INTEGER, NOT NULL, зовнішній ключ до drinks), назва напою drink_name (VARCHAR(100), NOT NULL – денормалізація), кількість quantity (INTEGER, NOT NULL, CHECK >= 1), ціна за одиницю на момент замовлення unit_price (DECIMAL(8,2), NOT NULL), загальна вартість позиції total_price (DECIMAL(8,2), NOT NULL) та примітки до

позиції notes (VARCHAR(200)). Індекси idx_order_items_order_id та idx_order_items_drink_id оптимізують запити. Кожна позиція належить лише одному замовленню, кількість не може бути менше 1, ціни зберігаються на момент замовлення, а при видаленні замовлення всі пов'язані позиції видаляються автоматично.

Зв'язки між таблицями.

Зв'язки між таблицями встановлюються за принципом "один до багатьох" (One-to-Many). Один співробітник може створити багато замовлень. Це некаскадний зв'язок, що означає збереження замовлень при видаленні співробітника. Одне замовлення може містити багато позицій, і цей зв'язок є каскадним: при видаленні замовлення автоматично видаляються всі його позиції (ON DELETE CASCADE). Один напій може бути присутнім у багатьох позиціях замовлень. Цей зв'язок також некаскадний, що дозволяє зберігати історичні дані про замовлені напої навіть після можливого видалення напою з каталогу.

Для оптимізації продуктивності бази даних, окрім автоматично створюваних індексів на первинних та зовнішніх ключах, додано низку спеціалізованих індексів. Це включає idx_orders_status для швидкого пошуку за статусом замовлень, idx_orders_created_at для пошуку за датою створення у зворотному порядку, idx_drinks_category для швидкої фільтрації напоїв за категорією, а також idx_drinks_active та idx_employees_active для фільтрації активних записів. Додатково створено композитний індекс idx_orders_status_date для оптимізації звітів. Ці індекси є критичними для забезпечення високої швидкості обробки типових запитів, таких як отримання активних замовлень, деталей замовлення з позиціями, аналітичних запитів щодо популярності напоїв та продуктивності офіціантів.

Безпека та цілісність даних забезпечуються через жорсткі політики обмежень. Референційна цілісність підтримується зовнішніми ключами. Перевірка даних на відповідність бізнес-правилам реалізована за допомогою CHECK обмежень для ролей співробітників, цін, кількості позицій та статусів замовлень. Унікальність назв напоїв гарантується UNIQUE обмеженням. Для

автоматизації бізнес-логіки використовуються тригери PostgreSQL. Зокрема, тригер trigger_update_order_total автоматично оновлює загальну суму замовлення при зміні позицій, а trigger_set_completed_time встановлює час завершення замовлення при переході в статус 'served'.

Статистика та моніторинг бази даних.

Для моніторингу та подальшої оптимізації бази даних надаються типові розміри таблиць та частота виконання запитів. Рекомендації з оптимізації PostgreSQL включають налаштування параметрів shared_buffers, effective_cache_size, maintenance_work_mem та work_mem для підвищення продуктивності.

Міграції та версіонування.

Система підтримує версіонування схеми бази даних за допомогою спеціальної таблиці schema_version, що дозволяє відстежувати зміни та автоматично застосовувати міграції. Це забезпечує керований процес оновлення структури бази даних при подальшому розвитку програмного забезпечення, наприклад, при додаванні нових колонок або таблиць для розширення функціоналу, як планується для версій з таблицями інгредієнтів та рецептів.

Підсумуємо. Структура бази даних базується на чотирьох основних таблицях з логічними зв'язками, дотримується 3NF нормалізації з практичними денормалізаціями для забезпечення продуктивності. Вона повністю підтримує властивості ACID, оптимізована індексацією для частих запитів, використовує автоматичні тригери для підтримки цілісності та готова до розширення через систему міграцій. Ключовими перевагами такої архітектури є її простота, висока продуктивність та масштабованість, особливо завдяки вибору PostgreSQL як основної СУБД.

2.5 Проектування і опис інтерфейсу користувача

Проектування інтерфейсу користувача (ІК) для системи управління замовленнями "Старий Млин" було центральним елементом розробки,

спрямованим на створення інтуїтивно зрозумілого, функціонального та ефективного середовища для персоналу. Згідно з наданим профайлом проєкту Qt, система розробляється з використанням фреймворку Qt Widgets, що є ключовим показником архітектурного рішення для графічного інтерфейсу.

Інтерфейс користувача створюється з дотриманням принципів, що забезпечують зручність та продуктивність:

Вибір Qt Widgets гарантує високу продуктивність та нативне відчуття додатку на різних операційних системах. Це важливо для швидкого реагування системи на дії користувача в умовах інтенсивної роботи ресторану.

Дизайн ІК прагне мінімізувати криву навчання для персоналу. Елементи управління розташовані логічно, а робочі процеси відображають реальні сценарії використання.

Інтерфейс чітко розмежовує доступ до функціоналу відповідно до ролей користувачів, що мінімізує плутанину та забезпечує безпеку даних.

Компоненти інтерфейсу користувача складається з кількох ключових компонентів:

– main.cpp / mainwindow.cpp (mainwindow.h) – це є ядром графічного інтерфейсу. MainWindow є основним вікном програми, яке, ймовірно, містить табову навігацію або інші елементи для перемикання між основними функціональними модулями.

– auth/logindialog.cpp (auth/logindialog.h) – вікно автентифікації (LoginDialog) є першим, що бачить користувач при запуску програми. Воно відповідає за введення облікових даних та перевірку прав доступу.

– widgets/ordermanagement.cpp (widgets/ordermanagement.h): – модуль управління замовленнями (OrderManagement) буде містити інтерфейс для прийому нових замовлень, перегляду активних замовлень (можливо, у вигляді таблиці), їх фільтрації та зміни статусу.

– widgets/menumanagement.cpp (widgets/menumanagement.h) – модуль управління меню (MenuManagement) надає інтерфейс для перегляду та редагування каталогу напоїв. Тут будуть елементи для додавання нових напоїв,

зміни цін, категорій та статусу доступності.

– widgets/employeemanagement.cpp (widgets/employeemanagement.h) – модуль управління персоналом (EmployeeManagement) буде доступний адміністраторам для керування обліковими записами співробітників, зміни їхніх ролей та статусу.

2.6 Опис файлової структури програми

Файлова структура системи управління замовленнями "Старий Млин" організована логічно, що забезпечує зручність розробки, підтримки та масштабування проєкту. Вона відображає модульний підхід, де кожен компонент системи має своє чітко визначене місце.

У кореневому каталозі проєкту розташовані основні файли конфігурації та запуску:

– BarOrderSystem.pro – це головний профайл Qt проєкту, який визначає структуру збірки, включає всі джерельні та заголовкові файли, налаштування компілятора, бібліотеки, версії, інформацію про додаток та правила для різних операційних систем. Лістинг коду наведено у додатку А.

– main.cpp – точка входу в програму, відповідальна за ініціалізацію додатку та головного вікна, лістинг коду наведено у додатку Б.

– mainwindow.h / mainwindow.cpp – файли, що містять код для головного вікна додатку та його базового функціоналу. Лістинги даних файлів наведено у додатках В і Г відповідно

Далі файли групуються за їхньою функціональною приналежністю до певних модулів або логічних шарів системи:

– widgets/ – каталог містить файли, що відповідають за графічний інтерфейс користувача (UI) та логіку представлення (Views та частково ViewModels). Тут знаходяться:

– ordermanagement.cpp / ordermanagement.h – код для модуля управління замовленнями.

– menumanagement.cpp / menumanagement.h – код для модуля управління меню напоїв.

– employeemanagement.cpp / employeemanagement.h – Код для модуля управління персоналом.

– models/ – каталог містить файли моделей даних, що представляють бізнес-сутності системи, такі як:

– employee.h – модель даних для співробітників.

– drink.h – модель даних для напоїв.

– order.h – модель даних для замовлень.

– database/ – в каталозі розташовані файли, що забезпечують доступ до бази даних (DAL):

– databasemanager.cpp / databasemanager.h – менеджер підключення до бази даних та виконання загальних запитів.

– repositories.cpp / repositories.h – реалізації репозиторіїв для взаємодії з таблицями orders, drinks, employees та order_items.

– config/ – цей каталог містить файли для управління конфігурацією системи: configmanager.cpp / configmanager.h – менеджер читання та запису налаштувань, наприклад, параметрів підключення до БД.

– auth/ – в каталозі знаходяться файли, що відповідають за автентифікацію та авторизацію користувачів:

– authmanager.cpp / authmanager.h – менеджер управління сесіями користувачів та їхніми правами доступу.

– logindialog.cpp / logindialog.h – вікно входу в систему.

2.7 Тестування програми

Процес тестування програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин» був виконаний у кілька етапів, щоб гарантувати високу якість, надійність та відповідність системи всім встановленим

вимогам. Тестування охопило функціональні та нефункціональні аспекти, дозволивши виявити та усунути дефекти перед впровадженням системи в експлуатацію.

Функціональне тестування було спрямоване на перевірку коректності реалізації всіх заявлених можливостей системи, відповідно до її призначення та ролей користувачів.

– Тестування модуля автентифікації.

Перевірявся процес входу в систему для кожної ролі (офіціант, бармен, адміністратор), включаючи сценарії успішного та неуспішного входу, а також реакцію на неактивні облікові записи.

– Тестування модуля замовлень.

Охоплювало повний цикл роботи із замовленнями. Проведено тестування створення нових замовлень, їх редагування (додавання/видалення позицій, зміна кількості), відстеження та зміна статусів замовлень (Нове → В процесі → Готово → Видано), а також функції перегляду та фільтрації замовлень за різними критеріями.

– Тестування модуля меню.

Перевірялися всі CRUD-операції (створення, читання, оновлення, видалення) для напоїв та категорій. Особлива увага приділялась коректності відображення цін, категорій та статусу доступності напоїв.

– Тестування модуля персоналу.

Перевірено функціонал управління співробітниками, доступний лише адміністратору. Це включало додавання нових облікових записів, їх редагування (зміна імені, ролі) та деактивацію.

– Тестування прав доступу.

Проведено ретельну перевірку розмежування прав доступу для кожної ролі користувача, щоб переконатися, що офіціанти, бармени та адміністратори мають доступ виключно до дозволеного функціоналу.

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція з інсталяції програмного забезпечення

Системні вимоги.

Мінімальні вимоги:

– ОС: Windows 10 (64-bit) або новіша;
– процесор: Intel Core i3 або AMD еквівалент;
– оперативна пам'ять: 4 ГБ RAM;
– вільне місце: 500 МБ на диску;
– розширення екрану: 1024x768 пікселів;
– мережа: Ethernet або Wi-Fi.

Рекомендовані вимоги:

– ОС: Windows 11 (64-bit);
– процесор: Intel Core i5 або AMD Ryzen 5;
– оперативна пам'ять: 8 ГБ RAM;
– диск: SSD накопичувач;
– розширення екрану: 1920x1080 пікселів або вище.

Інсталяція програми:

Крок 1: Завантаження файлів:

– отримайте інсталяційний пакет BarOrderSystem-Setup.exe
– переконайтеся в цілісності файлу (MD5: xxx)
– запустіть файл від імені адміністратора

Крок 2: Запуск інсталятора:

– клікніть правою кнопкою на BarOrderSystem-Setup.exe
– виберіть "Запустити від імені адміністратора"
– підтвердіть запуск в діалозі User Account Control (UAC)

Крок 3: Процес інсталяції:

Вітальний екран:

– натисніть "Далі" для продовження.

Ліцензійна угода:

– прочитайте умови ліцензії;

– позначте "Я приймаю умови ліцензійної угоди";

– натисніть "Далі".

Вибір папки інсталяції:

– за замовчуванням: «C:\Program Files\BarOrderSystem\»;

– або виберіть іншу папку натиснувши "Огляд";

– натисніть "Далі".

Вибір компонентів:

– Основна програма (обов'язково);

– Демонстраційні дані (рекомендовано);

– Ярлики робочого столу;

– Документація;

– Натисніть "Далі".

Готовність до інсталяції:

– перевірте налаштування;

– натисніть "Встановити".

Процес копіювання файлів:

– зачекайте завершення (2-3 хвилини);

– не закривайте інсталятор.

Завершення інсталяції:

– позначте "Запустити програму";

– натисніть "Готово".

3.2 Інструкція з використання тестових наборів

При розробці програмного забезпечення на C++/Qt для забезпечення

високої якості продукту доцільно використовувати різноманітні тестові набори. Їхнє застосування дозволяє верифікувати функціональність, продуктивність, надійність та зручність системи на різних рівнях абстракції.

Типи тестових наборів та їх застосування:

1. Набори для Модульного Тестування (Unit Tests)

Модульні тести фокусуються на перевірці окремих, ізольованих компонентів або функцій програмного коду (методів класів, функцій). Підтвердження коректності роботи найдрібніших логічних одиниць. Раннє виявлення дефектів.

Склад наборів:

Вхідні дані: Набори значень для параметрів функції/методу, що охоплюють типові випадки, граничні умови (мінімальні/максимальні значення, нульові або порожні значення), та некоректні дані для перевірки обробки помилок.

Очікувані результати: Заздалегідь визначені вихідні значення або очікувана поведінка (наприклад, генерація виключення) для кожного набору вхідних даних.

Ізольовані оточення (Mocks/Stubs): Спеціально створені об'єкти, що імітують поведінку залежностей (наприклад, базу даних, мережеві сервіси, системні файли), щоб ізолювати тестування конкретного модуля. Це дозволяє зосередитися на логіці тестованого компонента, незважаючи на зовнішні фактори.

Приклади для C++/Qt: Тестові сценарії для OrderRepository::saveOrder(), Drink::calculatePrice(), AuthManager::authenticateUser(), які перевіряють логіку роботи з даними, математичні обчислення або алгоритми валідації. Застосовується Qt Test Framework.

