Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

**Бакалаврська кваліфікаційна робота**

на тему: Розробка програмного застосунку для адміністрування пункту оренди туристичного спорядження

Виконав: студент IV курсу
групи 1ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Матюха Олег Віталійович
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)
Рецензент: д.ф., ст. викл. каф. ОТ Обертюх М.Р.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ д.т.н. проф. О.Н. Романюк
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної
інженерії Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач
кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

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

Матюсі Олегу Віталійовичу

1. Тема роботи – розробка програмного застосунку для адміністрування пункту оренди туристичного спорядження.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «20» березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025

3. Вихідні дані до роботи: Список актуального спорядження, інформація про замовлення, оренду та клієнтів

4. Зміст текстової частини: аналіз предметної області, проєктування структури даних для обліку інвентарю, клієнтів та замовлень, побудова інтерфейсу користувача засобами WinAPI, розробку загального алгоритму роботи застосунку, а також алгоритмів обробки оренди та повернення спорядження. Структуру внутрішньої бази даних і базові принципи адміністрування. Загальна економічна оцінка ефективності впровадження програмного продукту

5. Перелік ілюстративного матеріалу: структура застосунку для оренди

туристичного спорядження, приклади форм обліку інвентарю, схему бази даних, етапи обробки замовлення, діаграму алгоритму, приклади інтерфейсу користувача та таблицю з економічним порівнянням ручного і автоматизованого обліку.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Ракитянська Г.Б., к.т.н., доц. кафедри ПЗ | 24.03.25 | 01.06.25 |

7. Дата видачі завдання 24 березня 2025

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області та визначення вимог до програмного забезпечення | 25.03.2025 – 05.04.2025 | вик. |
| 2 | Проєктування структури бази даних та моделі обліку інвентарю | 06.04.2025 – 17.04.2025 | вик. |
| 3 | Розробка користувацького інтерфейсу засобами WinAPI | 18.04.2025 – 01.05.2025 | вик. |
| 4 | Реалізація алгоритмів обробки оренди, повернення спорядження та формування звітів | 02.05.2025 – 13.05.2025 | вик. |
| 5 | Оформлення матеріалів до захисту БКР | 14.05.2025 – 30.05.2025 | вик. |

**Студент** _____ **О.В. Матюха**
( підпис ) (ініціали та прізвище)

**Керівник бакалаврської кваліфікаційної роботи** _____ **Г.Б. Ракитянська**
( підпис ) (ініціали та прізвище)

# АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається зі 107 сторінок формату А4, на яких є 42 рисунки, 2 таблиці, список використаних джерел містить 22 найменування.

У бакалаврській кваліфікаційній роботі проведено аналіз стану програмних застосунків для адміністрування пунктів оренди туристичного спорядження та постановку задач дослідження. Було проведено порівняльний аналіз аналогів програного застосунку, визначено їх переваги та недоліки та доведено доцільність розробки програмного застосунку.

Під час виконання роботи було створено програмний застосунок, що дозволяє адміністраторам пунктів оренди туристичного спорядження швидко та зручно організовувати здачу в оренду певних одиниць спорядження.

Програмний застосунок розроблено з використанням мови програмування C++ із застосуванням WinAPI для створення графічного інтерфейсу. Розроблені модулі можуть бути використані в туристичних центрах, базах активного відпочинку, а також в організаціях, що надають послуги оренди туристичного спорядження.

Отримані в бакалаврській кваліфікаційній роботі результати можуть бути використані компаніями для покращення процесу адміністрування пунктів оренди.

Ключові слова: оренда, інвентаризація, автоматизація, програмний застосунок, адміністрування, туристичне спорядження.

# ABSTRACT

The bachelor's thesis consists of 107 pages of A4 format, which include 42 figures, 2 tables, and the list of sources used contains 22 names.

The bachelor's thesis analyzes the state of software applications for administering tourist equipment rental points and sets research objectives. A comparative analysis of analogues of the lost application was conducted, their advantages and disadvantages were identified, and the feasibility of developing a software application was proven.

During the work, a software application was created that allows administrators of tourist equipment rental points to quickly and conveniently organize the rental of certain units of equipment.

The software application was developed using the C++ programming language using WinAPI to create a graphical interface. The developed modules can be used in tourist centers, active recreation centers, as well as in organizations that provide tourist equipment rental services.

The results obtained in the bachelor's thesis can be used by companies to improve the process of administering rental points.



Keywords: rental, inventory, automation, software application, administration, tourist equipment.

ЗМІСТ

# ВСТУП

**Обґрунтування вибору теми дослідження.** У сучасних умовах цифровізації дедалі більше процесів в організаціях переходить у цифрове середовище, що зумовлює зростаючу потребу в автоматизації різних сфер діяльності, зокрема у сфері обліку ресурсів, взаємодії з клієнтами та управління внутрішніми процесами [1]. Особливо гостро ця потреба проявляється в організаціях, що надають послуги з тимчасового користування майном, таких як пункти оренди туристичного спорядження. Такі пункти щодня обслуговують значну кількість клієнтів, оперують широким асортиментом інвентарю, контролюють його технічний стан, терміни повернення, а також здійснюють фінансові операції. У подібних умовах виникає потреба в системному підході до управління всіма цими аспектами.

На практиці часто застосовуються застарілі методи ведення обліку - наприклад, паперові журнали або електронні таблиці. Проте вони не забезпечують необхідного рівня точності, оперативності та безпеки даних. Ручне введення інформації є трудомістким, а відсутність централізованої бази даних ускладнює контроль за актуальністю інформації. У результаті знижується ефективність роботи персоналу, виникають затримки в обслуговуванні клієнтів, що негативно впливає на репутацію організації.

У зв’язку з цим надзвичайно актуальним є створення спеціалізованого програмного забезпечення, яке дозволить комплексно автоматизувати діяльність пунктів оренди туристичного спорядження [2]. Такий застосунок має забезпечити зручний інтерфейс для введення та обробки даних, функціональність для управління замовленнями, контролю термінів повернення інвентарю, ведення клієнтської бази, формування звітності. Впровадження подібного програмного продукту сприятиме підвищенню ефективності управління, зменшенню адміністративних витрат, мінімізації ризиків людського фактору та покращенню якості обслуговування клієнтів.

**Зв’язок роботи з науковими програмами, планами, темами.** Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

**Мета та завдання дослідження.** Метою роботи є розробка програмного застосунку [3] для ефективного адміністрування пункту оренди туристичного спорядження, що забезпечує облік інвентарю, обробку замовлень, ведення клієнтської бази, прогнозування попиту та автоматизоване управління поверненням спорядження.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати предметну область та визначити вимоги до функціональності системи;

– розробити структуру бази даних, що відображає основні сутності орендного процесу;

– реалізувати графічний інтерфейс користувача засобами WinAPI;

– створити модулі для обробки замовлень, формування звітності;

– протестувати роботу застосунку в умовах моделювання реальної експлуатації;

– виконати економічне обґрунтування доцільності використання програмного продукту.

**Об’єкт дослідження** – процес розробки програмного застосунку для автоматизації обліку та управління орендою туристичного спорядження.

**Предмет дослідження** – методи і засоби розробки програмного застосунку для адміністрування пункту оренди туристичного спорядження.

**Методи дослідження.** У роботі застосовано методи структурного аналізу, об’єктно-орієнтованого проєктування, побудови баз даних, а також програмування інтерфейсів за допомогою API Windows [4].

**Новизна отриманих результатів.**

1. Розроблено метод прогнозування попиту на туристичне спорядження, що дозволяє підвищити ефективність адміністрування за рахунок аналізу часових рядів.

2. Розроблено прикладне програмне рішення для автоматизації процесів оренди, адаптоване до специфіки пунктів прокату туристичного спорядження. Запропоновано спрощену модель обліку з використанням внутрішньої файлової бази та реалізацією інтерфейсу на базі Windows API без залучення сторонніх фреймворків, що забезпечує легку підтримку та запуск на слабких системах.

**Практична цінність отриманих результатів.** Розроблений програмний продукт може бути впроваджений у малих підприємствах та туристичних центрах для полегшення ведення обліку, та підвищення загальної ефективності роботи пункту оренди.

**Особистий внесок здобувача.** Отримані результати дослідження та розробки представлені автором особисто. Зокрема, у тезах доповіді автор представляє аналіз відомих рішень.

**Апробація результатів** здійснена на міжнародній науково-практичній інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ. МН-2025», (Вінниця, 2025)

**Публікації.** Тези доповідей представлені в збірнику міжнародної науково-практичної інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ. МН-2025», (Вінниця, 2025) [22].

**Структура та обсяг БКР.** БКР складається зі вступу, чотирьох розділів, висновків, списку літератури що містить 22 найменувань, 4 додатків. Робота містить 42 ілюстрації, 2 таблиці. Загальний обсяг роботи складає 107 сторінок, основний зміст викладено на 54 сторінках друкованого тексту.

# 1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

## 1.1 Аналіз стану програмних застосунків для адміністрування пункту оренди туристичного спорядження

Сучасний етап розвитку цифрових технологій зумовлює необхідність автоматизації бізнес-процесів у різноманітних сферах, зокрема в галузі надання послуг оренди туристичного спорядження. Зростання попиту на туристичні послуги, активний відпочинок та екологічний туризм призводить до збільшення кількості пунктів прокату. У цих умовах важливо забезпечити ефективне управління ресурсами, облік інвентарю та взаємодію з клієнтами, що практично неможливо реалізувати за допомогою паперових журналів або неуніверсальних електронних таблиць.