2. Набори для Інтеграційного Тестування (Integration Tests)

Інтеграційні тести перевіряють взаємодію між двома або більше модулями, підсистемами чи компонентами, що працюють разом.

Призначення: Виявлення дефектів, що виникають на стику компонентів, при передачі даних або використанні спільних ресурсів.

Послідовності операцій: Сценарії, що описують послідовність викликів функцій або методів різних модулів, які разом реалізують певну функціональність

(наприклад, офіціант створює замовлення, система зберігає його, бармен змінює статус).

Дані для взаємодії: Набори даних, що передаються між модулями, включаючи коректні, частково коректні та некоректні дані для перевірки обробки помилок на межах інтерфейсів.

Імітація зовнішніх систем: У деяких випадках, якщо інтеграція відбувається із зовнішніми системами (наприклад, платіжними шлюзами або іншими API), можуть використовуватися тестові середовища цих систем або їх імітації.

Приклади для C++/Qt: Тести для взаємодії OrderManagement (Presentation Layer) з OrderRepository (Data Access Layer) та DatabaseManager. Перевірка, як створення замовлення через UI відображається у базі даних, і як зміни статусу замовлення впливають на відображення у черзі бармена.

3. Набори для Системного Тестування (System Tests)

Системні тести оцінюють поведінку всієї інтегрованої системи у відповідності до функціональних та нефункціональних вимог.

Призначення: Валідація системи в цілому в умовах, максимально наближених до реальної експлуатації.

Повноцінні сценарії використання (Use Cases): Описи повних бізнес-процесів, що виконуються в системі (наприклад, повний цикл від прийому замовлення офіціантом до його виконання барменом та відмітки як "видано").

Дані реального обсягу: Використання великих обсягів даних (каталоги напоїв, кількість замовлень) для перевірки продуктивності та стабільності.

Тестові середовища: Розгортання системи в середовищі, ідентичному або максимально схожому на виробниче, включаючи реальну базу даних (PostgreSQL), мережеві налаштування тощо.

Стрес-тести: Набори даних та сценарії, що імітують пікові навантаження (наприклад, одночасне створення великої кількості замовлень), для перевірки стійкості та здатності системи витримувати високе навантаження.

Приклади для C++/Qt: Тестові сценарії, що імітують одночасну роботу кількох офіціантів та барменів, перевіряють швидкість оновлення даних у

реальному часі, генерацію звітів за великий період.

4. Набори для Приймального Тестування (User Acceptance Testing – UAT)

Приймальне тестування виконується замовником або кінцевими користувачами.

Призначення: Підтвердження того, що система відповідає бізнес-потребам замовника та зручна для використання кінцевими користувачами в реальних умовах.

Кейси користувача (User Cases): Сценарії, що відтворюють щоденні операції користувачів, написані зрозумілою для бізнесу мовою.

Реальні дані: Використання фактичних даних або їх максимально точних аналогів для перевірки коректності бізнес-логіки.

Сценарії з помилками: Навмисне введення некоректних даних або виконання дій, що викликають помилки, для перевірки коректності повідомлень користувачеві та поведінки системи.

Приклади для C++/Qt: Офіціант перевіряє швидкість створення замовлення та додавання позицій; бармен оцінює зручність роботи з чергою замовлень; адміністратор перевіряє легкість додавання нових співробітників або зміни цін у меню.

Використання цих різнорівневих тестових наборів у поєднанні з відповідними інструментами (наприклад, Qt Test Framework, власні тестові утиліти) дозволяє систематично підходити до контролю якості та забезпечувати високу надійність програмного забезпечення.

3.3 Інструкція з експлуатації програмного забезпечення

Інструкція з Експлуатації Програмного Забезпечення "Система Управління Замовленнями Бармена" для Ресторану "Старий Млин"

Ця інструкція є посібником для користувачів та адміністраторів програмного забезпечення "Система управління замовленнями бармена" (далі – Система) та містить детальний опис функціоналу та порядку роботи з нею.

Дотримання викладених правил забезпечить ефективну та безперебійну роботу Системи в ресторані "Старий Млин".

1. Загальні Відомості про Систему

Система призначена для автоматизації базових процесів прийому, обробки та виконання замовлень напоїв у барі. Вона забезпечує електронний облік замовлень, централізоване управління каталогом напоїв та контроль доступу персоналу.

2. Вхід до Системи

Щоб розпочати роботу, запустіть виконуваний файл Системи BarOrderSystem.exe, якщо ви користуєтеся Windows, або відповідний файл для вашої операційної системи. Після запуску на екрані з'явиться вікно автентифікації. Вам необхідно обрати своє ім'я зі списку доступних співробітників, а потім ввести свій пароль. Зауважте, для демонстраційної версії 1.0 пароль відповідає прізвищу співробітника, однак у виробничому середовищі ці паролі будуть змінені. Щойно дані будуть успішно перевірені, система надасть вам доступ до відповідних модулів згідно з вашою роллю.

3. Ролі та Права Доступу

Система передбачає розмежування прав доступу для різних категорій персоналу:

Офіціант має доступ лише до модуля "Замовлення" для створення та перегляду власних замовлень.

Бармен має доступ до модулів "Замовлення" (для управління чергою та статусами замовлень) та "Меню" (для перегляду каталогу напоїв).

Адміністратор має повний доступ до всіх модулів, включаючи "Замовлення", "Меню" та "Персонал".

4. Робота з Модулями Системи.

Основні функціональні можливості системи реалізовані у вигляді окремих модулів, доступ до яких залежить від ролі користувача.

4.1. Модуль "Замовлення" (для Офіціанта та Бармена)

Цей модуль є центральним для управління замовленнями.

Щоб переглянути замовлення, увійдіть до модуля "Замовлення". Тут ви побачите список активних замовлень. Ви можете звузити цей список, використовуючи фільтри за статусом, наприклад "Нове", "В процесі" або "Готово", а також за офіціантом або часом створення. Щоб оновити список і отримати найактуальніші дані, просто натисніть кнопку "Оновити".

Для створення нового замовлення натисніть кнопку "Нове замовлення". У діалоговому вікні, що з'явиться, оберіть офіціанта зі списку та вкажіть номер столу. Щоб додати напої, натисніть кнопку "Додати напій", оберіть потрібну позицію з каталогу, вкажіть кількість та, за потреби, додайте примітки до напою. Повторіть ці дії для всіх позицій замовлення. Система автоматично розрахує загальну суму. Щоб відправити замовлення, натисніть "Зберегти".

Для редагування замовлення оберіть незавершене замовлення зі списку та натисніть кнопку "Редагувати". Ви можете додавати, видаляти або змінювати кількість позицій, а також коригувати примітки. Після внесення всіх змін натисніть "Зберегти". Зверніть увагу, редагування можливе лише для замовлень зі статусами "Нове" або "В процесі".

Зміна статусу замовлення доступна бармену. Бармен може змінювати статус замовлень у черзі. Для цього оберіть замовлення зі списку та натисніть кнопку "Змінити статус". Статуси змінюються послідовно: Нове переходить у В процесі, потім у Готово, а потім у Видано. Коли статус встановлюється на "Видано", система автоматично фіксує час завершення замовлення.

Модуль "Меню" (для Бармена та Адміністратора).

Цей модуль призначений для управління каталогом напоїв.

Щоб переглянути каталог напоїв, перейдіть до вкладки "Меню". Тут ви побачите список усіх доступних напоїв. Ви можете легко знайти потрібні позиції, використовуючи фільтри за категоріями та поле пошуку за назвою.

Для додавання нового напою натисніть кнопку "Додати напій". Уведіть назву, ціну, категорію та встановіть початковий статус (активний чи неактивний), потім натисніть "Зберегти".

Щоб редагувати напій, оберіть його зі списку та натисніть кнопку

"Редагувати". Ви можете змінити назву, ціну, категорію або переключити статус доступності. Натисніть "Зберегти" для застосування змін. Важливо пам'ятати, що неактивні напої автоматично приховуються в інтерфейсі створення замовлень.

Модуль "Персонал" (для Адміністратора)

Цей модуль доступний виключно адміністратору для управління обліковими записами співробітників.

Щоб переглянути список співробітників, відкрийте вкладку "Персонал". Тут відображається перелік усіх зареєстрованих співробітників.

Для додавання нового співробітника натисніть кнопку "Додати співробітника". Введіть ім'я, прізвище та оберіть роль для нового облікового запису. Встановіть початковий статус активності, потім натисніть "Зберегти".

Щоб редагувати інформацію про співробітника, оберіть його зі списку та натисніть кнопку "Редагувати". Ви можете змінити ім'я, прізвище, роль або переключити статус активності. Натисніть "Зберегти". Зверніть увагу, деактивовані облікові записи не можуть використовуватися для входу в систему.

Вихід з Системи.

Для завершення роботи з Системою, оберіть меню "Файл" у головному вікні. Потім натисніть "Вийти" (або "Logout"), щоб повернутися до екрану входу, або "Вихід" (чи "Exit"), щоб повністю закрити додаток.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини кваліфікаційної роботи є виконання здійснення економічних розрахунків, спрямованих на визначення економічної ефективності розробки програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин».

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

Для визначення загальної тривалості проведення НДР доцільно дані витрат часу по окремих операціях технологічного процесу звести у таблицю 4.1.

Таблиця 4.1 - Середній час виконання робіт по обслуговуванню та стадії (операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1.	Підготовка технічного завдання	кер. проекту	4
2.	Розробка системи класів	програміст	4
3.	Розробка бази даних сайту	програміст	8
4.	Реалізація методів класів	програміст	40
5.	Розробка інтерфейсу користувача	програміст	24
6.	Тестування програмного забезпечення	тестувальник	20
7.	Підготовка документації	кер. проекту	4
8.	Здача програми	програміст	4
		кер. проекту	4
	Р а з о м		112

Сумарний час виконання операцій технічного процесу розробки програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин» становить 112 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Оплата праці - грошовий вираз вартості і ціни робочої сили, який виступає у формі будь-якого заробітку, виплаченого керівником підприємства найманому працівникові за виконану роботу.

Заробітна плата працівника залежить від кінцевих результатів його роботи, регулюється податками і максимальними розмірами не обмежується.

Основна заробітна плата розраховується за формулою 4.1:

$$З_{осн.} = Т_с \cdot К_г, \quad (4.1)$$

де $Т_с$ – тарифна ставка, грн.; $К_г$ – кількість відпрацьованих годин.

Рекомендовані тарифні ставки: керівник проєкту – 350 грн./год., програміст – 250 грн./год., тестувальник – 150 грн./год.

Отже основна заробітна плата становитиме:

– керівник проєкту: $З_{осн1} = 400 \cdot 12 = 4800$ грн.;

– програміст: $З_{осн2} = 220 \cdot 80 = 17600$ грн.;

– тестувальник: $З_{осн3} = 120 \cdot 20 = 2400$ грн.

Сумарна основна заробітна плата становить:

$$З_{осн} = 4800 + 17600 + 2400 = 24800 \text{ грн.}$$

Додаткова заробітна плата становить 10 - 15% від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.}, \quad (4.2)$$

де $К_{допл.}$ – коефіцієнт додаткових виплат працівникам (приймаємо 15%).

Отже додаткова заробітна плата по категоріях працівників становить:

– керівник проєкту: $З_{дод1} = 4800 \cdot 0,15 = 720$ грн.;

– програміст: $З_{дод2} = 17600 \cdot 0,15 = 2640$ грн.;

– тестувальник: $З_{дод3} = 2400 \cdot 0,15 = 360$ грн.

Сумарна додаткова заробітна плата становить:

$$З_{дод} = 720 + 2640 + 360 = 3720 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($В_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.} \quad (4.3)$$

$$В_{о.п} = 24800 + 3720 = 28520 \text{ грн.}$$

Необхідно визначити відрахування на соціальні заходи:

– фонд страхування на випадок безробіття – 1,6 %;

– фонд по тимчасовій втраті працездатності – 1,4 %;

– пенсійний фонд – 33,2 %;

– внески на страхування від нещасного випадку на виробництві та професійного захворювання - 1,4%.

Загальна сума зазначених відрахувань становить 37,6 %.

Отже, сума нарахувань на заробітну плату буде становити згідно формули 4.4:

$$В_{с.з.} = ФОП \cdot 0,376 \quad (4.4)$$

де ФОП – фонд оплати праці, грн.

$$В_{с.з.} = 28520 \cdot 0,376 = 10723,52 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці зведемо у таблицю 4.2.

Таблиця 4.2 - Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додат-кова заробітна плата, грн.	Нарахув. на ФОП, грн.	Нарахування на оплату праці, грн. 6=3+4+5
		Тарифна ставка, грн.	К-сть від-працьов. год.	Фактично нарах. з/пл., грн.			
1	Кер.проекту	400	12	4800	720	-	-
2	Програміст	220	80	17600	2640	-	-
3	Тестуваль-ник	120	20	2400	360	-	-
Разом				24800	3720	10723,52	39243,52

Отже, загальні витрати на оплату праці становлять 39243,52 грн.

4.3 Розрахунок матеріальних витрат

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни (формула 4.5):

$$M_{Bi} = q_i \cdot p_i \,, \tag{4.5}$$

де q_i – кількість витраченого матеріалу і-го виду; p_i – ціна матеріалу і-го виду.

Звідси, загальні матеріальні витрати можна визначити за формулою 4.6:

$$З_{м.в.} = \sum M_{B\,i} \,, \tag{4.6}$$

Проведені розрахунки занесемо у таблицю 4.3.

Таблиця 4.3 - Зведені розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Факт. витрачено матеріалів	Ціна 1-ці, грн.	Загальна сума витрат, грн.
1	Друк листів А3	листів	6	10	60
2	Друк листів А4	листів	100	2	200
3	DVD-диск	шт.	1	20	20
Р а з о м					280

Загальна сума матеріальних витрат на розробку програмного забезпечення становить 275 грн.