На сьогодні на ринку існує низка програмних продуктів, що частково вирішують задачу автоматизації прокатних сервісів. Серед них — платформи загального призначення для ведення обліку (наприклад, Excel, Google Sheets), CRM-системи для малого бізнесу (Bitrix24, Zoho CRM), а також спеціалізовані сервіси для орендного бізнесу (Rentle, Booqable, Rentman). Однак ці рішення мають ряд суттєвих обмежень: складність налаштування під специфіку туристичного спорядження, висока вартість ліцензій, залежність від постійного інтернет-з’єднання, відсутність локалізації українською мовою та обмежені можливості кастомізації [5].

Крім того, більшість існуючих систем не підтримують гнучке управління обліком різнотипного спорядження (наприклад, спорядження зі змінними характеристиками або змінними комплектуючими), а також не враховують потребу в контролі термінів повернення та автоматичному відстеженні наявності на складі. У багатьох випадках користувачі змушені або адаптувати наявні засоби під свої потреби, або зовсім відмовляються від автоматизації, що призводить до низької ефективності та втрат часу.

Ситуація ускладнюється і тим, що більшість існуючих рішень орієнтовані на великі компанії та включають надлишкову функціональність, яка не є необхідною для невеликих пунктів прокату. Як наслідок, такі системи стають складними у використанні для непідготовленого персоналу, нераціональними у витратах та неефективними в умовах змінного попиту.

У контексті цієї проблеми актуальною є розробка спеціалізованого програмного забезпечення, орієнтованого саме на потреби малих пунктів оренди туристичного спорядження. Така система повинна бути простою в освоєнні, мати інтуїтивно зрозумілий інтерфейс, дозволяти швидке введення, редагування та перегляд інформації про інвентар, клієнтів і замовлення. Бажано, щоб система працювала в офлайн-режимі, зберігаючи дані локально, а також підтримувала можливість подальшого розширення функціоналу.

Враховуючи актуальні вимоги та існуючі обмеження, створення спеціалізованої системи для адміністрування пункту оренди туристичного спорядження є важливим кроком для забезпечення якісного сервісу. Це дозволить автоматизувати ключові операції, знизити ризики людських помилок, підвищити обслуговування клієнтів та ефективність управління ресурсами. Крім того, використання локальних, кастомізованих рішень забезпечує незалежність від зовнішніх платформ та кращу адаптацію до конкретних умов роботи підприємства.

## 1.2 Порівняльний аналіз аналогів

У сучасних умовах розвитку сфери туризму та активного відпочинку питання ефективного адміністрування пунктів оренди туристичного спорядження стає надзвичайно актуальним. Автоматизація обліку, контролю наявності інвентарю, оформлення замовлень та взаємодії з клієнтами є ключовими аспектами, які потребують впровадження програмних рішень. На ринку існує ряд програмних засобів, що частково або повністю реалізують подібний функціонал, проте жоден із них не орієнтований безпосередньо на специфіку оренди туристичного спорядження. До найбільш відомих рішень

відносять:

- Booqable;
- Rentle;
- OpenRMS;
- Excel;
- Zoho Inventory

Одним із прикладів є Booqable — це SaaS-рішення (рис. 1.1) для управління орендою інвентарю, популярне серед компаній, що надають обладнання на прокат [6].

Рисунок 1.1 – Приклад роботи застосунку Booqable

Програма працює через вебінтерфейс та надає наступний функціонал:

- створення онлайн-вітрини оренди;
- управління замовленнями та платежами;
- автоматичні нагадування про повернення спорядження;
- інтеграція з платіжними системами.

Переваги:

- простий та сучасний інтерфейс;

- підходить для малого та середнього бізнесу;
- наявність мобільної версії.

Недоліки:

- відсутність української мови;
- потрібен постійний доступ до Інтернету;
- висока абонплата, що робить сервіс малодоступним для бюджетних організацій.

Інший приклад – застосунок Rentle — ще одна SaaS-платформа (рис.1.2), орієнтована на оренду спорядження, включаючи туристичне. Система дозволяє створити віртуальну крамницю, де клієнти можуть здійснювати бронювання [7].

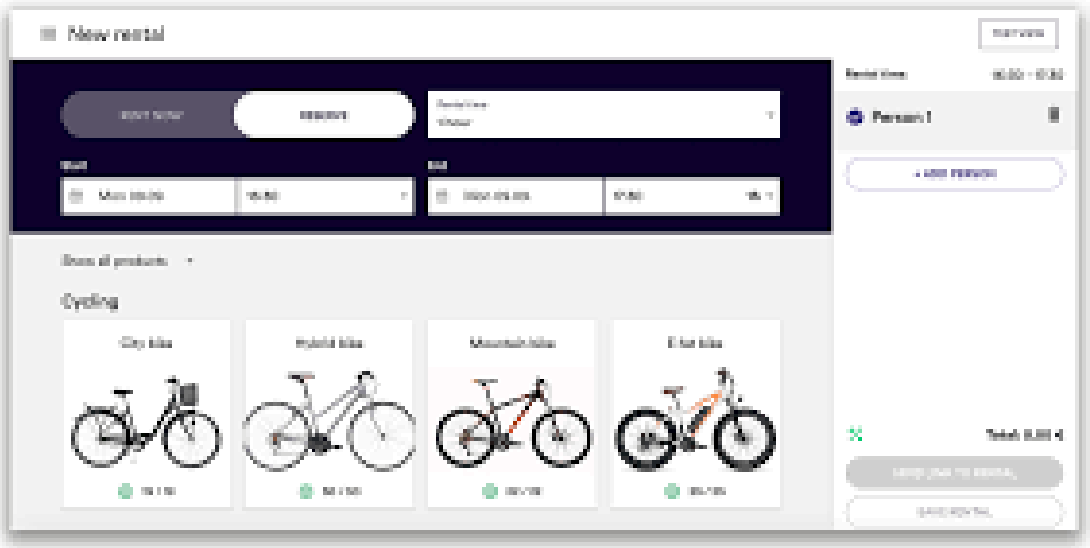


Рисунок 1.2 – Приклад роботи застосунка Rentle

Функціонал:

- керування замовленнями та наявністю спорядження;
- онлайн-оплати;
- інтеграція з сайтом компанії;
- базова аналітика.

Переваги:

- хороша підтримка мобільних пристроїв;
- зручна система оформлення замовлень;
- можливість інтеграції з вебсайтом компанії.

Недоліки:

- відсутність офлайн-режиму;
- лише англомовний інтерфейс;
- менші можливості кастомізації під нестандартні види спорядження;
- обмеження у безкоштовній версії.

Ще одним прикладом є OpenRMS — це система (рис. 1.3) з відкритим кодом, яка використовується для управління прокатом ресурсів. Спочатку була створена для оренди книг, техніки тощо, але її можна адаптувати під прокат спорядження [8].

Рисунок 1.3 – Приклад роботи застосунку OpenRMS

Переваги:

- безкоштовна та доступна для модифікації;

- відкрита структура бази даних;
- немає обмежень на кількість клієнтів чи інвентарю.

Недоліки:

- складна для початкового налаштування;
- не має графічно зручного інтерфейсу;
- потрібна глибока технічна підготовка для налаштування і розширення;
- відсутність специфічних функцій для туристичного інвентарю (комплекти, стан зносу тощо).

Excel (рис.1.4) – найпростіше рішення, яке досі використовується багатьма малими пунктами прокату — електронні таблиці [9].

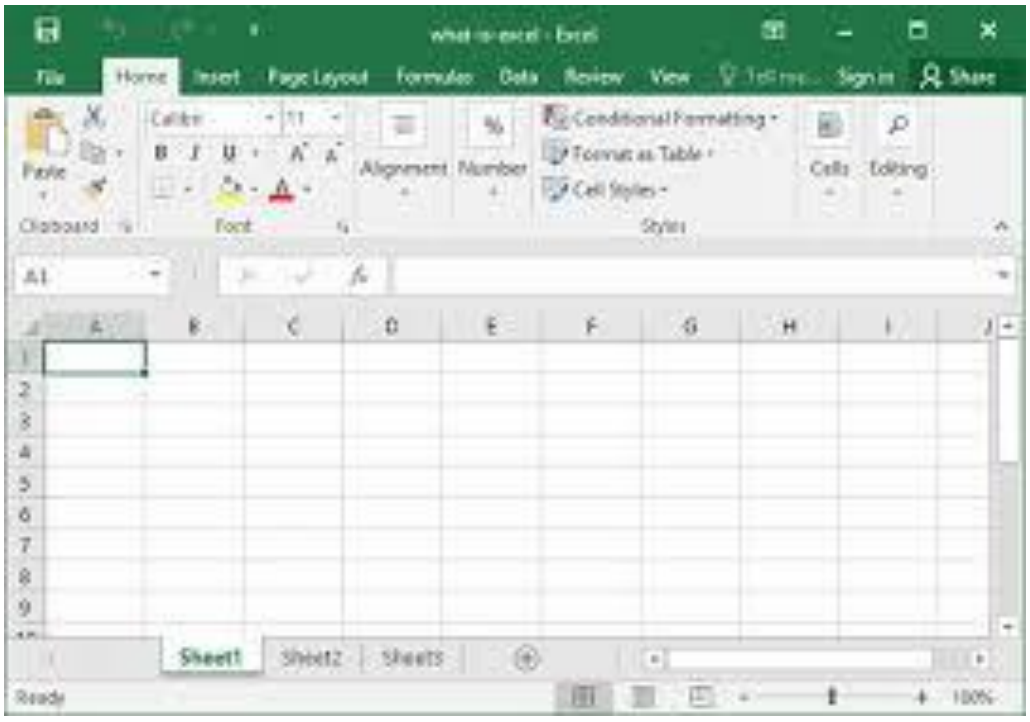


Рисунок 1.4 – Приклад роботи застосунку Excel

Переваги:

- безкоштовність;
- простота;
- доступність без спеціального програмного забезпечення.

Недоліки:

- відсутність автоматизації;
- висока ймовірність людських помилок;
- відсутність зручної візуалізації чи пошуку;
- не підтримує багатокористувацький режим без конфліктів;
- неможливо інтегрувати оплату, історію оренди, нагадування та інші бізнес-функції.

І ще одним прикладом є Zoho Inventory (рис. 1.5) – це складна CRM-система, яка має модулі управління запасами і може частково використовуватись для оренди [10].


Рисунок 1.5 – Приклад роботи застосунку Zoho Inventory

Переваги:

- розвинена система аналітики;
- підтримка великої кількості користувачів;
- інтеграція з іншими бізнес-процесами (оплата, клієнтська база, документообіг).

Недоліки:

- не адаптована під оренду, особливо туристичного спорядження;
- потребує часу на налаштування;
- складний інтерфейс для новачків;
- частина функцій доступна лише у платних версіях.

У межах бакалаврської кваліфікаційної роботи розробляється спеціалізоване програмне забезпечення, яке буде включати функції, орієнтовані саме на пункт оренди туристичного спорядження:

Функціональні можливості:

- облік інвентарю з можливістю формування комплектів;
- ведення історії оренди та повернення;
- робота в офлайн-режимі;
- інтуїтивно зрозумілий графічний інтерфейс;
- локалізація українською мовою;
- гнучке налаштування типів спорядження та умов оренди.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

| Критерії | Booqable | Rentle | OpenRMS | Excel | Zoho Inventory | Власна розробка |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Автоматизація процесів оренди | + | + | - | - | - | + |
| Підтримка обліку спорядження | + | + | + | - | - | + |

Продовження таблиці 1.1

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Зручність інтерфейсу | + | - | - | - | + | + |
| Можливість офлайн-роботи | - | - | + | + | - | + |
| Наявність української локалізації | - | - | - | + | + | + |
| Можливість кастомізації | - | - | + | + | - | + |
| Орієнтованість на туристичне спорядження | - | - | + | - | - | + |
| Загальна оцінка | 70% | 60% | 50% | 40% | 55% | 100% |

Як видно з порівняльного аналізу, жодне із розглянутих рішень не надає повноцінної підтримки специфіки роботи пунктів оренди туристичного спорядження. Найчастіше зустрічаються обмеження у вигляді відсутності можливості роботи без підключення до інтернету, складного налаштування, надмірної універсальності або платного доступу до основного функціоналу.

Розробка власного програмного застосунку вирішує ці проблеми завдяки орієнтації саме на специфіку туристичного прокату: облік спорядження з комплектуючими, ведення історії оренди, контроль за термінами повернення, гнучке налаштування видів інвентарю, зручний графічний інтерфейс. Крім того, впровадження офлайн-режиму роботи дозволяє використовувати програму навіть у віддалених пунктах прокату, де немає постійного доступу до

мережі.

Отже, проведений аналіз підтверджує доцільність розробки власного спеціалізованого програмного забезпечення для адміністрування пункту оренди туристичного спорядження. Такий підхід забезпечує адаптацію під конкретні потреби організації, спрощує роботу персоналу та підвищує ефективність обслуговування клієнтів.

### 1.3 Аналіз методів розв’язання задачі та постановка задач

Розробка програмного забезпечення для адміністрування пункту оренди туристичного спорядження передбачає вирішення ряду завдань, серед яких: ефективне ведення обліку спорядження, автоматизація процесу видачі та повернення, формування історії оренди, управління клієнтськими замовленнями, забезпечення контролю зношеності інвентарю тощо. Для реалізації подібної системи можна використовувати різноманітні методи та підходи, кожному з яких притаманні певні переваги та недоліки.

Моделювання предметної області на основі реляційної бази даних

Найбільш поширеним і ефективним підходом до реалізації облікової системи є побудова реляційної моделі даних. Такий підхід дозволяє представити інвентар, комплекти, клієнтів, замовлення та операції оренди у вигляді взаємопов’язаних таблиць.

Переваги:

- забезпечення цілісності та зв'язності даних;
- зручність реалізації фільтрації, сортування, пошуку;
- можливість масштабування структури БД.

Недоліки:

- потреба в чіткому проєктуванні логіки ще до реалізації;
- складність модифікацій при слабкій нормалізації.

У межах даної роботи передбачається побудова БД з такими основними сутностями: Спорядження, Комплекти, Клієнти, Операції оренди, Статуси,

Історія використання. Реляційна модель дозволяє ефективно реалізувати всі основні функції системи.

Організація обліку комплектного спорядження

Одним із важливих аспектів при створенні системи є підтримка комплектів спорядження (наприклад, набір для кемпінгу: намет, спальник, килимок тощо). Для цього необхідно реалізувати метод ієрархічного групування спорядження.

Метод: використання проміжної таблиці (наприклад, Комплекти_Спорядження), яка дозволяє створювати довільні набори з окремих одиниць.

Переваги:

- можливість повторного використання одиниць у різних комплектах;
- динамічна побудова структури без дублювання даних.

Недоліки:

- ускладнення логіки перевірки наявності;
- додаткова перевірка стану кожного елементу комплекту при оренді.

Управління станом зношеності

Для реалізації обліку стану спорядження застосовується метод класифікації за категоріями стану (наприклад: "нове", "добрий", "зношене", "непридатне").

Методи реалізації:

- використання статусних полів для одиниці спорядження;
- додавання історії змін стану в окрему таблицю для аудиту.

Переваги:

- контроль якості на кожному етапі використання;
- можливість фіксувати зношення в розрізі клієнтів.

Недоліки:

- необхідність ручного оновлення або періодичного огляду;
- складність автоматизації оновлення стану без датчиків чи фотоаналізу.

Побудова графічного інтерфейсу (GUI)

Для реалізації графічного інтерфейсу на Windows використовується підхід WinAPI. Це дозволяє створити нативне настільне застосування без залежності від браузера чи додаткових платформ [11].

Переваги:

- швидкодія;
- повна інтеграція з ОС Windows;
- доступ до низькорівневих елементів керування.

Недоліки:

- складність розробки та стилізації;
- потреба у більшій кількості коду порівняно з фреймворками на кшталт Qt чи C# WPF.

У межах роботи GUI реалізується з акцентом на зручність: перегляд спорядження у таблицях, додавання замовлень через форми, швидке перемикання між вкладками, відображення доступності.

Формування історії оренди

Хоча система не формує офіційні звіти, але історія оренди реалізується як лог змін — це дозволяє відстежувати, коли і ким було орендовано чи повернено спорядження, а також перевірити, хто був останнім користувачем. Для цього використовується таблиця Історія_оренди з полями: дата видачі, дата повернення, клієнт, спорядження, стан, відповідальний.

Проаналізувавши можливі методи реалізації основних функцій, обрано комбінований підхід: використання реляційної моделі даних для логіки обліку, підтримка ієрархії комплектів, фіксація стану спорядження, а також створення зручного графічного інтерфейсу з використанням WinAPI. Така комбінація дозволяє створити універсальну, гнучку та ефективну систему, адаптовану саме під потреби пункту оренди туристичного спорядження.

На основі проведеного аналізу існуючих аналогів програмного забезпечення для обліку та адміністрування прокатного інвентарю, а також вивчення методів реалізації відповідних функціональних модулів,

сформульовано перелік основних завдань, які необхідно реалізувати в рамках бакалаврської кваліфікаційної роботи [12].

Метою розробки є створення ефективного, зручного та функціонального програмного продукту, що забезпечить автоматизацію процесів в пункті оренди туристичного спорядження. Для досягнення цієї мети передбачено виконання таких задач:

- Розробити структуру реляційної бази даних, яка дозволить зберігати інформацію про клієнтів, спорядження, комплекти, операції оренди та історію використання;
- реалізувати механізм обліку одиниць спорядження, з можливістю відображення стану зношеності, статусу доступності та належності до конкретного комплекту;
- розробити модуль обробки замовлень, що дозволить реєструвати видачу та повернення спорядження, враховуючи терміни оренди, конфлікти бронювання та перевірку наявності;
- забезпечити підтримку обліку комплектного спорядження, із реалізацією взаємозв’язку між окремими одиницями та наборами (комплектами);
- створити інтерфейс користувача, реалізований з використанням WinAPI, який дозволить оператору пункту оренди швидко та зручно взаємодіяти з системою;
- реалізувати можливість ведення історії операцій, з фіксацією даних про всі дії, пов’язані з орендою, поверненням і зміною стану спорядження;
- передбачити перевірку актуальності спорядження, з можливістю фіксації критичних станів і недоступності певних одиниць для повторної видачі;
- провести тестування системи, з метою перевірки коректності роботи основних функцій, надійності збереження даних і зручності користувацької взаємодії.

## 1.4 Висновки

У першому розділі було проведено аналіз актуального стану програмного забезпечення для адміністрування пунктів оренди туристичного спорядження. Розглянуто функціональні можливості наявних аналогів, таких як Rentle, Booqable, Excele, OpenRMs та Zoho Inventory, які забезпечують облік інвентарю, бронювання, управління клієнтами та інтеграцію з іншими сервісами. В ході порівняльного аналізу було виявлено, що жодна з існуючих систем не враховує повною мірою специфіку локальних пунктів оренди, особливо в контексті зношеності спорядження, роботи з комплектами та ведення історії операцій.

Було також проаналізовано методи вирішення задач обліку оренди, серед яких: централізоване зберігання даних, використання уніфікованої структури бази даних для опису інвентарю та клієнтів, автоматизоване керування станом спорядження. Обґрунтовано вибір інструментів і підходів до реалізації системи — зокрема, використання мови C++ та WinAPI для створення зручного настільного додатку.

На основі цього аналізу сформульовано конкретні задачі для реалізації власного програмного рішення. Зокрема, це проєктування бази даних, створення модулів обліку спорядження, бронювання, взаємодії з клієнтами, розробка графічного інтерфейсу та проведення тестування.