4.4 Розрахунок витрат на електроенергію

Затрати на електроенергію 1-ці обладнання визначаються за формулою 4.7:

$$З_е = W \cdot T \cdot S, \quad (4.7)$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії (приймаємо 7 грн).

Для розробки інтернет-магазину використовувався один персональний комп’ютер. Витрати на електроенергію обчислимо взявши за основу, що час роботи обладнання обчислюється в залежності від виконуваних робіт (згідно таблиці 4.1) при вартості 1 кВт електроенергії – 7 грн. та потужністю у 0,5 кВт.

$$З_{ве} = 0{,}5 \cdot 112 \cdot 7 = 392 \text{ грн.}$$

Витрати на електроенергію становлять 392 грн.

4.5 Розрахунок суми амортизаційних відрахувань

Комп’ютери та оргтехніка належать до четвертої групи основних фондів.

Мінімально допустимі строки їх використання 2 роки. Для визначення амортизаційних відрахувань застосовуємо формулу 4.8:

$$А = \frac{Б_В \cdot Н_А}{100\%} \cdot T, \quad (4.8)$$

де А – амортизаційні відрахування за звітний період, грн.; $Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.; Т - кількість годин роботи обладнання, год.; $Н_А$ – норма амортизації .

Оскільки для написання програми та її тестування використовується один ПК вартістю 32500 грн., тоді сума амортизаційних відрахувань становить:

$$А = \frac{32500 \cdot 0{,}04}{150} \cdot 112 = 970{,}67 \text{ грн.}$$

4.6 Обчислення накладних витрат

Накладні витрати - це витрати, не пов'язані безпосередньо з технологічним процесом виготовлення продукції, а утворюються під впливом певних умов роботи по організації, управлінню та обслуговуванню виробництва.

В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20 – 60 % від суми основної та додаткової заробітної плати працівників, обчислюються за формулою 4.9:

$$Нв = Воп. \cdot 0{,}2 \ldots 0{,}6, \quad (4.9)$$

де $Н_В$ – накладні витрати.

$$Н_В = 28520 \cdot 0{,}2 = 5704 \text{ грн.}$$

4.7 Складання кошторису витрат та визначення собівартості НДР

Кошторис витрат являє собою зведений план усіх витрат підприємства на

майбутній період виробничо-фінансової діяльності.

Результати проведених вище розрахунків зведемо у таблиці 4.4.

Таблиця 4.4 - Кошторис витрат

Зміст витрат	Сума, грн.	В % до загальної суми
Витрати на оплату праці	28520,00	61,22
Відрахування на соціальні заходи	10723,52	23,02
Матеріальні витрати	280,00	0,60
Витрати на електроенергію	392,00	0,84
Амортизаційні відрахування	970,67	2,08
Накладні витрати	5704,00	12,24
Собівартість	46590,19	100

Собівартість ($С_В$) НДР знайдемо за формулою 4.10:

$$С_В = В_{О.П.} + В_{С.З.} + З_М + З_е + А + Н_в \quad (4.10)$$

Отже, собівартість дорівнює: $С_В$ = 46590,19 грн.

4.8 Розрахунок ціни НДР

Ціну НДР можна визначити за формулою 4.11:

$$Ц = Св \cdot (1+Р_{рен}) \cdot (1+ПДВ), \quad (4.11)$$

де Св – собівартість розробки; $Р_{рен.}$ – рівень рентабельності, (30 %); ПДВ – ставка податку на додану вартість, (20 %).

$$Ц = 46590,19 \cdot (1 + 0,3) \cdot (1 + 0,2) = 72680,70 \text{ грн.}$$

4.9 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Прибуток розраховується за формулою 4.12:

$$П = Ц – С_В \quad (4.12)$$

$$П = 72680,70 – 46590,19 = 26090,51 \text{ грн.}$$

Економічна ефективність (Ер) полягає у відношенні результату виробництва до затрачених ресурсів і розраховується за формулою 4.13:

$$Е_р = П / С_в \quad (4.13) \quad (4.13)$$

де П– прибуток; $С_в$ – собівартість.

$$Ер = 26090,51 / 46590,19 = 0,56$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень $Т_р$ (формула 4.14)

$$Т_р = 1 / Ер \quad (4.14)$$

Допустимим вважається термін окупності до 5 років. В даному випадку:

$$Т_р=1/0,56 = 1,8.$$

Всі дані внесемо в зведену таблицю 4.5 техніко-економічних показників

Таблиця 4.5 - Техніко-економічні показники розробки програмного забезпечення

№ п/п	Показник	Значення
1.	Собівартість, грн.	46590,19
2.	Плановий прибуток, грн.	26090,51
3.	Ціна, грн.	72680,70
4.	Економічна ефективність, грн.	0,56
5.	Термін окупності, рік	1,8

Загальна вартість розробки програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин» становить 91622,37 грн. Зважаючи на високі показники економічної ефективності - 0,56, кошти, вкладені в розробку програмного забезпечення окупляться за 1,8 року.

5 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

5.1 Розрахунок системи штучного освітлення для приміщення, де здійснюється розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»

Метою цього розділу є розрахунок системи штучного освітлення для приміщення, де здійснюється розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин». При виконанні при розрахунку освітлення необхідно використовувати ДБН В.2.5-28:2018 [12] для загальних вимог та ДСТУ EN 12464-1:2016 [13] для специфічних вимог до освітлення внутрішніх робочих місць, особливо тих, що обладнані візуальними дисплейними терміналами. Ці нормативні документи визначають вимоги до нормованої освітленості, якісних показників освітлення (рівномірність, показник осліпленості, коефіцієнт пульсації), а також до вибору типів світильників та їх розташування. Правильний розрахунок дозволить створити оптимальні світлові умови, які сприятимуть збереженню зору працівників, підвищенню їхньої працездатності та забезпеченню відповідності робочих місць нормам охорони праці.

Ефективне та належне освітлення є одним із ключових чинників, що визначають комфорт, продуктивність і, найголовніше, безпеку праці у будь-якому виробничому чи офісному приміщенні. Особливого значення це набуває в приміщеннях, де здійснюється розробка програмного забезпечення, оскільки робота програміста пов'язана з високим зоровим навантаженням і тривалим перебуванням за монітором. Недостатнє, надмірне або неправильно організоване освітлення може призвести до швидкої втоми очей, головного болю, зниження концентрації уваги та загального самопочуття, що, в свою чергу, негативно позначається на якості виконуваної роботи та може спричинити розвиток професійних захворювань.

Залежно від джерела світла виробниче освітлення може бути: природним,

що створюється прямими сонячними променями та розсіяним світлом небосхилу; штучним, що створюється електричними джерелами світла та суміщеним, при якому недостатнє за нормами природне освітлення доповнюється штучним.

Штучне освітлення може бути загальним та комбінованим. Загальним називають освітлення, при якому світильники розміщуються у верхній зоні приміщення (не нижче 2,5м над підлогою) рівномірно (загальне рівномірне освітлення) або з врахуванням розташування робочих місць (загальне локалізоване освітлення). Комбіноване освітлення складається із загального та місцевого. Його доцільно застосовувати при роботах високої точності, а також, якщо необхідно створити певний або змінний, в процесі роботи, напрямок світла. Місцеве освітлення створюється світильниками, що концентрують світловий потік безпосередньо на робочих місцях. Застосування лише місцевого освітлення не допускається з огляду на небезпеку виробничого травматизму та професійних захворювань.

За функціональним призначенням штучне освітлення поділяється на робоче, аварійне, евакуаційне, охоронне, чергове.

Штучне освітлення передбачається у всіх виробничих та побутових приміщеннях, де недостатньо природного світла, а також для освітлення приміщень в темний період доби. При організації штучного освітлення необхідно забезпечити сприятливі гігієнічні умови для зорової роботи і одночасно враховувати економічні показники.

Найменша освітленість робочих поверхонь у виробничих приміщеннях визначається, в основному, характеристикою зорової роботи.

В якості джерел штучного освітлення широко використовують лампи розжарювання та газорозрядні лампи.

Лампи розжарювання відносяться до теплових джерел світла. Під дією електричного струму нитка розжарювання (вольфрамовий дріт) нагрівається до високої температури і випромінює потік променевої енергії. Ці лампи характеризуються простотою конструкції та виготовлення, відносно низькою

вартістю, зручністю експлуатації, широким діапазоном напруг та потужностей.

Поряд з перевагами їм притаманні і суттєві недоліки - велика яскравість (засліплююча дія); низька світлова віддача (7—20 лм/Вт); відносно малий термін експлуатації (2,5 тис. год); переважання жовто-червоних променів в порівняні з природним світлом; висока температура нагрівання до 140°С і вище що робить їх пожежонебезпечними.

Лампи розжарювання використовують, як правило, для місцевого освітлення, а також освітлення приміщень з тимчасовим перебуванням людей.

Газорозрядні лампи внаслідок електричного розряду в середовищі інертних газів і парів металу та явища люмінесценції випромінюють світа оптичного діапазону спектру.

Основною перевагою газорозрядних ламп є їх економічність.

Світлова віддача цих ламп становить 40—100 лм/Вт, що в 3 рази перевищує світлову віддачу ламп розжарювання. Термі експлуатації — до 10 тис. год., а температура нагрівання (люмінесцентні) — 30-60 °С. Окрім того, газорозрядні ламп забезпечують світловий потік практично будь-якого спектра, шляхом підбирання відповідним чином інертних газів, парів металу, люмінофору. Так, за спектральним складом видимого світла розрізняють люмінесцентні лампи: денного світла (ЛД), денного світла з покращеною передачею кольорів (ЛДЦ), холодного білого (ЛХБ), теплого білого (ЛТБ) та білого (ЛБ) кольорів.

Основним недоліком газорозрядних ламп є пульсація світлового потоку, що може зумовити виникнення стробоскопічного ефекту, котрі полягає у спотворенні зорового сприйняття об'єктів, що рухаються обертаються. До недоліків цих ламп можна віднести також складність схеми включення, шум дроселів, значний час між включенням та запалюванням ламп, відносна дороговизна.

При проектуванні штучного освітлення необхідно вирішити наступне: вибрати систему освітлення, тип джерела світла, тип світильників, визначити розташування світлових приладів, виконати розрахунки штучного освітлення та визначити потужності світильників та ламп.

Розраховуємо систему загального рівномірного освітлення

люмінесцентними лампами для офісного приміщення, в якому виконуються зорові роботи дуже високої точності ($II_г$), мінімальне освітлення якого становить Е = 300лк. Як світлові пристрої приймаємо світильники типу ЛПО01 (з двома лампами). Як джерело світла при штучному освітленні застосовуватимуться люмінесцентні лампи типу ЛБ-40.

Розрахунок виконаємо методом коефіцієнта використання світлового потоку.

Загальне освітлення має бути виконане у вигляді суцільних або переривчастих ліній світильників, що розміщуються збоку від робочих місць (переважно зліва) паралельно лінії зору працівників.

Розміри офісного приміщення, де встановлено 10 ПК та 2 мережевих принтери: довжина а=14,90м. (прийнята усереднено для спрощення розрахунків), ширина b=7,70м, висота Н=3м. Приміщення має такі показники: коефіцієнт відбиття $\rho_{стелі}$=70%, $\rho_{стін}$=50%.

Висота робочих поверхонь (столів) $h_р$=0,75м. Оскільки світильники кріпляться до стелі, то їх висота над підлогою майже рівна висоті приміщення h_0=3м, що не суперечить вимогам ДБН В.2.5-28:2018, відповідно до яких h_{0min}=2,6-4 м.

Скориставшись формулою 5.1 визначимо висоту світильника над робочою поверхнею:

Висота світильника над робочою поверхнею:

$$h = h_0 - h_{р =} = 3,8-0,7 = 3,1м \quad (5.1)$$

Показник *і* становить:

$$I = \frac{a \cdot b}{h(a+b)} \quad (5.2)$$

$$I = \frac{9,5 \cdot 6,4}{3,1(9,5+6,4)} = 1.23$$

Згідно таблиці «Коефіцієнтів використання світлового потоку світильників з люмінесцентними лампами», коефіцієнт використання для світильника ЛСП 01 дорівнює η=0,44.

Згідно таблиці «Технічних даних деяких ламп розжарювання та люмінесцентних ламп» світловий потік однієї лампи ЛД - 40 становить Фл=2500лм.

Необхідна кількість світильників, для забезпечення необхідної освітленості робочих поверхонь з урахуванням коефіцієнту запасу, що враховує зниження освітленості в результаті забруднення та старіння ламп $К_З$=1,3, коефіцієнту нерівномірності освітлення Z=1,12 і мінімального освітлення приміщення, в якому виконують зорові роботи розряду IIIв становить Е=300лк. складає:

$$N= \frac{E \cdot a \cdot b \cdot K \cdot Z}{2 \cdot Фл \cdot \eta} \quad (5.3)$$

$$N= \frac{300 \cdot 9{,}5 \cdot 6{,}4 \cdot 1{,}3 \cdot 1{,}12}{2 \cdot 2500 \cdot 0{,}44} = 12$$

Отже в даному приміщені повинно для забезпечення правильного освітлення необхідно розмістити 12 світильників типу ЛСП 01 з двома лампами ЛД - 40 в кожному. Схема розташування світильників у приміщенні показана на рисунку 5.1.