Таким чином, було доведено раціональність створення власної системи адміністрування пункту оренди, що відповідає практичним потребам, забезпечує автоматизацію основних операцій та дозволяє ефективно керувати туристичним спорядженням у реальних умовах.

# 2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

## 2.1 Аналіз вхідних даних системи

Одним із найважливіших аспектів проєктування інформаційної системи є чітке визначення вхідних даних, від якості та структури яких залежить точність обліку, коректність роботи функціональних модулів і загальна ефективність функціонування програми. У системі адміністрування пункту оренди туристичного спорядження основні вхідні дані охоплюють кілька груп:

Інформація про туристичне спорядження

Ця категорія даних включає базові характеристики, які дозволяють ідентифікувати, класифікувати та контролювати стан кожної одиниці інвентарю:

- Назва спорядження (наприклад, "Намет Tramp Lair 3");
- Унікальний ідентифікатор (штрих-код або внутрішній ID, наприклад, TRM345);
- Тип (намет, спальник, пальник, килимок тощо);
- Стан (нове, задовільне, потребує ремонту, списане);
- Дата придбання (формат: ДД.ММ.РРРР);
- Рівень зношеності (відсоткове значення або умовна шкала);
- Доступність до оренди (доступне, зарезервоване, видане, недоступне).

Відомості про клієнтів

Цей набір даних використовується для ідентифікації особи, ведення журналу оренд і зв’язку з користувачем:

- ПІБ клієнта;
- Контактні дані (номер телефону, email, за потреби — адреса);
- Історія оренди (дати, спорядження, стан повернення, суми оплат).

Операції оренди

Ця група даних відповідає за фіксацію усіх транзакцій оренди:

- Дата видачі та дата повернення;
- Відповідальна особа (адміністратор, що здійснив видачу);
- Вартість оренди;
- Штрафи (у випадку пошкоджень, втрати або запізнень).

Комплекти спорядження

Для зручності обліку система підтримує логіку групування інвентарю у набори:

- Наприклад, "Набір для кемпінгу" може включати: намет + спальник + килимок.
- Кожен комплект має власний ID і перелік складників.
- Комплекти можуть бути попередньо зібрані або формуватися вручну.

Внутрішні параметри системи

Ці дані використовуються для конфігурації системи та захисту:

- Статуси спорядження (для фільтрації та пошуку);
- Налаштування періодів бронювання (максимальна тривалість оренди, буфер між орендами тощо);
- Облікові дані адміністратора (логін, пароль).

Формат зберігання вхідних даних

У якості бази даних у бакалаврській кваліфікаційній роботі використовується локальний текстовий файл або бінарний файл, що забезпечує:

- автономність роботи програми (не потребує інтернету або серверу);
- простоту реалізації;
- можливість серіалізації/десеріалізації структур C++.

Дані можуть зберігатися у таких форматах:

- .txt – для простого зчитування у вигляді списків чи CSV;
- .dat – бінарне збереження структур для пришвидшеної обробки;
- .json – структуроване зберігання з використанням сторонніх бібліотек, наприклад, nlohmann/json.

Грамотно структуровані вхідні дані є фундаментом для реалізації ефективної логіки обліку, оренди та контролю спорядження. Завдяки

використанню простих форматів зберігання (txt/dat/json) забезпечується швидкодія та зручність розробки, а також можливість масштабування у майбутньому.

### 2.2 Розробка методу прогнозування попиту на туристичне спорядження

Прогнозування попиту є важливим етапом у процесі оптимізації діяльності пункту оренди туристичного спорядження. Раціональне управління запасами та орендними одиницями дозволяє підвищити рівень обслуговування клієнтів, уникнути дефіциту популярного спорядження в пікові періоди та зменшити витрати, пов'язані з простоєм обладнання.

Для реалізації методу прогнозування попиту було обрано підхід на основі історичних даних про оренду. Основною ідеєю є використання часових рядів для аналізу попередніх періодів активності та визначення тенденцій змін попиту.

Основні етапи побудови методу прогнозування:

1. Збір та підготовка даних.

З бази даних системи адміністрування витягуються записи про попередні оренди з вказанням дати, типу спорядження, кількості орендованих одиниць. Дані групуються за днями, тижнями або місяцями в залежності від бажаної точності прогнозу.

2. Попередня обробка.

Видаляються пропуски, аномалії, які можуть спотворити прогноз. Дані нормалізуються для уніфікації одиниць виміру.

3. Вибір моделі прогнозування.

Для задачі прогнозування сезонного попиту обрано метод простої ковзної середньої як базову модель. Її перевагою є простота реалізації та адекватність для обмежених обсягів даних. Результат розраховується за формулами:

- Популярність = (Кількість замовлень цього типу) / (Загальна кількість замовлень)
- Сезонний коефіцієнт = 1.8 (липень) – 0.5 (грудень)

- Прогнозований попит = Популярність × 100 × Сезонність

4. Врахування сезонності.

Для більш точного прогнозу у весняно-літній період (традиційний сезон активного туризму) коефіцієнт корекції попиту збільшується, наприклад, на 20–30%. Це дозволяє пристосувати модель до реальних умов.

5. Реалізація алгоритму.

Метод реалізовано як окремий модуль системи адміністрування, який щотижнево генерує звіт з прогнозованими обсягами попиту по кожному типу спорядження. У майбутньому можливе розширення алгоритму до більш складної моделі, наприклад, регресійної або на базі машинного навчання (наприклад, моделі ARIMA або LSTM).

Також було розроблено алгоритм роботи прогнозування попиту на туристичне спорядження (рис. 2.1).

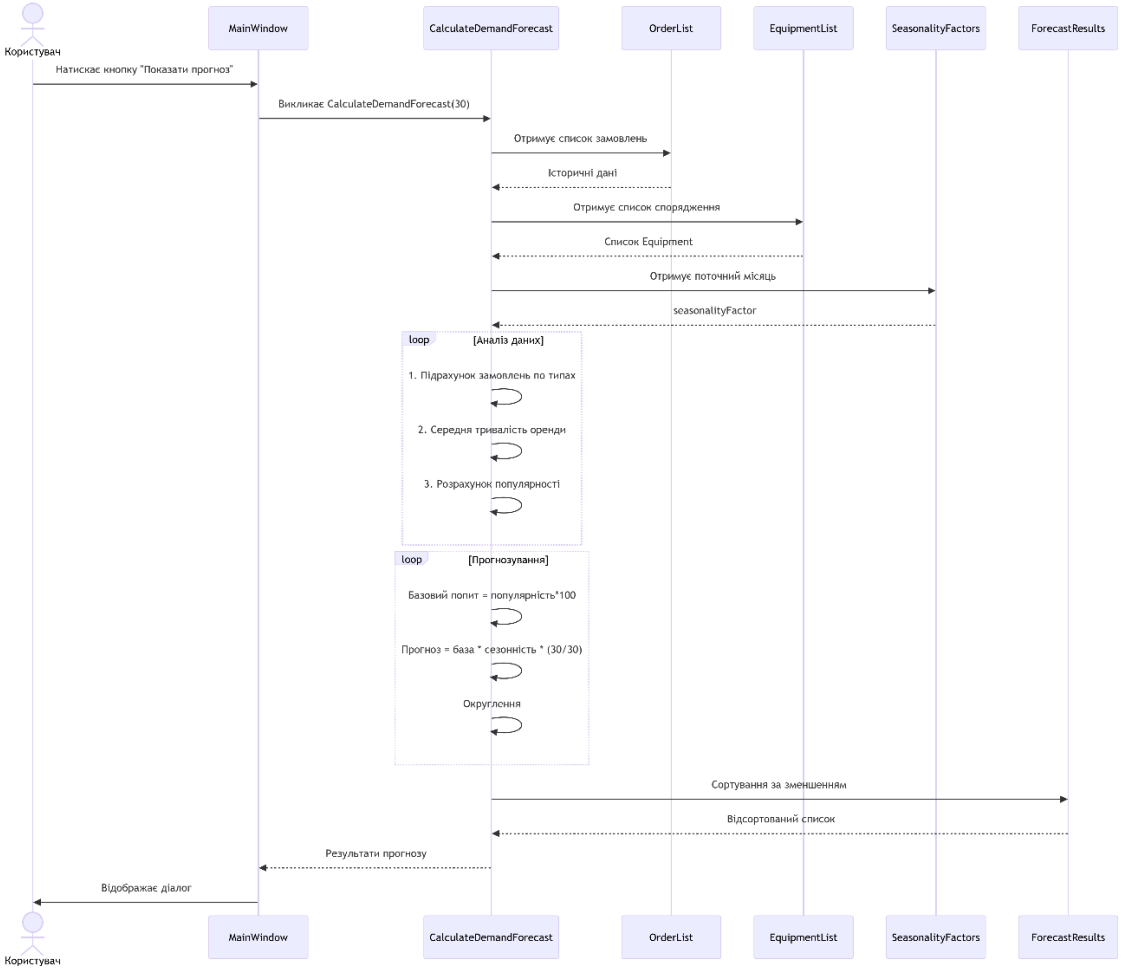


Рисунок 2.1 – Алгоритм роботи прогнозування попиту

Таким чином, впровадження методу прогнозування попиту підвищує ефективність управління пунктом оренди, дозволяє заздалегідь реагувати на зміну попиту та покращує клієнтський досвід.

### 2.3 Розробка моделі та алгоритмів роботи системи

Для кращого розуміння роботи модулів програмного застосунку адміністрування пункту оренди туристичного спорядження було прийнято рішення розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.2).

Згідно вказаної діаграми користувач для початку взаємодії із програмним застосунком потрібно відчинити додаток, підвантажити базу даних та додати потрібного клієнта. Наступним кроком буде формування замовлення для певного клієнта.

По завершенню формування замовлень адміністратору потрібно зберегти наявні дані у базу [14].


Рисунок 2.2 – Модель системи у вигляді діаграми діяльності

Розробка моделі системи була виконана у програмному забезпеченні Draw IO.

Для реалізації програмного забезпечення для адміністрування пункту оренди туристичного спорядження необхідно розробити загальну модель роботи системи, який відображатиме логіку обробки запитів, взаємодії з базою даних та інтерфейсом користувача (рис. 2.3).

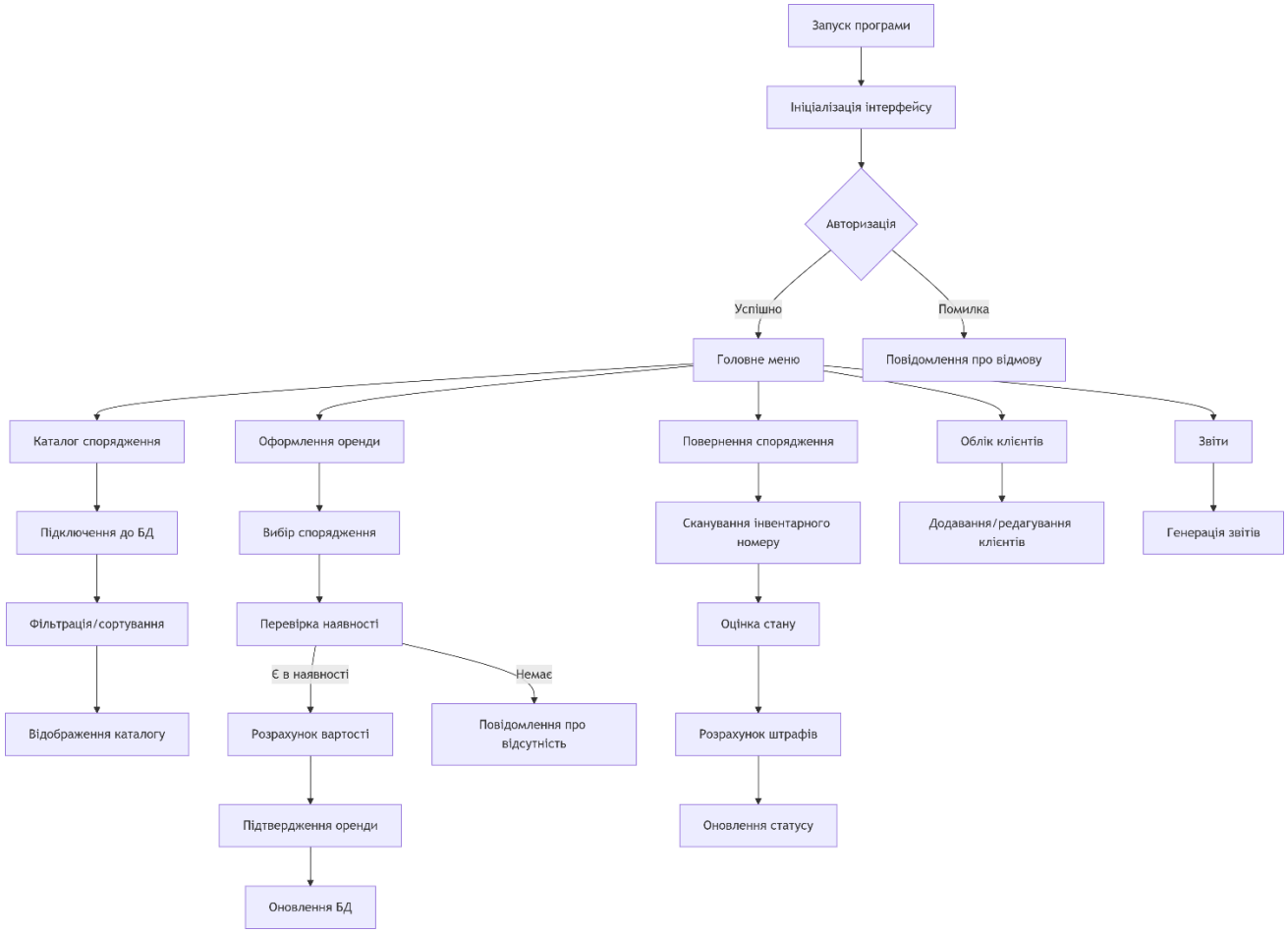


Рисунок 2.3 – Модель роботи програмного застосунку

Згідно з побудованою моделлю, після запуску програмного забезпечення ініціалізується головне вікно програми, яке забезпечує доступ користувача до основного функціоналу: перегляду спорядження, оформлення замовлень, управління клієнтами, генерації звітів тощо.

На першому етапі користувач (адміністратор чи працівник пункту оренди) може авторизуватись у системі. Після успішної авторизації

відкривається вікно головного меню. Якщо авторизація неуспішна — користувач отримує повідомлення про відмову в доступі.

Наступним кроком є вибір функціонального модуля: «Каталог спорядження», «Оформлення оренди», «Повернення спорядження», «Облік клієнтів» або «Звіти».

У випадку вибору модуля «Каталог спорядження» система під’єднується до бази даних і відображає перелік доступного туристичного спорядження. Користувач має змогу фільтрувати та сортувати дані за категоріями (тип, стан, наявність) або здійснювати пошук.

При переході до модуля «Оформлення оренди» система дозволяє обрати спорядження, ввести/вибрати клієнта, вказати термін оренди та автоматично розраховує вартість відповідно до тарифів. Після підтвердження замовлення система реєструє його у базі даних та зменшує кількість доступного спорядження.

У випадку повернення спорядження система оновлює його статус, надає можливість додати коментарі (наприклад, про пошкодження) та при потребі — розраховує штраф за прострочення.

Крім того, система передбачає генерацію звітів за періодами, типами спорядження, історією оренд тощо, що дає змогу адміністрації ефективно контролювати діяльність пункту.

Для кращого розуміння роботи модуля оформлення оренди розроблено допоміжну модель (рис. 2.4). Згідно з нею, після вибору клієнта та спорядження відбувається перевірка наявності спорядження, після чого розраховується сума до оплати з урахуванням днів оренди, тарифів, можливих знижок. Далі користувач підтверджує замовлення, і система реєструє запис у базі даних [15].

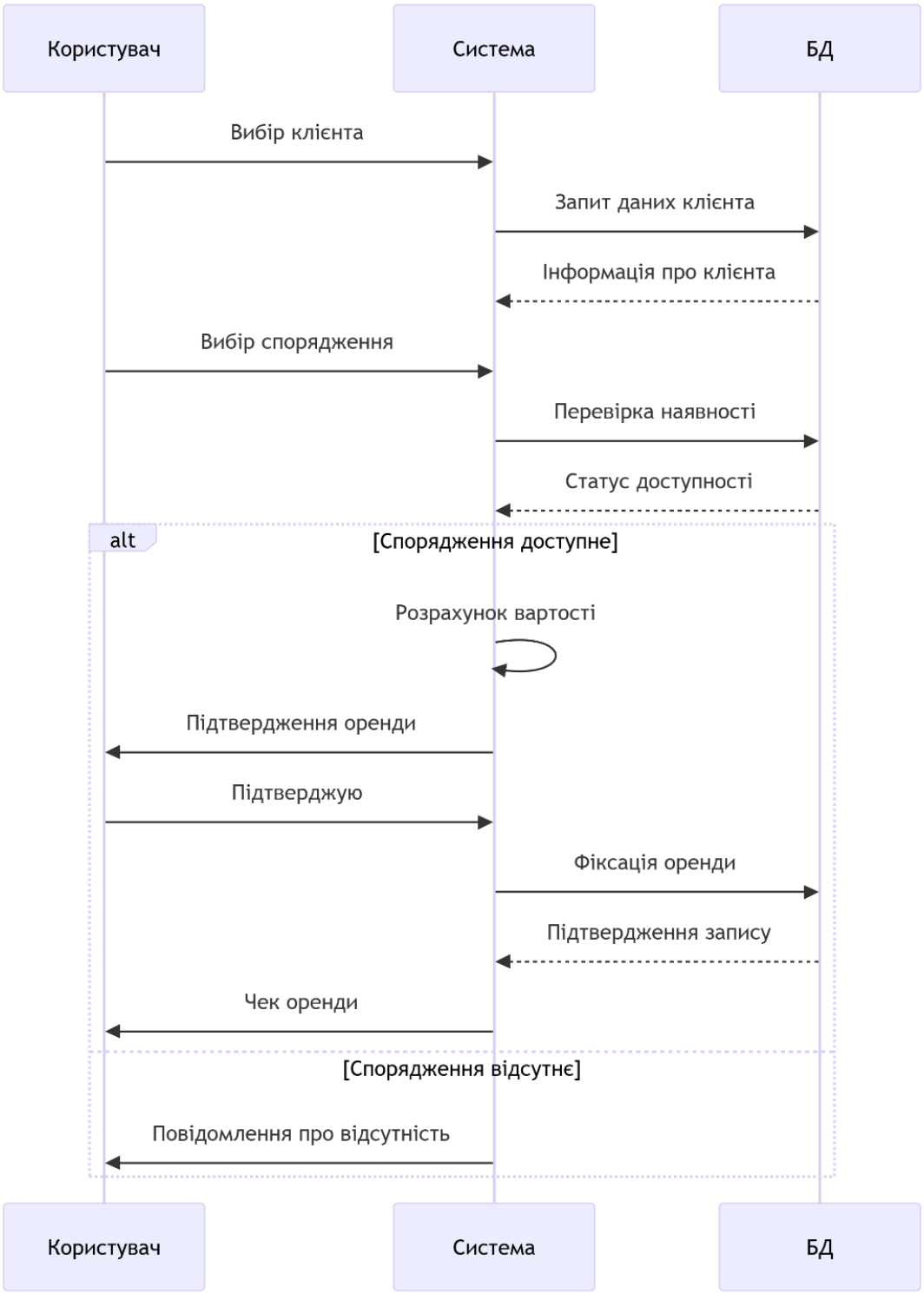


Рисунок 2.4 – Модель системи оформлення оренди

Також було створено основний алгоритм роботи програмного застосунку для адміністрування пункту оренди туристичного спорядження, який охоплює усі функції (рис. 2.5)