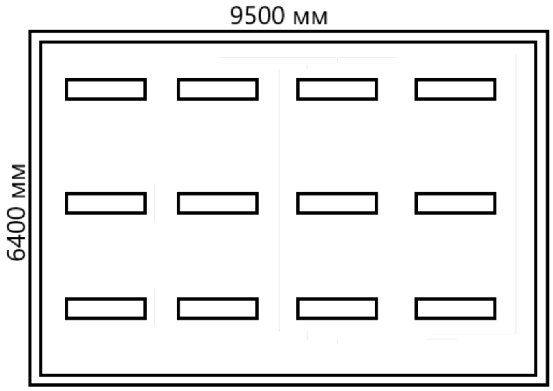


Рисунок 5.1 – Схема розміщення світильників в приміщені

5.2 Мотивація безпеки праці

Мотивація безпеки праці — це не просто формальність чи вимога законодавства; це глибоке розуміння та внутрішнє бажання кожного працівника дотримуватися правил і процедур, що забезпечують його власну безпеку та безпеку колег. Це складний психологічний процес, який формується під впливом багатьох чинників і виходить далеко за межі простих інструкцій чи покарань. Ефективна мотивація дозволяє створити культуру безпеки, де кожен відчуває особисту відповідальність і прагне до безтравматичного виробничого середовища, що повністю відповідає фундаментальним засадам Закону України "Про охорону праці".

Вплив керівництва та корпоративна культура.

Одним із найважливіших чинників, що формують мотивацію безпеки, є позиція керівництва компанії. Стаття 13 Закону України "Про охорону праці" чітко визначає, що роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці. Це не просто декларування, а вимога, що передбачає виділення ресурсів, впровадження сучасних систем управління безпекою та особисту участь керівництва у заходах з охорони праці. Коли вище керівництво демонструє справжню прихильність до питань охорони праці, працівники бачать, що їхня безпека є пріоритетом. Це формує так звану культуру безпеки, де норми безпечної поведінки стають невід'ємною частиною повсякденної діяльності, а не просто документом на папері. Така культура є основою для ефективного функціонування системи управління охороною праці (СУОП), вимоги до якої також визначені національними стандартами. [14]

Особистісне усвідомлення та цінності.

Мотивація до безпечної поведінки зростає, коли працівники усвідомлюють особисті ризики та потенційні наслідки нехтування правилами. Стаття 14 Закону України "Про охорону праці" та Стаття 159 Кодексу законів про працю України

(КЗпП) встановлюють обов'язок працівника дотримуватись вимог нормативно-правових актів з охорони праці, правил поводження з машинами, механізмами, устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту. Це зобов'язання підкріплене не лише можливістю дисциплінарної відповідальності, але й розумінням того, що нехтування правилами може призвести до реальних травм, інвалідності, втрати заробітку та негативного впливу на сім'ю. Програми навчання та інструктажі, що базуються на реальних випадках (звісно, без ефекту залякування), допомагають працівникам краще зрозуміти, що безпека — це пряма турбота про власне життя та здоров'я. Коли людина усвідомлює, що безпечна поведінка є вищою цінністю, вона свідомо прагне її дотримуватися. [15]

Активне залучення працівників до процесу забезпечення безпеки значно посилює їхню мотивацію. Хоча законодавство прямо не вимагає конкретних форм залучення, воно передбачає право працівників на участь у вирішенні питань охорони праці, зокрема через комісії з питань охорони праці підприємств, як це зазначено у Типовому положенні про комісію з питань охорони праці підприємства. Коли працівники мають можливість висловлювати свої пропозиції щодо покращення умов праці, брати участь у розробці інструкцій, ідентифікувати небезпеки та пропонувати рішення, вони відчувають себе частиною процесу. Це формує почуття власності та відповідальності. Замість того, щоб бути пасивними виконавцями, вони стають активними учасниками, які мають вплив на власне робоче середовище.

Стимулювання та заохочення.

Система стимулювання та заохочення за дотримання правил безпеки може бути ефективним інструментом, але лише як доповнення до інших чинників. Хоча українське законодавство не містить прямих норм щодо фінансового заохочення за безпечну працю, воно не забороняє роботодавцю використовувати такі механізми. Це можуть бути бонуси за безаварійну роботу, публічне визнання найбільш відповідальних працівників, надання додаткових вихідних або інші матеріальні та нематеріальні заохочення. Важливо, щоб ці заохочення були

справедливими, прозорими та не створювали ілюзії, що безпека праці є лише способом отримати винагороду, а не базовою необхідністю. Надмірний акцент на матеріальному стимулюванні без глибокого усвідомлення важливості безпеки може бути контрпродуктивним, адже основною мотивацією має бути запобігання нещасним випадкам та професійним захворюванням.

Безпечні умови праці та навчання.

Не менш важливою є фізична складова — забезпечення безпечних умов праці. Стаття 153 КЗпП та Стаття 13 Закону України "Про охорону праці" прямо зобов'язують роботодавця створити безпечні та нешкідливі умови праці. Навіть наймотивованіший працівник не зможе працювати безпечно, якщо обладнання несправне, засоби індивідуального захисту (ЗІЗ) відсутні або низької якості, а робоче місце не відповідає нормам. Компанія повинна інвестувати у сучасне обладнання, якісні ЗІЗ та підтримувати робочі місця у належному стані, дотримуючись відповідних державних стандартів та будівельних норм (ДБН). Крім того, якісне навчання та регулярні тренінги з охорони праці є не просто бажаними, а обов'язковими. Стаття 160 КЗпП та Стаття 18 Закону України "Про охорону праці" чітко вказують на обов'язок роботодавця щодо організації навчання, інструктажу і перевірки знань працівників з питань охорони праці. Ці навчання повинні бути зрозумілими, інтерактивними та відповідати специфіці роботи, допомагаючи працівникам набути необхідних знань та навичок для безпечного виконання своїх обов'язків.

Отже, мотивація безпеки праці – це не просто набір окремих заходів, а цілісна стратегія, що ґрунтується на вимогах українського законодавства. Вона включає лідерство керівництва, особистісне усвідомлення працівниками цінності власного здоров'я та життя (що є пріоритетом згідно зі Статтею 4 Закону України "Про охорону праці"), їх активну участь у процесі забезпечення безпеки, а також створення комфортних і безпечних умов праці. Лише комплексний підхід, що спирається на нормативно-правову базу України, може забезпечити стале покращення показників безпеки та створити середовище, де кожен працівник відчуває себе захищеним і відповідальним.

ВИСНОВКИ

Розробка програмного забезпечення "Система управління замовленнями бармена" для ресторану «Старий Млин» успішно завершена. Створена система версії 1.0 являє собою надійний та ефективний інструмент, що значно автоматизує ключові процеси прийому та обробки замовлень, замінюючи традиційний паперовий облік на цифрову платформу.

Цей етап розробки заклав міцний фундамент, дозволивши досягти поставлених цілей початкової фази проєкту. Система забезпечує централізоване управління замовленнями, каталогом напоїв та персоналом, що сприяє підвищенню операційної ефективності ресторану. Вибір технологічного стеку, що включає C++ та Qt, разом з базою даних PostgreSQL (як основною СУБД для виробничого середовища), гарантує високу продуктивність, надійність, гнучкість та кросплатформенність рішення.

Розроблений інтерфейс користувача вирізняється простотою та інтуїтивністю, що мінімізує час на навчання персоналу та спрощує щоденні операції. Проведений комплексний процес тестування — функціональне, інтеграційне та тестування зручності використання — підтвердив стабільність системи та її готовність до впровадження в реальне робоче середовище.

Таким чином, розроблена система є значним кроком у цифровізації робочих процесів ресторану «Старий Млин», відкриваючи шлях для майбутніх розширень та подальшого підвищення ефективності закладу.

ПЕРЕЛІК ПОСИЛАНЬ

1) Toast _ Restaurant Point of Sale & Management System _ Toast POS. toasttab.com. URL: https://pos.toasttab.com/ (дата звернення: 10.05.2025).

2) Leading Point of Sale (POS) & Commerce Platform | Lightspeed. Lightspeed. URL: https://www.lightspeedhq.com/ (дата звернення: 10.05.2025).

3) Poster POS. Poster POS – программа автоматизации общепита на планшете – Poster POS. URL: https://joinposter.com/ua (дата звернення: 12.05.2025).

4) Kitchen Display Systems. kitchendisplaysystem. URL: https://www.kitchendisplaysystems.com/ (дата звернення: 12.05.2025).

5) Uber eats, DoorDash and Glovo Checkout Process Analysis - Luis Fernandez. Luis Fernandez. URL: https://luiselias.com/uber-eats-doordash-and-glovo-checkout-process-analysis/ (дата звернення: 12.05.2025).

6) SQLite Home Page. SQLite Home Page. URL: https://sqlite.org/ (дата звернення: 14.06.2025).

7) What Is PostgreSQL?. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/current/intro-whatis.html (дата звернення: 14.06.2025).

8) Страуструп Б. Мова програмування C++. Спеціальне видання : навч. посібник. Львів : Вид-во Львівського Університету, 2018. 720 с.

9) Embedded Software Development Tools & Cross Platform IDE | Qt Creator. Qt | Tools for Each Stage of Software Development Lifecycle. URL: https://www.qt.io/product/development-tools (дата звернення: 14.06.2025).

10) Model-View-ViewModel – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel (дата звернення: 14.06.2025).

11) Нормалізація баз даних – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Нормалізація_баз_даних (дата звернення: 14.06.2025).

12) ДБН В.2.5-28:2018. Природне і штучне освітлення. На заміну ДБН В.2.5-28-2006 ; чинний від 2019-03-01. Вид. офіц. Київ : Мінрегіон України, 2018. 133 с.

13) ДСТУ EN 12464-1:2016. Світло та освітлення. Освітлення робочих місць. Частина 1. Внутрішні робочі місця. Чинний від 2017-12-01. Вид. офіц. Київ : ДП "УкрНДНЦ", 2016. 47 с.

14) Про охорону праці : Закон України від 14.10.1992 № 2694-XII : станом на 4 квіт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 07.06.2025).

15) Кодекс законів про працю України : Кодекс України від 10.12.1971 № 322-VIII : станом на 2 трав. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 07.06.2025).

ДОДАТКИ

Додаток А. Лістинг файлу «BarOrderSystem.pro»

```
#-------------------------------------------------
# Система управління замовленнями ресторану "Старий Млин"
# Версія: 1.0
# Qt проект файл для Qt Creator
#-------------------------------------------------

QT += core widgets sql

CONFIG += c++17

TARGET = BarOrderSystem
TEMPLATE = app

# Версія додатку
VERSION = 1.0.0

# Інформація про додаток
QMAKE_TARGET_COMPANY = "Ресторан Старий Млин"
QMAKE_TARGET_PRODUCT = "Система управління замовленнями"
QMAKE_TARGET_DESCRIPTION = "Програмне забезпечення для автоматизації прийому та обробки замовлень бармена"
QMAKE_TARGET_COPYRIGHT = "Copyright (c) 2025 ТОВ Ресторан Старий Млин"

# Конфігурація збірки
CONFIG(debug, debug|release) {
    TARGET = $$TARGET"_debug"
    DEFINES += DEBUG_BUILD
} else {
    DEFINES += QT_NO_DEBUG_OUTPUT
    DEFINES += RELEASE_BUILD
```

```
}

# Підтримка Unicode
DEFINES += UNICODE

# Основні джерельні файли
SOURCES += \
    main.cpp \
    mainwindow.cpp \
    widgets/ordermanagement.cpp \
    widgets/menumanagement.cpp \
    widgets/employeemanagement.cpp \
    database/databasemanager.cpp \
    database/repositories.cpp \
    config/configmanager.cpp \
    auth/authmanager.cpp \
    auth/logindialog.cpp

# Заголовкові файли
HEADERS += \
    mainwindow.h \
    models/employee.h \
    models/drink.h \
    models/order.h \
    widgets/ordermanagement.h \
    widgets/menumanagement.h \
    widgets/employeemanagement.h \
    database/databasemanager.h \
    database/repositories.h \
    config/configmanager.h \
    auth/authmanager.h \
    auth/logindialog.h

# Ресурси (якщо є)
```

```
RESOURCES += \
    resources/application.qrc

# Іконка програми
win32:RC_ICONS = resources/icons/app_icon.ico
macx:ICON = resources/icons/app_icon.icns

# Платформо-специфічні налаштування

# Windows
win32 {
    # MSVC конфігурація
    msvc {
        QMAKE_CXXFLAGS += /std:c++17
        QMAKE_CXXFLAGS_WARN_ON += /W4
    }

    # MinGW конфігурація
    gcc {
        QMAKE_CXXFLAGS += -std=c++17
        QMAKE_CXXFLAGS_WARN_ON += -Wall -Wextra
    }

    # Додаткові бібліотеки Windows
    LIBS += -luser32 -lshell32

    # Маніфест для Windows
    CONFIG += embed_manifest_exe
}

# Linux
linux {
    QMAKE_CXXFLAGS += -std=c++17
    QMAKE_CXXFLAGS_WARN_ON += -Wall -Wextra -Wpedantic
```

```
    # Додаткові шляхи для Linux
    target.path = /usr/local/bin
    INSTALLS += target

    # Desktop файл
    desktop.files = resources/linux/barordersystem.desktop
    desktop.path = /usr/share/applications
    INSTALLS += desktop
}

# macOS
macx {
    QMAKE_CXXFLAGS += -std=c++17
    QMAKE_MACOSX_DEPLOYMENT_TARGET = 10.15

    # Bundle інформація
    QMAKE_INFO_PLIST = resources/macos/Info.plist
}

# Додаткові включення для різних платформ
INCLUDEPATH += $$PWD
DEPENDPATH += $$PWD

# Створення необхідних каталогів при збірці
CONFIG += create_prl

# Налаштування для релізної збірки
release {
    # Оптимізація
    QMAKE_CXXFLAGS_RELEASE += -O2

    # Видалення debug інформації
    QMAKE_LFLAGS_RELEASE += -s
```

```
    # Статичне лінкування (опціонально)
    # CONFIG += static
}

# Налаштування для debug збірки
debug {
    # Debug символи
    QMAKE_CXXFLAGS_DEBUG += -g

    # Додаткові перевірки
    DEFINES += QT_QML_DEBUG
}

# Додаткові каталоги
OBJECTS_DIR = build/obj
MOC_DIR = build/moc
RCC_DIR = build/rcc
UI_DIR = build/ui

# Чистка тимчасових файлів
QMAKE_CLEAN += -r build/

# Перевірка версії Qt
lessThan(QT_MAJOR_VERSION, 6) {
    error("Потрібна Qt версія 6.0 або новіша!")
}

# Повідомлення про збірку
message("Збірка проекту BarOrderSystem")
message("Qt версія: $$QT_VERSION")
message("Компілятор: $$QMAKE_COMPILER")
message("Платформа: $$QMAKE_PLATFORM")
```

```
# Додаткові налаштування для розробки
CONFIG(debug, debug|release) {
    message("Debug збірка")
} else {
    message("Release збірка")
}

# Попередження компілятора
QMAKE_CXXFLAGS_WARN_ON += -Wno-unused-parameter

# Кодування файлів
CODECFORTR = UTF-8
CODECFORSRC = UTF-8

# Локалізація (майбутня можливість)
TRANSLATIONS += \
    translations/barordersystem_uk.ts \
    translations/barordersystem_en.ts

# Додаткові файли для включення в проект
OTHER_FILES += \
    README.md \
    LICENSE \
    docs/UserGuide.md \
    docs/AdminGuide.md \
    docs/DatabaseSetup.md \
    scripts/deploy.sh \
    scripts/install.nsi

# Генерація make файлів з правильним кодуванням
QMAKE_ENCODING = UTF-8

# Додаткові перевірки якості коду (якщо доступні)
exists($$PWD/.clang-format) {
```

```
    message("Знайдено .clang-format конфігурацію")
}

# Автоматичне копіювання ресурсів
copydata.commands = $(COPY_DIR) $$PWD/resources $$OUT_PWD
first.depends = $(first) copydata
export(first.depends)
export(copydata.commands)
QMAKE_EXTRA_TARGETS += first copydata

# Команди для очистки
distclean.commands = rm -rf build/ && rm -f Makefile*
QMAKE_EXTRA_TARGETS += distclean

# Підтримка для додаткових бібліотек
# (раскоментувати якщо потрібно)
#
# # QML підтримка (для майбутніх версій)
# # QT += qml quick
#
# # Мережа (для віддаленого доступу)
# # QT += network
#
# # Принтер підтримка (для друку чеків)
# # QT += printsupport
#
# # Multimedia (для звукових сповіщень)
# # QT += multimedia

# Перевірка наявності необхідних файлів
!exists(main.cpp) {
    error("Файл main.cpp не знайдено!")
}
```

```
!exists(mainwindow.h) {
    error("Файл mainwindow.h не знайдено!")
}

# Налаштування для CI/CD
ci {
    CONFIG += silent
    DEFINES += CI_BUILD
}

# Налаштування для Coverage тестування
coverage {
    QMAKE_CXXFLAGS += --coverage
    QMAKE_LFLAGS += --coverage
    LIBS += -lgcov
}

# Додаткові функції для генерації документації
docs.commands = doxygen Doxyfile
QMAKE_EXTRA_TARGETS += docs

# Пакування для різних платформ
win32 {
    package.commands = windeployqt $$TARGET.exe
    QMAKE_EXTRA_TARGETS += package
}

macx {
    package.commands = macdeployqt $$TARGET.app -dmg
    QMAKE_EXTRA_TARGETS += package
}

linux {
    package.commands = linuxdeploy --appdir AppDir --executable $$TARGET --
```

```
output appimage
    QMAKE_EXTRA_TARGETS += package
}

# Фінальне повідомлення
message("Конфігурація проекту завершена")
message("Для збірки виконайте: qmake && make")

# Налаштування для Qt Widgets Application
QT_WIDGETS_LIB = yes

# Завершення файлу проекту
```