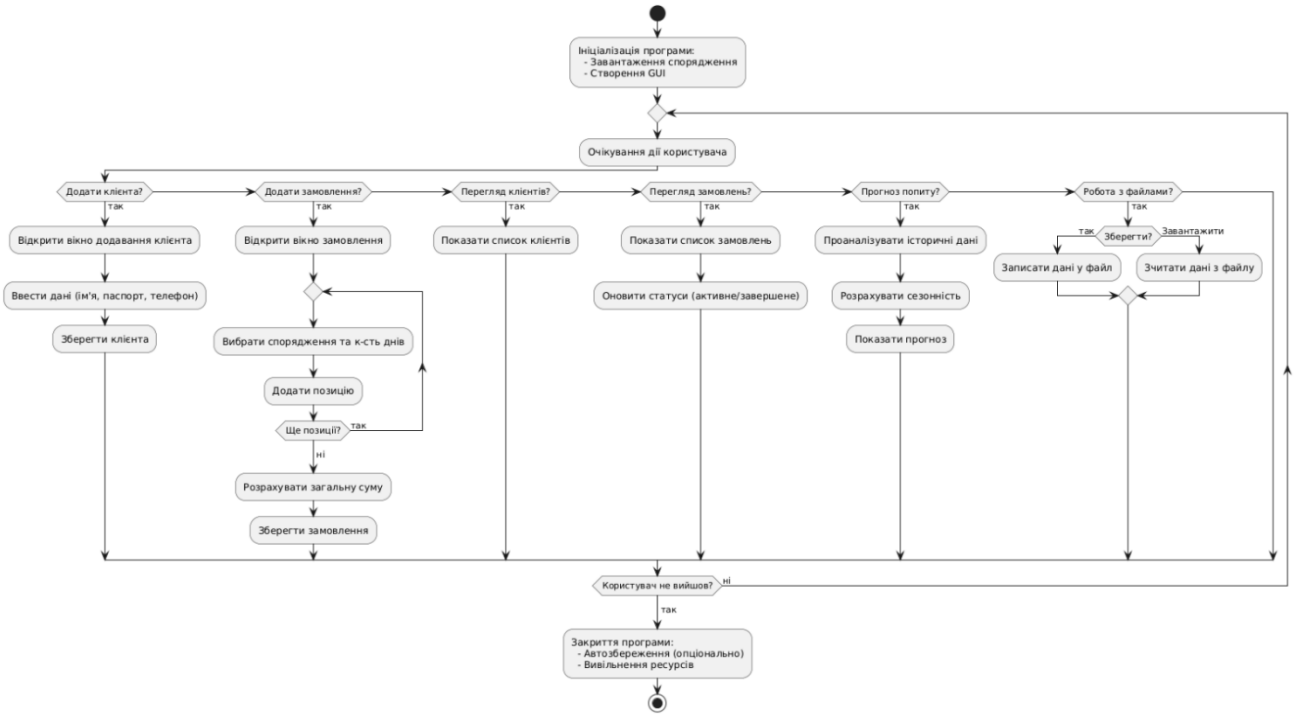


Рисунок 2.5 – Алгоритм роботи програмного застосунку

## 2.4 Висновки

У другому розділі було здійснено детальний аналіз вхідних та вихідних даних, що використовуються у програмному застосунку для адміністрування пункту оренди туристичного спорядження. Зокрема, розглянуто типи інформації, яка надходить до системи та формується на виході (квитанції про оренду, звіти про наявність спорядження, історія замовлень). Описано логіку створення нового замовлення, що включає перевірку доступності спорядження, реєстрацію клієнта, обчислення терміну оренди та формування відповідної документації. Було розроблено основний алгоритм роботи програмного забезпечення, який охоплює обробку запитів користувача, взаємодію з базою даних і формування вихідних результатів. Окрему увагу приділено алгоритму оформлення оренди — описано послідовність дій від моменту звернення клієнта до завершення операції, з урахуванням можливих виняткових ситуацій. Для більшої наочності побудовано алгоритм та моделі роботи програмного застосунку, модель системи оформлення оренди та алгоритм роботи прогнозування попиту на туристичне спорядження

# 3. РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

## 3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробка програмного забезпечення для адміністрування пункту оренди туристичного спорядження вимагає ретельного підбору засобів реалізації, які забезпечать високу продуктивність, зручність у користуванні, простоту в супроводі та адаптивність до майбутніх змін. У межах бакалаврської кваліфікаційної роботи прийнято рішення використовувати мову програмування C++ разом із технологією WinAPI для створення графічного інтерфейсу користувача, а також базу даних у вигляді текстового документа для зберігання інформації про клієнтів, обладнання, замовлення та історію оренди.

C++ — це високорівнева, компільована мова програмування з підтримкою об’єктно-орієнтованої парадигми, яка широко використовується для створення програм з високими вимогами до продуктивності та контролю над системними ресурсами. Однією з ключових переваг C++ є її низькорівнева природа, що дозволяє безпосередньо взаємодіяти з пам’яттю, файлами та обладнанням комп’ютера, а також висока швидкодія, що є критично важливим при розробці настільного застосунку з великою кількістю операцій введення/виведення.

WinAPI (Windows Application Programming Interface) — це стандартна бібліотека функцій, яка забезпечує доступ до основних можливостей операційної системи Windows. За її допомогою реалізується управління вікнами, обробка подій користувача, створення кнопок, полів введення, таблиць та інших компонентів графічного інтерфейсу. Крім того, WinAPI забезпечує взаємодію з файловою системою, дозволяє працювати з потоками, мережевими запитами та системними об'єктами. Використання WinAPI надає змогу створити легкий, нативний застосунок, що швидко виконується і не потребує додаткових бібліотек або середовищ виконання.

Microsoft Visual Studio (рис. 3.1) було обрано як основне інтегроване середовище розробки (IDE). Воно забезпечує всі необхідні інструменти для створення C++-додатків з використанням WinAPI, включаючи потужний редактор коду, зручну навігацію по проєкту, засоби налагодження, профілювання продуктивності та інтеграцію з системами контролю версій. Visual Studio також підтримує автоматичну генерацію шаблонного коду для вікон, обробників подій, структур даних тощо, що пришвидшує розробку та зменшує кількість помилок [16].

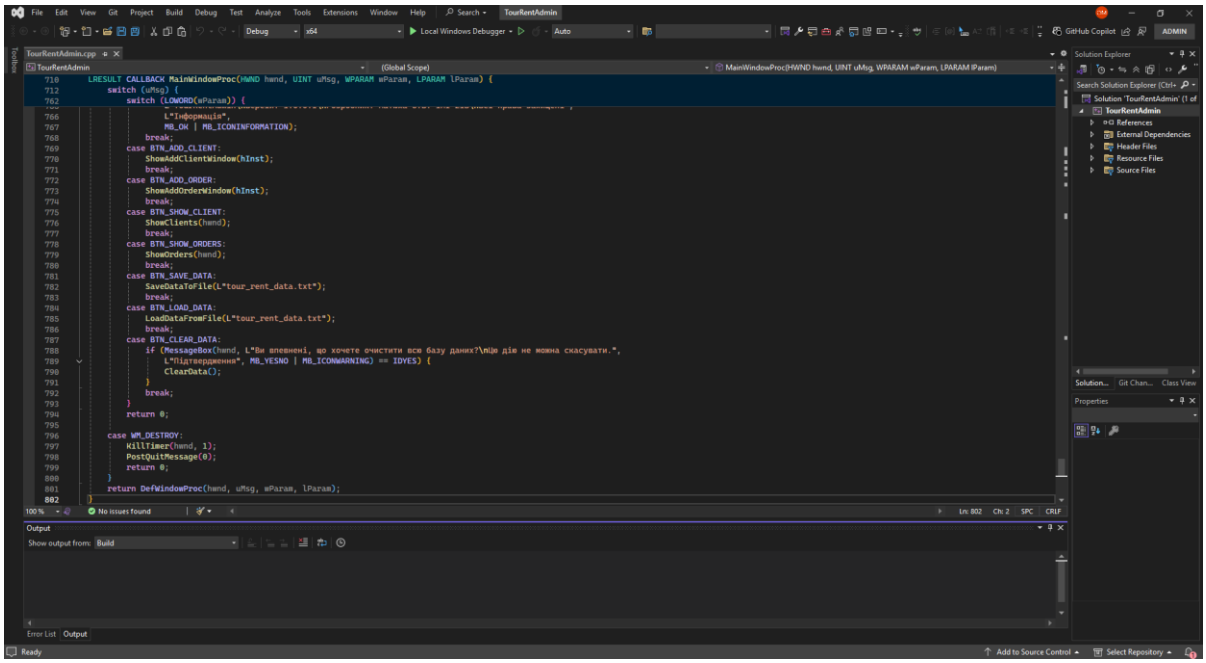


Рисунок 3.1 – Середовище розробки MS Visual Studio

Для забезпечення моделювання системи та її логічної структури використовувалися графічні інструменти, такі як Draw.io та Dia, які дозволяють створювати UML-діаграми (діаграми прецедентів, діяльності, класів тощо), що відіграють важливу роль на етапі проєктування, документування та обговорення архітектури програми. Створення діаграм дозволило краще структурувати програмні модулі, описати їх взаємодію та визначити логіку роботи ключових процесів у системі [17].