Додаток Б. Лістинг файлу «main.cpp»

```
/**
 * @file main.cpp
 * @brief Головний файл додатку системи управління замовленнями
 * @author Навроцький Сергій
 * @date Червень 2025
 * @version 1.0
 */

#include "mainwindow.h"
#include "config/configmanager.h"
#include "database/databasemanager.h"
#include <QApplication>
#include <QStyleFactory>
#include <QMessageBox>
#include <QDir>
#include <QStandardPaths>
#include <QTranslator>
#include <QLocale>
#include <QDebug>

/**
 * @brief Налаштування стилів додатку
 * @param app Посилання на QApplication
 */
void setupApplicationStyles(QApplication& app)
{
    // Встановлення стилю
    app.setStyle(QStyleFactory::create("Fusion"));

    // CSS стилі для поліпшення вигляду
    app.setStyleSheet(R"(
        /* Загальні стилі */
```

```
QMainWindow {
    background-color: #f0f0f0;
}

/* Стилі для табів */
QTabWidget::pane {
    border: 1px solid #c0c0c0;
    background-color: white;
    border-radius: 4px;
}

QTabBar::tab {
    background-color: #e0e0e0;
    padding: 8px 16px;
    margin-right: 2px;
    border-top-left-radius: 4px;
    border-top-right-radius: 4px;
}

QTabBar::tab:selected {
    background-color: white;
    border-bottom: 2px solid #2196F3;
}

QTabBar::tab:hover {
    background-color: #f0f0f0;
}

/* Стилі для таблиць */
QTableWidget {
    gridline-color: #d0d0d0;
    selection-background-color: #2196F3;
    alternate-background-color: #f5f5f5;
    border: 1px solid #c0c0c0;
```

```
        border-radius: 4px;
    }

    QTableWidget::item {
        padding: 8px;
        border-bottom: 1px solid #e0e0e0;
    }

    QTableWidget::item:selected {
        background-color: #2196F3;
        color: white;
    }

    QHeaderView::section {
        background-color: #e8e8e8;
        padding: 8px;
        border: none;
        border-right: 1px solid #c0c0c0;
        font-weight: bold;
    }

    /* Стилі для кнопок */
    QPushButton {
        padding: 8px 16px;
        border: 1px solid #c0c0c0;
        border-radius: 4px;
        background-color: #f8f8f8;
        font-size: 13px;
        min-width: 80px;
    }

    QPushButton:hover {
        background-color: #e8e8e8;
        border-color: #2196F3;
```

```
}

QPushButton:pressed {
    background-color: #d0d0d0;
}

QPushButton:disabled {
    background-color: #f0f0f0;
    color: #888888;
    border-color: #e0e0e0;
}

/* Стилі для форм */
QLineEdit, QComboBox, QSpinBox, QDoubleSpinBox, QTextEdit {
    padding: 6px;
    border: 1px solid #c0c0c0;
    border-radius: 4px;
    background-color: white;
}

QLineEdit:focus, QComboBox:focus, QSpinBox:focus,
QDoubleSpinBox:focus, QTextEdit:focus {
    border: 2px solid #2196F3;
}

QComboBox::drop-down {
    border: none;
    width: 20px;
}

QComboBox::down-arrow {
    image: none;
    border-style: solid;
    border-width: 4px 4px 0px 4px;
```

```
        border-color: #666666 transparent transparent transparent;
    }

    /* Стилі для статус-бару */
    QStatusBar {
        background-color: #e0e0e0;
        border-top: 1px solid #c0c0c0;
        color: #333333;
    }

    QStatusBar QLabel {
        color: #333333;
        padding: 2px 8px;
    }

    /* Стилі для груп */
    QGroupBox {
        font-weight: bold;
        border: 2px solid #c0c0c0;
        border-radius: 5px;
        margin-top: 1ex;
        padding-top: 10px;
    }

    QGroupBox::title {
        subcontrol-origin: margin;
        left: 10px;
        padding: 0 5px 0 5px;
    }

    /* Стилі для діалогів */
    QDialog {
        background-color: #f5f5f5;
    }
```

```
        /* Стилі для повідомлень */
        QMessageBox {
            background-color: white;
        }

        /* Стилі для прогрес-барів */
        QProgressBar {
            border: 1px solid #c0c0c0;
            border-radius: 4px;
            text-align: center;
        }

        QProgressBar::chunk {
            background-color: #2196F3;
            border-radius: 3px;
        }
    )");
}

/**
 * @brief Налаштування локалізації
 * @param app Посилання на QApplication
 */
void setupLocalization(QApplication& app)
{
    QTranslator* translator = new QTranslator(&app);

    // Спроба завантажити переклад для поточної локалі
    QString locale = QLocale::system().name();
    if          (translator->load(QString("barordersystem_%1").arg(locale),
":/translations")) {
        app.installTranslator(translator);
        qDebug() << "Завантажено переклад для локалі:" << locale;
```

```
    } else {
        qDebug() << "Переклад для локалі" << locale << "не знайдено,
використовується за замовчуванням";
        delete translator;
    }
}

/**
 * @brief Налаштування логування
 */
void setupLogging()
{
#ifdef DEBUG_BUILD
    // В debug режимі показуємо всі повідомлення
    QLoggingCategory::setFilterRules("*.debug=true");
    qDebug() << "Debug режим увімкнено";
#else
    // В release режимі приховуємо debug повідомлення
    QLoggingCategory::setFilterRules("*.debug=false");
#endif
}

/**
 * @brief Перевірка системних вимог
 * @return true якщо система відповідає вимогам
 */
bool checkSystemRequirements()
{
    // Перевірка версії Qt
    if (QT_VERSION < QT_VERSION_CHECK(6, 0, 0)) {
        QMessageBox::critical(nullptr, "Помилка системи",
                              "Потрібна Qt версія 6.0 або новіша!\n"
                              "Поточна версія: " + QString(QT_VERSION_STR));
        return false;
```

```
    }

    // Перевірка доступності SQL модуля
    if (!QSqlDatabase::isDriverAvailable("QPSQL")) {
        QMessageBox::critical(nullptr, "Помилка бази даних",
                              "SQL драйвер недоступний!\n"
                              "Переконайтеся що Qt встановлено з SQL
підтримкою.");
        return false;
    }

    // Перевірка прав на запис в папку додатку
    QString appDataPath =
QStandardPaths::writableLocation(QStandardPaths::AppDataLocation);
    QDir appDataDir;
    if (!appDataDir.mkpath(appDataPath)) {
        QMessageBox::critical(nullptr, "Помилка файлової системи",
                              "Неможливо створити папку для даних
додатку:\n" +
                              appDataPath + "\n\nПеревірте права доступу.");
        return false;
    }

    return true;
}

/**
 * @brief Обробник критичних помилок додатку
 */
void criticalErrorHandler(const QString& error)
{
    qCritical() << "Критична помилка:" << error;

    QMessageBox msgBox;
```

```
    msgBox.setIcon(QMessageBox::Critical);
    msgBox.setWindowTitle("Критична помилка");
    msgBox.setText("Виникла критична помилка в роботі додатку:");
    msgBox.setDetailedText(error);
    msgBox.setStandardButtons(QMessageBox::Ok);
    msgBox.exec();
}

/**
 * @brief Головна функція додатку
 * @param argc Кількість аргументів командного рядка
 * @param argv Масив аргументів командного рядка
 * @return Код виходу програми
 */
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    // Налаштування інформації про додаток
    app.setApplicationName("Bar Order System");
    app.setApplicationVersion("1.0.0");
    app.setApplicationDisplayName("Система управління замовленнями");
    app.setOrganizationName("Ресторан Старий Млин");
    app.setOrganizationDomain("staryi-mlyn.ua");

    // Налаштування атрибутів додатку
    app.setAttribute(Qt::AA_UseHighDpiPixmaps);
    app.setAttribute(Qt::AA_EnableHighDpiScaling);

    // Налаштування логування
    setupLogging();

    qInfo() << "Запуск додатку" << app.applicationName()
            << "версії" << app.applicationVersion();
```

```
    // Перевірка системних вимог
    if (!checkSystemRequirements()) {
        qCritical() << "Системні вимоги не виконані";
        return 1;
    }

    // Налаштування стилів
    setupApplicationStyles(app);

    // Налаштування локалізації
    setupLocalization(app);

    // Ініціалізація конфігурації
    ConfigManager* config = ConfigManager::instance();
    qInfo() << "Файл конфігурації:" << config->getConfigFilePath();

    // Отримання параметрів БД з конфігурації
    DatabaseManager::ConnectionParams dbParams = config-
>getDatabaseParams();

    // Якщо SQLite і шлях не заданий, використовуємо шлях за замовчуванням
    if (dbParams.type == DatabaseManager::SQLite &&
dbParams.filePath.isEmpty()) {
        dbParams = config->getDefaultConfig();
        config->saveDatabaseParams(dbParams);
        qInfo() << "Використовується SQLite БД за замовчуванням:" <<
dbParams.filePath;
    }

    // Підключення до бази даних
    qInfo() << "Підключення до бази даних...";
    if (!DatabaseManager::instance()->connect(dbParams)) {
        QString dbTypeStr = (dbParams.type == DatabaseManager::SQLite) ?
```

```
"SQLite" : "PostgreSQL";
        QString errorMsg = QString("Не вдалося підключитися до бази даних %1!\n\n"
                                   "Тип БД: %2\n"
                                   "Конфігурація: %3\n\n"
                                   "Перевірте параметри підключення у файлі конфігурації.")
                               .arg(dbTypeStr)
                               .arg(dbTypeStr)
                               .arg(config->getConfigFilePath());

        criticalErrorHandler(errorMsg);
        return 2;
    }

    qInfo() << "База даних підключена успішно";

    // Створення та показ головного вікна
    try {
        MainWindow window;

        // Центрування вікна на екрані
        window.move(QApplication::primaryScreen()->geometry().center() - window.rect().center());

        window.show();

        qInfo() << "Головне вікно відображено";

        // Запуск головного циклу обробки подій
        int result = app.exec();

        qInfo() << "Завершення роботи додатку з кодом:" << result;
```

```
        return result;

    } catch (const std::exception& e) {
        criticalErrorHandler(QString("Виняток C++: %1").arg(e.what()));
        return 3;
    } catch (...) {
        criticalErrorHandler("Невідомий виняток при створенні головного
вікна");
        return 4;
    }
}
```

Додаток В. Лістинг файлу «mainwindow.h»

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTabWidget>
#include <QMenuBar>
#include <QStatusBar>
#include <QLabel>
#include <QAction>
#include <QTimer>
#include <QCloseEvent>

class OrderManagement;
class MenuManagement;
class EmployeeManagement;

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
        ~MainWindow();

protected:
    void closeEvent(QCloseEvent *event) override;

private slots:
    void initializeDatabase();
    void onDatabaseError(const QString& error);
    void onUserLoggedIn();
    void onUserLoggedOut();
```

```
    void showAbout();
    void logout();
    void changePassword();
    void updateUserInterface();
    void updateCurrentTime();

private:
    void setupMenuBar();
    void setupStatusBar();
    void setupTabWidget();
    void updateTabsAccess();
    void showLoginDialog();
    void createActions();
    void connectSignals();

private:
    QTabWidget *m_tabWidget;
    OrderManagement *m_orderManagement;
    MenuManagement *m_menuManagement;
    EmployeeManagement *m_employeeManagement;
    QMenuBar *m_menuBar;
    QStatusBar *m_statusBar;
    QLabel *m_userLabel;
    QLabel *m_timeLabel;
    QLabel *m_statusLabel;
    QAction *m_logoutAction;
    QAction *m_changePasswordAction;
    QAction *m_aboutAction;
    QAction *m_exitAction;
    QTimer *m_timeUpdateTimer;
    bool m_isInitialized;
    bool m_isFirstRun;
};
#endif // MAINWINDOW_H
```

Додаток Г. Лістинг файлу «mainwindow.cpp»

```
#include "mainwindow.h"
#include "widgets/ordermanagement.h"
#include "widgets/menumanagement.h"
#include "widgets/employeemanagement.h"
#include "database/databasemanager.h"
#include "auth/authmanager.h"
#include "auth/logindialog.h"
#include <QApplication>
#include <QMessageBox>
#include <QStatusBar>
#include <QMenuBar>
#include <QTimer>
#include <QDateTime>
#include <QCloseEvent>
#include <QDesktopServices>
#include <QUrl>
#include <QKeySequence>
#include <QIcon>
#include <QDebug>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , m_tabWidget(nullptr)
    , m_orderManagement(nullptr)
    , m_menuManagement(nullptr)
    , m_employeeManagement(nullptr)
    , m_menuBar(nullptr)
    , m_statusBar(nullptr)
    , m_userLabel(nullptr)
    , m_timeLabel(nullptr)
    , m_statusLabel(nullptr)
    , m_logoutAction(nullptr)
```

```
    , m_changePasswordAction(nullptr)
    , m_aboutAction(nullptr)
    , m_exitAction(nullptr)
    , m_timeUpdateTimer(nullptr)
    , m_isInitialized(false)
    , m_isFirstRun(true)
{
    qDebug() << "Ініціалізація головного вікна...";

    // Налаштування основних властивостей вікна
    setWindowTitle("Система управління замовленнями - Ресторан 'Старий
Млин'");
    setMinimumSize(1000, 700);
    setWindowIcon(QIcon(":/icons/app_icon.png"));

    // Ініціалізація бази даних
    initializeDatabase();

    // Налаштування UI компонентів
    setupMenuBar();
    setupStatusBar();
    setupTabWidget();

    // Створення дій та підключення сигналів
    createActions();
    connectSignals();

    // Ініціалізація таймерів
    m_timeUpdateTimer = new QTimer(this);
    connect(m_timeUpdateTimer, &QTimer::timeout, this,
&MainWindow::updateCurrentTime);
    m_timeUpdateTimer->start(1000); // Оновлення кожну секунду

    // Початкове оновлення часу
```

```
    updateCurrentTime();

    qDebug() << "Головне вікно ініціалізовано";

    // Показ діалогу входу після створення UI
    QTimer::singleShot(100, this, &MainWindow::showLoginDialog);
}

MainWindow::~MainWindow()
{
    qDebug() << "Знищення головного вікна...";

    // Зупинка таймерів
    if (m_timeUpdateTimer) {
        m_timeUpdateTimer->stop();
    }

    // Відключення від бази даних
    DatabaseManager::instance()->disconnect();

    qDebug() << "Головне вікно знищено";
}

void MainWindow::setupMenuBar()
{
    qDebug() << "Налаштування головного меню...";

    m_menuBar = menuBar();

    // Меню "Файл"
    QMenu* fileMenu = m_menuBar->addMenu("&Файл");

    m_logoutAction = fileMenu->addAction("&Вийти з облікового запису");
    m_logoutAction->setShortcut(QKeySequence("Ctrl+L"));
```

```
    m_logoutAction->setIcon(QIcon(":/icons/logout.png"));
    m_logoutAction->setStatusTip("Вихід з поточного облікового запису");
    m_logoutAction->setEnabled(false); // Спочатку неактивна

    fileMenu->addSeparator();

    m_exitAction = fileMenu->addAction("&Вихід");
    m_exitAction->setShortcut(QKeySequence::Quit);
    m_exitAction->setIcon(QIcon(":/icons/exit.png"));
    m_exitAction->setStatusTip("Закриття додатку");

    // Меню "Налаштування"
    QMenu* settingsMenu = m_menuBar->addMenu("&Налаштування");

    m_changePasswordAction = settingsMenu->addAction("&Змінити пароль");
    m_changePasswordAction->setIcon(QIcon(":/icons/password.png"));
    m_changePasswordAction->setStatusTip("Зміна паролю поточного
користувача");
    m_changePasswordAction->setEnabled(false); // Спочатку неактивна

    settingsMenu->addSeparator();

    QAction* preferencesAction = settingsMenu->addAction("&Налаштування
системи");
    preferencesAction->setIcon(QIcon(":/icons/settings.png"));
    preferencesAction->setStatusTip("Загальні налаштування системи");
    preferencesAction->setEnabled(false); // Буде реалізовано в майбутніх
версіях

    // Меню "Допомога"
    QMenu* helpMenu = m_menuBar->addMenu("&Допомога");

    QAction* userGuideAction = helpMenu->addAction("&Посібник
користувача");
```

```
    userGuideAction->setIcon(QIcon(":/icons/help.png"));
    userGuideAction->setShortcut(QKeySequence::HelpContents);
    userGuideAction->setStatusTip("Відкрити посібник користувача");

    helpMenu->addSeparator();

    m_aboutAction = helpMenu->addAction("&Про програму");
    m_aboutAction->setIcon(QIcon(":/icons/about.png"));
    m_aboutAction->setStatusTip("Інформація про програму");

    QAction* aboutQtAction = helpMenu->addAction("Про &Qt");
    aboutQtAction->setStatusTip("Інформація про фреймворк Qt");

    // Підключення дій до слотів (буде зроблено в connectSignals)

    qDebug() << "Головне меню налаштовано";
}

void MainWindow::setupStatusBar()
{
    qDebug() << "Налаштування статус-бару...";

    m_statusBar = statusBar();

    // Лейбл для повідомлень системи (ліворуч)
    m_statusLabel = new QLabel("Готово до роботи");
    m_statusBar->addWidget(m_statusLabel, 1); // stretch factor = 1

    // Розділювач
    QLabel* separator1 = new QLabel(" | ");
    separator1->setStyleSheet("color: #888888;");
    m_statusBar->addPermanentWidget(separator1);

    // Інформація про користувача (центр)
```

```
    m_userLabel = new QLabel("Не авторизований");
    m_userLabel->setStyleSheet("color: red; font-weight: bold; padding: 2px
8px;");
    m_statusBar->addPermanentWidget(m_userLabel);

    // Розділювач
    QLabel* separator2 = new QLabel(" | ");
    separator2->setStyleSheet("color: #888888;");
    m_statusBar->addPermanentWidget(separator2);

    // Поточний час (праворуч)
    m_timeLabel = new QLabel();
    m_timeLabel->setStyleSheet("color:  #333333;  font-family:  monospace;
padding: 2px 8px;");
    m_statusBar->addPermanentWidget(m_timeLabel);

    qDebug() << "Статус-бар налаштовано";
}

void MainWindow::setupTabWidget()
{
    qDebug() << "Налаштування табового віджету...";

    m_tabWidget = new QTabWidget(this);
    m_tabWidget->setTabPosition(QTabWidget::North);
    m_tabWidget->setDocumentMode(true);
    m_tabWidget->setMovable(false); // Заборонити переміщення табів

    setCentralWidget(m_tabWidget);

    // Створення модулів (але не додаємо їх до табів поки користувач не
авторизований)
    m_orderManagement = new OrderManagement(this);
m_menuManagement = new MenuManagement(this);
```

```
    m_employeeManagement = new EmployeeManagement(this);

    // Початково табів немає - вони додаються після авторизації
    m_tabWidget->hide();

    qDebug() << "Табовий віджет налаштовано";
}

void MainWindow::createActions()
{
    // Дії вже створені в setupMenuBar(), тут можна додати додаткові налаштування
    // Наприклад, іконки, якщо не були встановлені раніше
}

void MainWindow::connectSignals()
{
    qDebug() << "Підключення сигналів та слотів...";

    // Підключення дій меню
    connect(m_logoutAction, &QAction::triggered, this, &MainWindow::logout);
    connect(m_exitAction, &QAction::triggered, this, &QWidget::close);
    connect(m_changePasswordAction, &QAction::triggered, this, &MainWindow::changePassword);
    connect(m_aboutAction, &QAction::triggered, this, &MainWindow::showAbout);

    // Підключення до AuthManager
    connect(AuthManager::instance(), &AuthManager::loginSuccessful,
            this, &MainWindow::onUserLoggedIn);
    connect(AuthManager::instance(), &AuthManager::loggedOut,
            this, &MainWindow::onUserLoggedOut);
```

```
    // Підключення до DatabaseManager
    connect(DatabaseManager::instance(), &DatabaseManager::errorOccurred,
            this, &MainWindow::onDatabaseError);

    qDebug() << "Сигнали підключено";
}

void MainWindow::showLoginDialog()
{
    qDebug() << "Показ діалогу входу...";

    if (AuthManager::instance()->isLoggedIn()) {
        qDebug() << "Користувач вже авторизований";
        return;
    }

    LoginDialog loginDialog(this);
    loginDialog.setModal(true);

    if (loginDialog.exec() != QDialog::Accepted) {
        qInfo() << "Користувач скасував вхід";
        QApplication::quit();
        return;
    }

    // Успішна авторизація обробляється в onUserLoggedIn()
    qDebug() << "Діалог входу завершено";
}

void MainWindow::onUserLoggedIn()
{
    qDebug() << "Обробка успішного входу користувача...";

    Employee currentUser = AuthManager::instance()->getCurrentUser();
```

```
    // Оновлення статус-бару
    QString roleText;
    if (currentUser.role == "admin") {
        roleText = "Адміністратор";
    } else if (currentUser.role == "bartender") {
        roleText = "Бармен";
    } else if (currentUser.role == "waiter") {
        roleText = "Офіціант";
    } else {
        roleText = currentUser.role;
    }

    m_userLabel->setText(QString("%1 %2 (%3)")
                         .arg(currentUser.firstName)
                         .arg(currentUser.lastName)
                         .arg(roleText));
    m_userLabel->setStyleSheet("color: green; font-weight: bold; padding:
2px 8px;");

    m_statusLabel->setText("Успішно авторизований");

    // Активація дій меню
    m_logoutAction->setEnabled(true);
    m_changePasswordAction->setEnabled(true);

    // Налаштування доступу до табів
    updateTabsAccess();
    updateUserInterface();

    m_isInitialized = true;
    m_isFirstRun = false;

    qInfo()     <<     "Користувач"     <<     currentUser.firstName     <<
```

```
currentUser.lastName
            << "(" << roleText << ") увійшов в систему";
}

void MainWindow::onUserLoggedOut()
{
    qDebug() << "Обробка виходу користувача...";

    // Очищення інтерфейсу
    m_tabWidget->clear();
    m_tabWidget->hide();

    // Оновлення статус-бару
    m_userLabel->setText("Не авторизований");
    m_userLabel->setStyleSheet("color: red; font-weight: bold; padding: 2px
8px;");
    m_statusLabel->setText("Вийшли з системи");

    // Деактивація дій меню
    m_logoutAction->setEnabled(false);
    m_changePasswordAction->setEnabled(false);

    m_isInitialized = false;

    // Показати діалог входу знову через невелику затримку
    QTimer::singleShot(500, this, &MainWindow::showLoginDialog);

    qInfo() << "Користувач вийшов з системи";
}

void MainWindow::updateTabsAccess()
{
    qDebug() << "Оновлення доступу до табів...";
```

```
    if (!AuthManager::instance()->isLoggedIn()) {
        qDebug() << "Користувач не авторизований";
        return;
    }

    m_tabWidget->clear();

    // Додавання табів згідно з правами доступу
    if (AuthManager::instance()->canAccessOrders()) {
        m_tabWidget->addTab(m_orderManagement,  QIcon(":/icons/order.png"),
"Замовлення");
        qDebug() << "Додано таб 'Замовлення'";
    }

    if (AuthManager::instance()->canManageMenu()) {
        m_tabWidget->addTab(m_menuManagement,    QIcon(":/icons/menu.png"),
"Меню");
        qDebug() << "Додано таб 'Меню'";
    }

    if (AuthManager::instance()->canManageEmployees()) {
        m_tabWidget->addTab(m_employeeManagement,
QIcon(":/icons/employee.png"), "Персонал");
        qDebug() << "Додано таб 'Персонал'";
    }

    // Показ віджету якщо є хоча б один таб
    if (m_tabWidget->count() > 0) {
        m_tabWidget->show();
        qDebug()  <<  "Табовий  віджет  показано,  кількість  табів:"  <<
m_tabWidget->count();
    } else {
        m_tabWidget->hide();
        m_statusLabel->setText("Немає доступних розділів для вашої ролі");
```