Також доцільно відзначити можливість подальшого розширення функціональності системи з використанням бібліотек STL (Standard Template

Library), які включають реалізації алгоритмів, структур даних та контейнерів (наприклад, std::vector, std::map, std::sort). Це дозволяє реалізувати внутрішню логіку обробки даних ефективно і надійно, забезпечуючи масштабованість застосунку.

Отже, поєднання мови програмування C++, засобів Windows API, бази даних SQLite та середовища розробки Visual Studio дозволяє реалізувати ефективну, функціональну та зручну для користувачів систему адміністрування пункту оренди туристичного спорядження. Обрані технології не лише задовольняють функціональні вимоги роботи, але й надають потенціал для подальшого розвитку програмного продукту — наприклад, додавання мережевої взаємодії, генерації звітів у PDF, інтеграції з мобільними застосунками або хмарними сервісами. Таким чином, сформована технічна база забезпечує надійну основу для створення конкурентоспроможного програмного рішення.

### 3.2 Розробка модуля додавання клієнтів

При створенні модуля додавання клієнтів до системи адміністрування пункту оренди туристичного спорядження важливо забезпечити простоту введення даних користувачем, коректність запису інформації у файл та зручність подальшої обробки записів. У межах бакалаврської кваліфікаційної роботи було прийнято рішення використовувати текстовий файл як базу даних, що значно спрощує реалізацію, дозволяючи уникнути складних механізмів взаємодії з повноцінною СКБД, що цілком виправдано для локального, автономного застосунку [18].

Усі дані про клієнтів зберігаються у текстовому файлі clients.txt. Кожен запис містить такі поля: ID клієнта, ПІБ, номер телефону, адреса та електронна пошта. Формат збереження даних – рядок із роздільниками |, що дозволяє легко одержувати інформацію построково та парсити значення при потребі.

Для реалізації модуля (рис. 3.2) було створено структуру Client, яка зберігає всі необхідні поля.

```
struct Client {
    std::string id;
    std::string fullName;
    std::string phoneNumber;
    std::string address;
    std::string email;
};
```

Рисунок 3.2 – Структура Client

Введення даних клієнта реалізується через діалогове вікно WinAPI, яке містить відповідні поля введення (Edit), кнопки підтвердження (Button) та обробку подій через механізм WM_COMMAND. Після натискання кнопки "Додати", дані з полів вводяться у структуру Client, формуються у вигляді одного рядка та записуються у файл. Це відображено на рисунку 3.3.

```
void WriteClientToFile(const Client& client) {
    std::ofstream file("clients.txt", std::ios::app); // відкриття в режимі дописування
    if (file.is_open()) {
        file << client.id << "|"
             << client.fullName << "|"
             << client.phoneNumber << "|"
             << client.address << "|"
             << client.email << "\n";
        file.close();
    }
    else {
        MessageBox(NULL, L"Не вдалося відкрити файл для запису", L"Помилка", MB_OK | MB_ICONERROR);
    }
}
```

Рисунок 3.3 – Введення даних клієнта

Наступним кроком є забезпечення унікальності ID клієнта (рис. 3.4). Для цього було реалізовано функцію автоматичного генерування ID на основі поточної дати й часу.

```
std::string GenerateUniqueID() {
    SYSTEMTIME st;
    GetLocalTime(&st);
    char buffer[30];
    sprintf(buffer, "CL%04d%02d%02d%02d%02d%02d",
        st.wYear, st.wMonth, st.wDay,
        st.wHour, st.wMinute, st.wSecond);
    return std::string(buffer);
}
```

Рисунок 3.4 – ID клієнта

Дана функція викликається перед збереженням клієнта і забезпечує, що кожен клієнт матиме власний унікальний ідентифікатор, що спрощує подальшу обробку записів.

Для полегшення подальшого пошуку та редагування клієнтів інформація з файлу зчитується у вектор структур Client, що відображено на рисунку 3.5.

```
std::vector<Client> ReadClientsFromFile() {
    std::ifstream file("clients.txt");
    std::vector<Client> clients;
    std::string line;
    while (std::getline(file, line)) {
        std::istringstream ss(line);
        std::string field;
        Client client;

        std::getline(ss, client.id, '|');
        std::getline(ss, client.fullName, '|');
        std::getline(ss, client.phoneNumber, '|');
        std::getline(ss, client.address, '|');
        std::getline(ss, client.email, '|');

        clients.push_back(client);
    }
    return clients;
}
```

Рисунок 3.5 – Вектор структур Client

Ця функція дозволяє легко сформувати таблицю всіх клієнтів у іншому модулі системи (наприклад, перегляду або редагування), що забезпечує масштабованість та гнучкість у роботі з даними.

Таким чином, розроблений модуль додавання клієнтів реалізує всі необхідні функції: зручне введення інформації, збереження у текстовий файл, гарантування унікальності клієнта, а також подальшу обробку збережених даних. Обраний підхід є простим та ефективним для невеликої десктопної системи, яка не потребує складних засобів збереження даних, таких як повноцінні СКБД.

### 3.3 Розробка модуля створення замовлень

Головною задачею модуля управління замовленнями є автоматизація процесів оформлення, обліку та аналізу оренди туристичного спорядження. Для реалізації цих функцій було створено клас OrderManager, який відповідає за обробку всіх операцій, пов'язаних із замовленнями (рис. 3.5).

```
class OrderManager {
private:
    vector<Order> orders;
    string dataFile = "orders.txt";

public:
    void CreateOrder(Client client, vector<OrderItem> items) {
        Order newOrder;
        newOrder.client = client;
        newOrder.items = items;
        newOrder.startTime = time(0);
        newOrder.isActive = true;

        // Розрахунок загальної вартості
        for (auto& item : items) {
            newOrder.totalPrice += item.equipment.pricePerDay * item.days;
            if (item.days > newOrder.durationDays) {
                newOrder.durationDays = item.days;
            }
        }

        orders.push_back(newOrder);
        SaveToFile();
    }

    void SaveToFile() {
        // Реалізація збереження у текстовий файл
    }
};
```

Рисунок 3.6 – Обробка операцій замовлення

Для перевірки доступності спорядження було розроблено окрему функцію CheckEquipmentAvailability (рис. 3.6), яка аналізує поточний стан інвентарю.

```
bool CheckEquipmentAvailability(string equipmentName, int requiredDays) {
    // Перевірка наявності у базі даних
    for (auto& item : equipmentList) {
        if (item.name == equipmentName) {
            // Перевірка активних замовлень
            for (auto& order : orderList) {
                if (order.isActive) {
                    for (auto& orderItem : order.items) {
                        if (orderItem.equipment.name == equipmentName) {
                            // Логіка перевірки доступності
                        }
                    }
                }
            }
            return true;
        }
    }
    return false;
}
```

Рисунок 3.7 – Аналіз стану інвентарю

Для розрахунку вартості оренди було реалізовано алгоритм зображений на рисунку 3.8.

```
double CalculateBaseCost(OrderItem item) {
    return item.equipment.pricePerDay * item.days;
}
```

Рисунок 3.8 – Розрахунок вартості замовлень

Для автоматичного оновлення статусу замовлень було розроблено функцію UpdateOrdersStatus, яка щохвилини перевіряє терміни оренди (рис. 3.9).

```
void UpdateOrdersStatus() {
    time_t now = time(0);
    for (auto& order : orders) {
        if (order.isActive) {
            time_t endTime = order.startTime + (order.durationDays * 24 * 3600);
            if (now >= endTime) {
                order.isActive = false;
                // Автоматичне оновлення статусу спорядження
                UpdateEquipmentStatus(order.items, true);
            }
        }
    }
}
```

Рисунок 3.9 – Таймер терміну оренди

### 3.4 Розробка модуля збереження, завантаження та очищення даних

У межах розробки програмного застосунку для адміністрування пункту оренди туристичного спорядження було реалізовано окремий модуль для збереження, завантаження та очищення даних клієнтів, які зберігаються у звичайному текстовому файлі clients.txt.

Цей модуль має важливу роль у забезпеченні сталого доступу до інформації між сесіями програми, дозволяючи зчитувати дані після запуску, зберігати нові записи та очищати файл при необхідності.

Дані про клієнтів (рис.3.10) додаються у текстовий файл у вигляді форматованого рядка. Для забезпечення консистентності всі поля розділяються символом |, що дозволяє зручно парсити записи при завантаженні.

```
void SaveClientToFile(const Client& client) {
    std::ofstream file("clients.txt", std::ios::app);
    if (file.is_open()) {
        file << client.id << "|"
            << client.fullName << "|"
            << client.phoneNumber << "|"
            << client.address << "|"
            << client.email << "\n";
        file.close();
    }
    else {
        MessageBox(NULL, L"Помилка відкриття файлу для запису", L"Помилка", MB_OK | MB_ICONERROR);
    }
}
```

Рисунок 3.10 – Збереження клієнтів

Ця функція використовується кожного разу, коли користувач додає нового клієнта до системи через інтерфейс.

Для коректної роботи модуля перегляду клієнтів реалізовано функцію зчитування файлу та перетворення кожного рядка у структуру Client. Записи додаються у вектор (рис 3.11), який зручно обробляти в логіці програми.

```
std::vector<Client> LoadClientsFromFile() {
    std::ifstream file("clients.txt");
    std::vector<Client> clients;
    std::string line;

    while (std::getline(file, line)) {
        std::istringstream ss(line);
        Client client;

        std::getline(ss, client.id, '|');
        std::getline(ss, client.fullName, '|');
        std::getline(ss, client.phoneNumber, '|');
        std::getline(ss, client.address, '|');
        std::getline(ss, client.email, '|');

        clients.push_back(client);
    }

    return clients;
}
```

Рисунок 3.11 – Завантаження даних

Ця функція викликається при запуску програми або при оновленні списку клієнтів у відповідному вікні.