```
        qWarning() << "Немає доступних табів для поточного користувача";
    }
}

void MainWindow::updateUserInterface()
{
    qDebug() << "Оновлення користувацького інтерфейсу...";

    if (!m_isInitialized) {
        qDebug() << "Система не ініціалізована";
        return;
    }

    // Оновлення всіх табів
    if (AuthManager::instance()->canAccessOrders()) {
        m_orderManagement->refreshOrdersList();
    }

    if (AuthManager::instance()->canManageMenu()) {
        m_menuManagement->refreshDrinksList();
    }

    if (AuthManager::instance()->canManageEmployees()) {
        m_employeeManagement->refreshEmployeesList();
    }

    qDebug() << "Інтерфейс оновлено";
}

void MainWindow::logout()
{
    qDebug() << "Запит на вихід з системи...";

    int ret = QMessageBox::question(this, "Підтвердження",
```

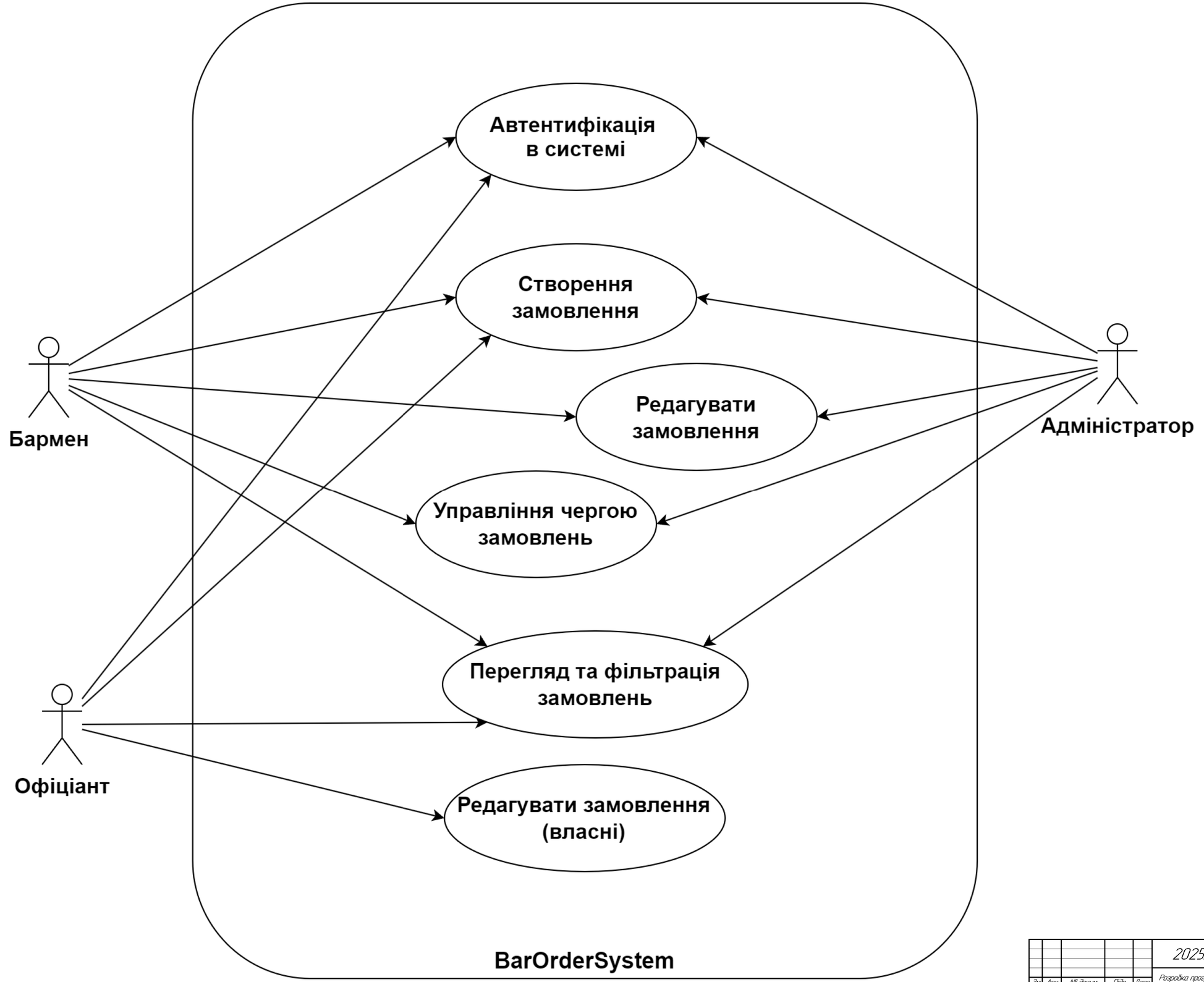
```
                                    "Ви дійсно хочете вийти з системи?",
                                    QMessageBox::Yes | QMessageBox::No,
                                    QMessageBox::No);

    if (ret == QMessageBox::Yes) {
        qInfo() << "Користувач підтвердив вихід з системи";
        AuthManager::instance()->logout();
    } else {
        qDebug() << "Вихід з системи скасовано";
    }
}

void MainWindow::changePassword()
{
    qDebug() << "Запит на зміну паролю...";

    // Поки що заглушка - функція буде реалізована в майбутніх версіях
    QMessageBox::information(this, "Зміна паролю",
                             "Функція зміни паролю буде додана в наступній
версії.\n\n"
                             "Поки що використовуйте демонстраційні
паролі:\n"
                             "• Офіціанти: прізвище в нижньому регістрі\n"
                             "• Бармени: прізвище в нижньому регістрі\n"
                             "• Адміністратори: прізвище в нижньому
регістрі");
}

void MainWindow::showAbout()
{
    qDebug() << "Показ інформації про програму...";

    QString aboutText = QString(
        "<h2>%1</h2>"
```

```
        "<p><b>Версія:</b> %2</p>"
        "<p><b>Організація:</b> %3</p>"
        "<br>"
        "<p>Система для автоматизації прийому та обробки замовлень у ресторані.</p>"
        "<br>"
        "<p><b>Основні можливості:</b></p>"
        "<ul>"
        "<li>Управління замовленнями</li>"
        "<li>Контроль меню та цін</li>"
        "<li>Розмежування прав персоналу</li>"
        "<li>Відстеження статусів замовлень</li>"
        "</ul>"
        "<br>"
        "<p><b>Технології:</b></p>"
        "<ul>"
        "<li>Qt %4 Framework</li>"
        "<li>C++17</li>"
        "<li>SQLite / PostgreSQL</li>"
        "</ul>"
        "<br>"
        "<p><b>Підтримка:</b> support@staryi-mlyn.ua</p>"
        "<p><b>Сайт:</b>  <a  href='https://staryi-mlyn.ua'>https://staryi-mlyn.ua</a></p>"
        "<br>"
        "<p>© 2024 ТОВ \"Ресторан Старий Млин\". Всі права захищені.</p>"
    ).arg(QApplication::applicationDisplayName())
     .arg(QApplication::applicationVersion())
     .arg(QApplication::organizationName())
     .arg(QT_VERSION_STR);

    QMessageBox aboutBox(this);
    aboutBox.setWindowTitle("Про програму");
    aboutBox.setTextFormat(Qt::RichText);
```

```
    aboutBox.setText(aboutText);
    aboutBox.setIconPixmap(QIcon(":/icons/app_icon.png").pixmap(64, 64));
    aboutBox.setStandardButtons(QMessageBox::Ok);

    // Додавання кнопки "Про Qt"
    QPushButton* aboutQtButton = aboutBox.addButton("Про Qt",
QMessageBox::ActionRole);
    connect(aboutQtButton, &QPushButton::clicked, qApp,
&QApplication::aboutQt);

    aboutBox.exec();
}

void MainWindow::updateCurrentTime()
{
    // Оновлення поточного часу в статус-барі
    QString currentTime = QDateTime::currentDateTime().toString("dd.MM.yyyy
hh:mm:ss");
    m_timeLabel->setText(currentTime);
}

void MainWindow::initializeDatabase()
{
    qDebug() << "Перевірка стану бази даних...";

    // Ініціалізація вже виконана в main.cpp
    // Тут просто перевіряємо стан підключення
    if (!DatabaseManager::instance()->isConnected()) {
        QString errorMsg = "База даних не підключена!\n\n"
                           "Перевірте налаштування підключення та
перезапустіть програму.";

        QMessageBox::critical(this, "Помилка бази даних", errorMsg);
        qCritical() << "База даних не підключена при ініціалізації
```

```
MainWindow";
        QApplication::quit();
        return;
    }

    qInfo() << "База даних готова до роботи";
}

void MainWindow::onDatabaseError(const QString& error)
{
    qWarning() << "Помилка бази даних:" << error;

    // Показ попередження користувачу
    QMessageBox::warning(this, "Помилка бази даних",
                         "Виникла помилка при роботі з базою даних:\n\n" +
error +
                         "\n\nДеякі функції можуть працювати некоректно.");

    // Оновлення статус-бару
    m_statusLabel->setText("Помилка    БД:    "    +    error.left(50)    +
(error.length() > 50 ? "..." : ""));
}

void MainWindow::closeEvent(QCloseEvent *event)
{
    qDebug() << "Обробка події закриття вікна...";

    if (AuthManager::instance()->isLoggedIn()) {
        int ret = QMessageBox::question(this, "Підтвердження виходу",
                                        "Ви     дійсно     хочете     закрити
програму?",
                                        QMessageBox::Yes | QMessageBox::No,
                                        QMessageBox::No);
```

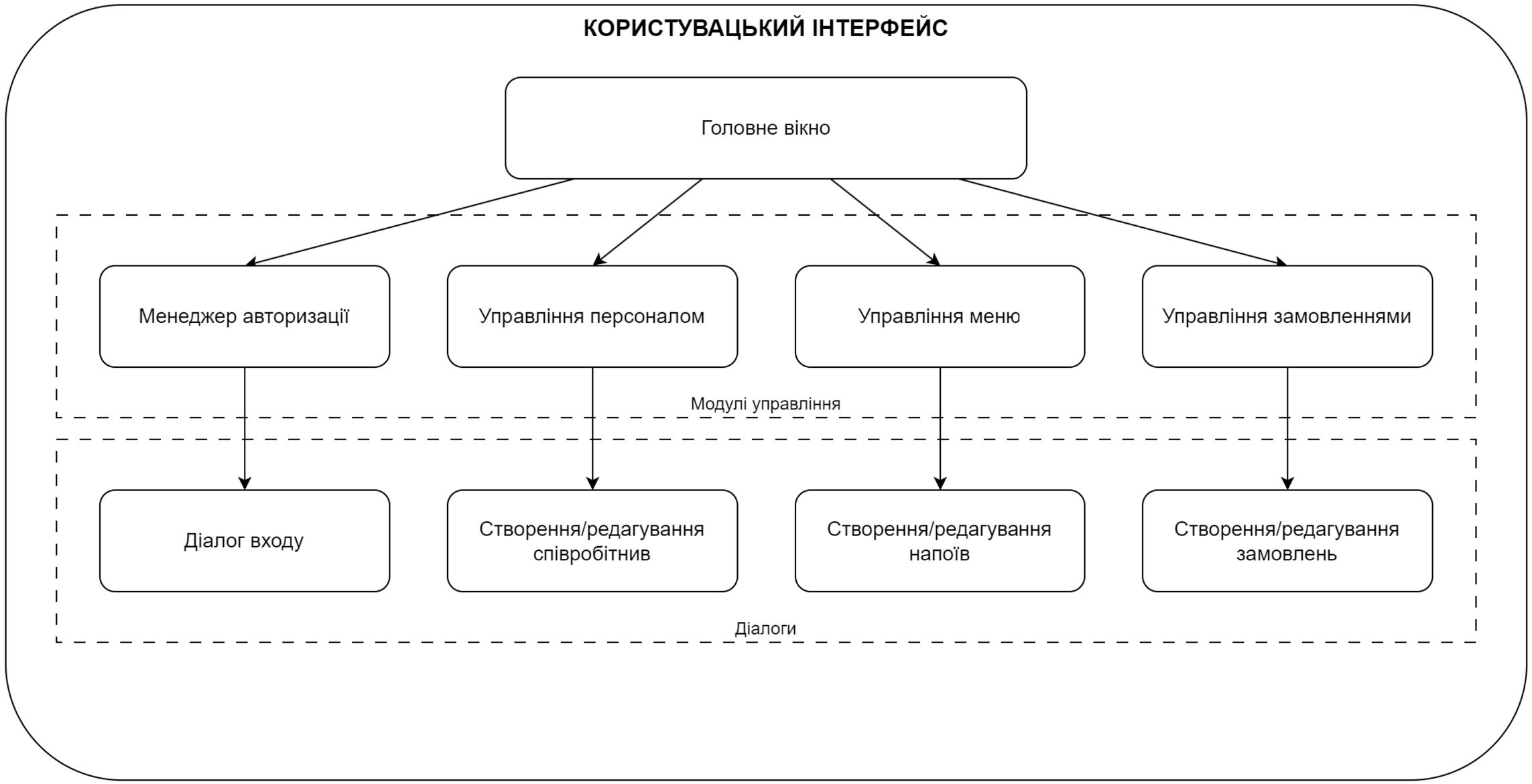
```
        if (ret == QMessageBox::Yes) {
            qInfo() << "Користувач підтвердив закриття програми";

            // Вихід з системи
            AuthManager::instance()->logout();

            // Збереження налаштувань вікна
            // (може бути додано в майбутніх версіях)

            event->accept();
        } else {
            qDebug() << "Закриття програми скасовано";
            event->ignore();
        }
    } else {
        qInfo() << "Закриття програми без активного користувача";
        event->accept();
    }
}
```



					2025.КРБ.123.602.20.00.00 ДВ	
Зм.	Арк.	№ докум.	Підп.	Дата	Розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»	Літ. / Маса / Масштаб
Розроб.		Новроцький С.З.				
Перев.		Лиж В.С.			UML-діаграма варіантів використання	Аркуш 1 / Аркушів 1
Т.контр.						ВСП ТФК ТНТУ КІБ-602
Рецензент						м. Тернопіль
Н.контр.		Приймак В.А.				
Затв.						

Копіював Формат А1

Перш. застос. | Справ. № | Підп. і дата | Інв. № дубл. | Взам. інв. № | Підп. і дата | Інв. № подл.



Зм.	Арк.	№ докум.	Підп.	Дата	2025.КРБ.123.602.20.00.00 СС		
Розроб.		Новроцький С.З.			Розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»	Літ.	Маса	Масштаб
Перев.		Пиж В.С.						
Т.контр.					Схема структурна програмного забезпечення	Аркуш	Аркушів 1	
Рецензент						ВСП ТФК ТНТУ КІБ-602 м. Тернопіль		
Н.контр.		Приймак В.А.						
Затв.								

Копіював Формат А1

Перш. застос.

Довід. №

Підп. і дата

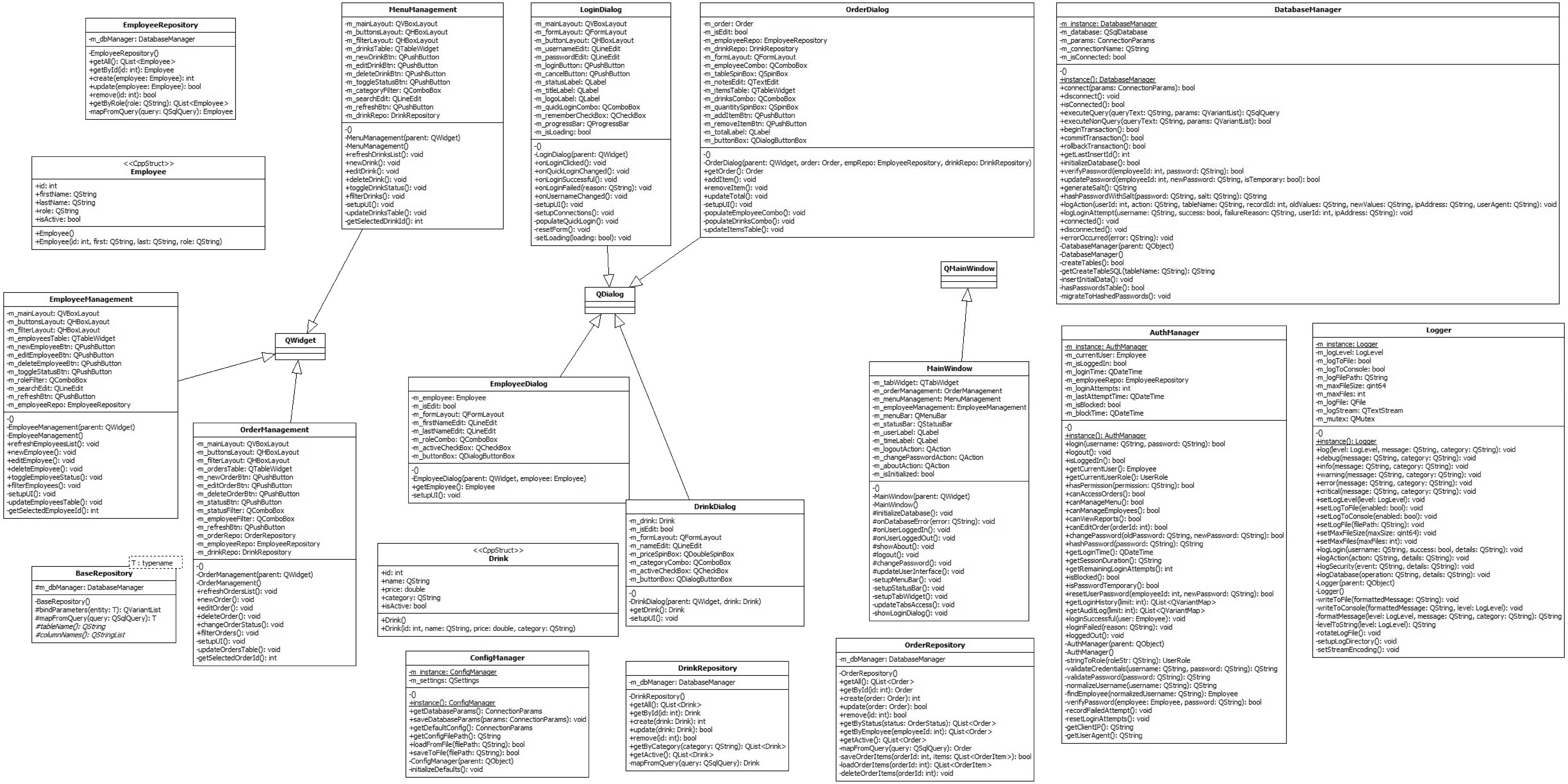
Інв. № дубл.

Взам. інв. №

Підп. і дата

Інв. № орігін.

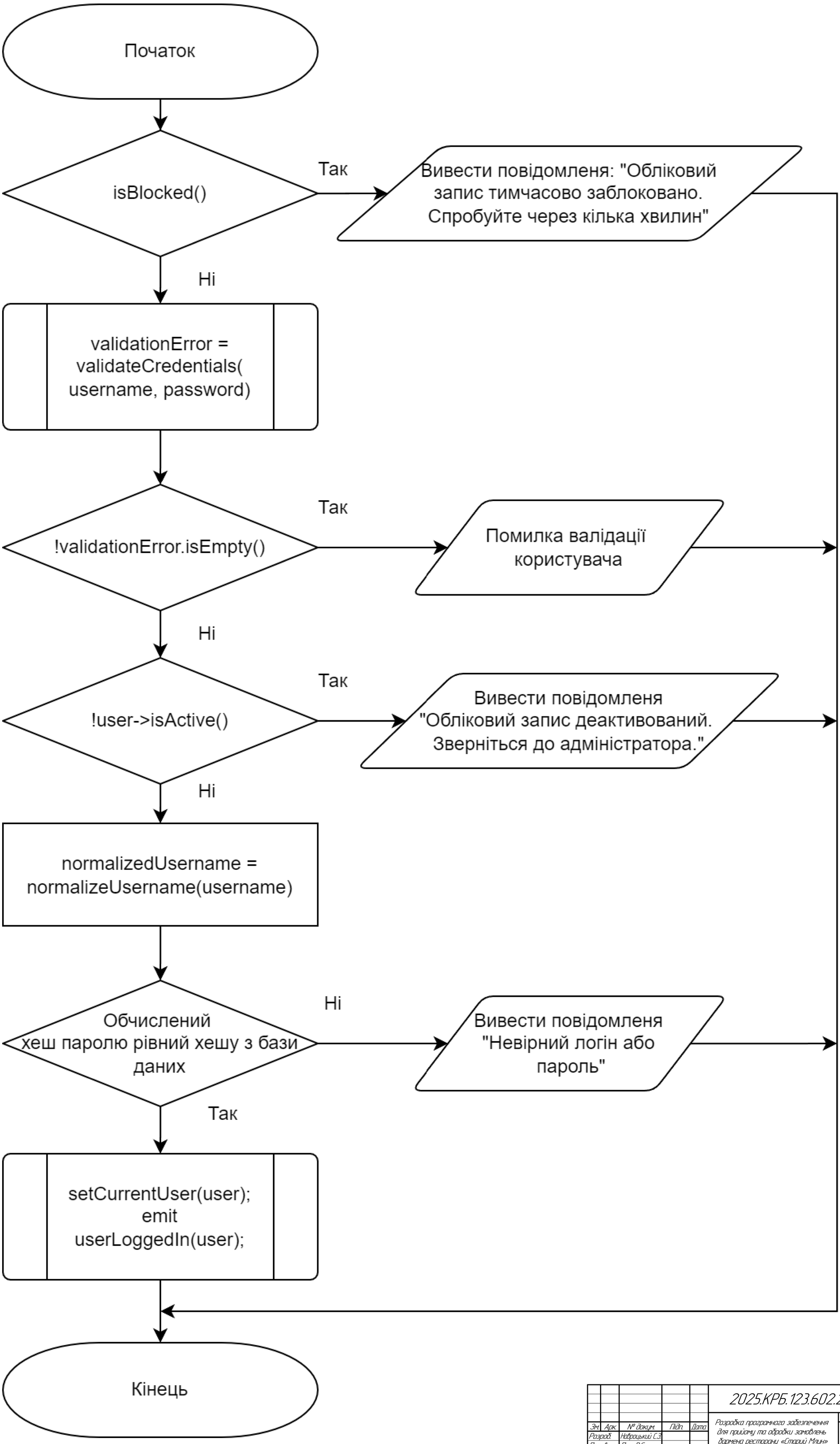
2025.КРБ.123.602.20.00.00 ДК



Перв. застос.
Справ. №
Підп. і дата
Інв. № дубл.
Взам. інв. №
Підп. і дата
Інв. № оригін.

Зм.	Арк.	№ докум.	Підп.	Дата	2025.КРБ.123.602.20.00.00 ДК
Розроб.		Навроцький С.З.			Розробка програмного забезпечення для прийому та обробки замовлень бармена ресторану «Старий Млин»
Перев.		Пиж В.С.			UML-діаграма класів програмного забезпечення
Т.контр.					Літ. / Маса / Масштаб
Рецензент					Аркуш / Аркушів 1
Н.контр.		Приймак В.А.			ВСП ТФК ТНТУ КІб-602
Затв.					м. Тернопіль

Копіював Формат А1



Зм.	Арк.	№ докум.	Підп.	Дата	2025.КРБ.123.602.20.00.00 БС	
Розроб.		Навроцький С.З.			Розробка програмного забезпечення для приймоу та обробки замовлень бармена ресторану «Старий Млин»	Літ. / Маса / Масштаб
Перев.		Пиж В.С.			Блок-схема алгоритму роботи функції	Аркуш 1 / Аркушів 1
Т.контр.						ВСП ТФК ТНТУ КІБ-602 м. Тернопіль
Рецензент						
Н.контр.		Приймак В.А.				
Затв.						

Копіював

Формат А1

Перш. застос.

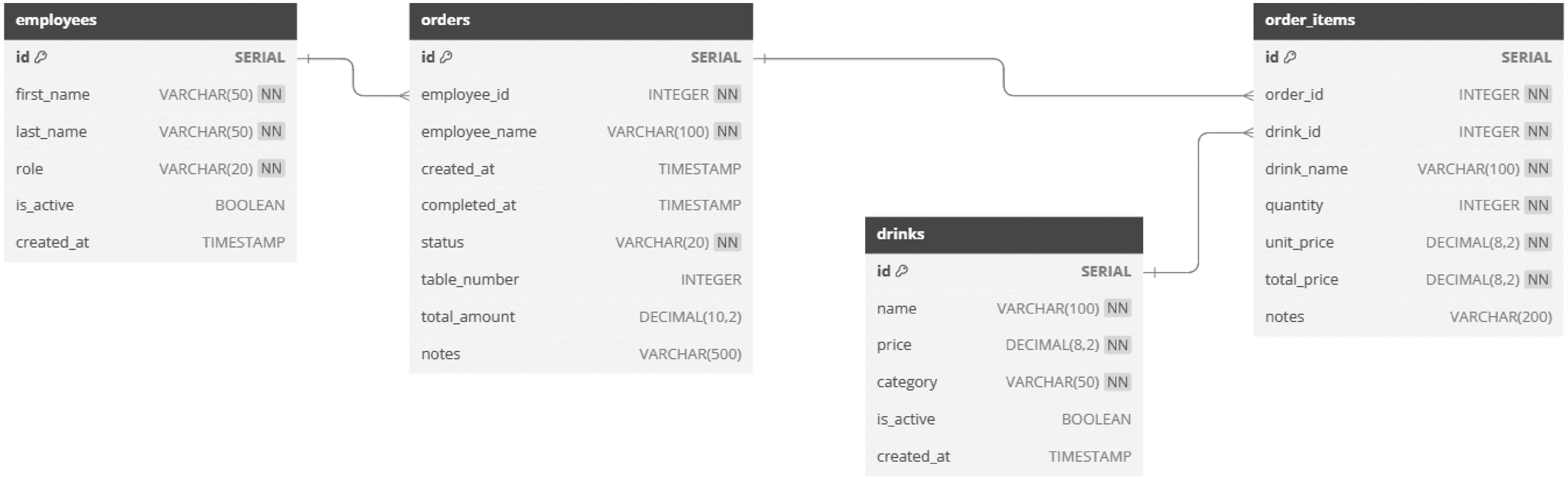
Справ. №

Інв. № дубл.

Взам. інв. №

Підп. і дата

Інв. № оригін.



Зм.	Арк.	№ докум.	Підп.	Дата	2025.КРБ.123.602.20.00.00 БД		
Розроб.		Набровський С.З.			Розробка програмного забезпечення для приому та обробки замовлень бармена ресторану «Старий Млин»	Літ.	Маса	Масштаб
Перев.		Пиж В.С.						
Т.контр.					ER-діаграма бази даних	Аркуш	Аркушів 1	
Реценз.						ВСП ТФК ТНТУ КІБ-602		
Н.контр.		Прийомк В.А.				м. Тернопіль		
Затв.								

Копіював Формат А1

Таблиця техніко-економічних показників

№ п/п	Показник	Одиниці вимірювання	Значення
1	Мова програмування, фреймворк	–	C++, Qt v6.8.3
2	База даних	–	PostgreSQL v17
3	Середовище розробки	–	Qt Creator v16
4	Встановлення	–	Інсталяційний файл
5	Захист	–	Засобами ОС
6	Собівартість	грн.	46590,19
7	Плановий прибуток	грн.	26090,51
8	Ціна	грн.	72680,70
9	Економічна ефективність	грн.	0,56
10	Термін окупності	рік	1,8