Функціональність очищення всіх клієнтів (рис. 3.12) реалізовано через просте перезаписування файлу з його відкриттям у режимі trunc, що миттєво видаляє весь вміст файлу.

```
void ClearClientFile() {
    std::ofstream file("clients.txt", std::ios::trunc);
    if (file.is_open()) {
        file.close();
        MessageBox(NULL, L"Дані успішно очищено", L"Інформація", MB_OK | MB_ICONINFORMATION);
    }
    else {
        MessageBox(NULL, L"Не вдалося очистити файл", L"Помилка", MB_OK | MB_ICONERROR);
    }
}
```

Рисунок 3.12 – Очищення даних

Очищення даних може бути викликане через меню програми або окрему кнопку у вікні адміністрування клієнтів.

Усі три функції інтегруються з GUI через обробку повідомлень WM_COMMAND (рис.3.13). Наприклад, при натисканні кнопки "Очистити" у вікні, відповідний ідентифікатор надсилає повідомлення, яке викликає ClearClientFile().

```
case IDC_BUTTON_CLEAR:
    if (MessageBox(hWnd, L"Очистити всі дані клієнтів?", L"Підтвердження", MB_YESNO | MB_ICONQUESTION) == IDYES) {
        ClearClientFile();
    }
    break;
```

Рисунок 3.13 – Інтеграція з WinAPI

Таким чином, модуль збереження, завантаження та очищення даних реалізовано просто і ефективно. Він забезпечує стабільне зберігання інформації у форматі, зручному для обробки, не вимагає додаткових залежностей і є достатнім для потреб локальної десктопної системи. У разі необхідності розширення функціоналу (наприклад, підтримки фільтрації чи пошуку), структура збереження також дозволяє легко адаптувати існуючі рішення.

### 3.5 Розробка структури інтерфейсу програмного застосунку

Під час розробки інтерфейсу програмного застосунку було наділено велику увагу його зрозумілості для користувача, а також та зручності, а саме адміністратора пункту оренди туристичного спорядження.

Головними кольорами інтерфейсу було обрано білий та сірий, оскільки данні кольори асоціюється з простотою та максимальною зрозумілістю інтерфейсу користувача.

Під час відкриття програмного застосунку, користувача зустрічає головний екран із назвою застосунку (рис. 3.14).

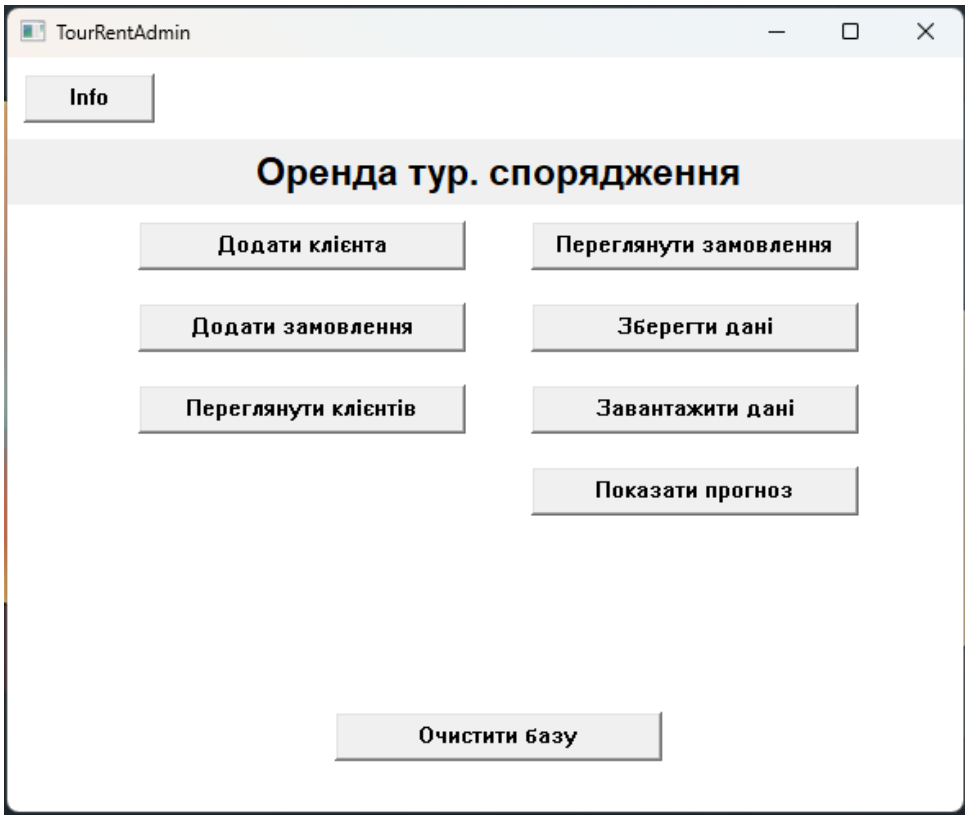


Рисунок 3.14 – Головний екран

Оскільки програмні засоби передбачають взаємодію із адміністрацією оренди туристичного спорядження, було вирішено створити робоче середовище із чітким розмежуванням на розділи та певними кнопками які відповідають за основні функції. Було створено меню додавання клієнтів (рис. 3.15), меню додавання замовлень (рис. 3.16).

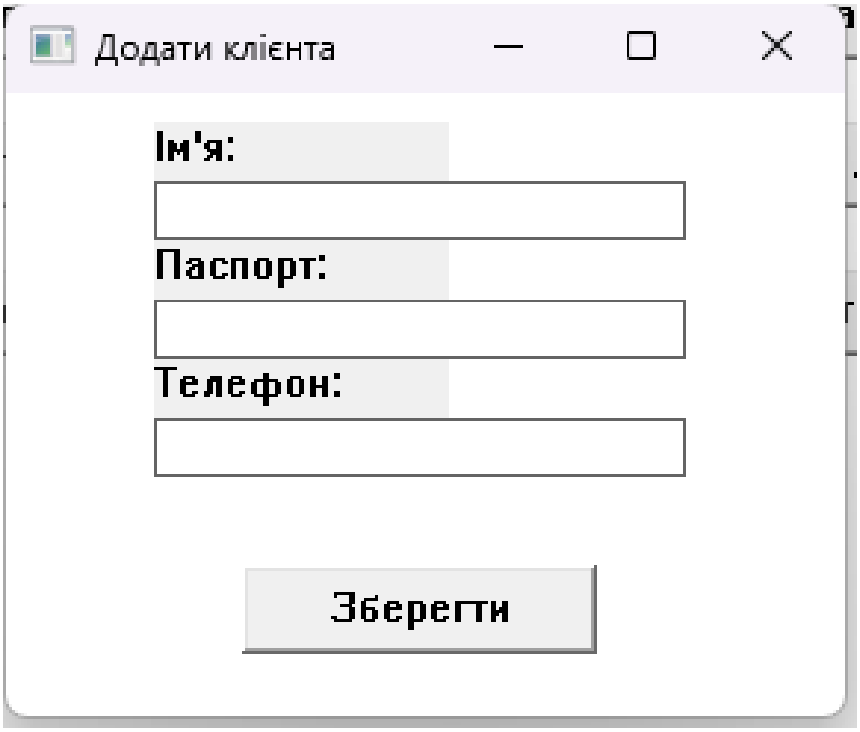


Рисунок 3.15 – Меню додавання клієнтів

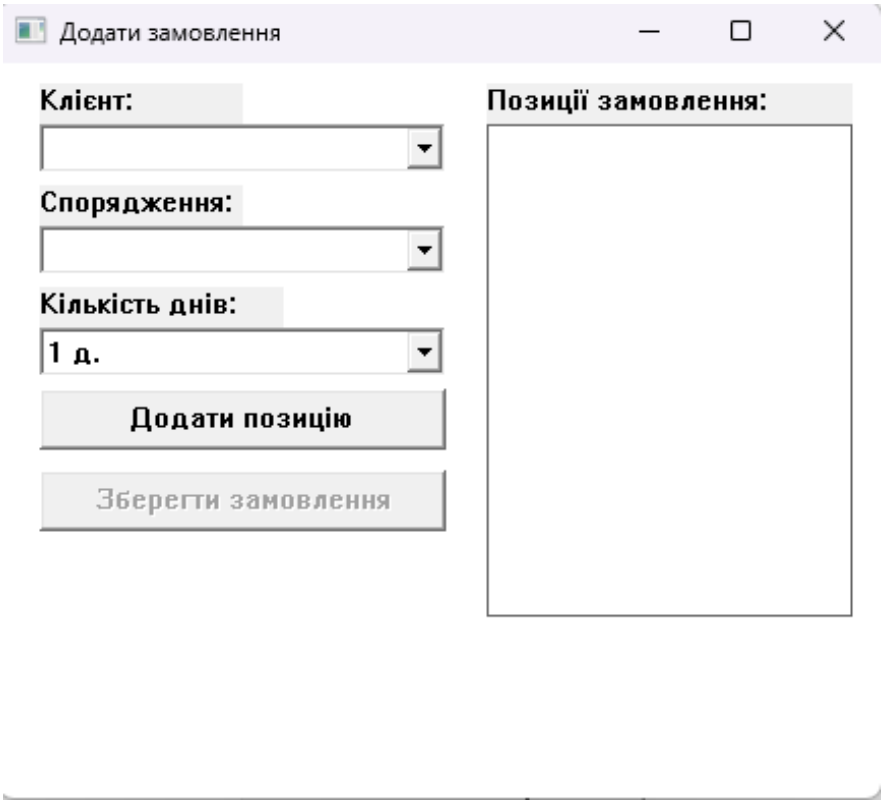


Рисунок 3.16 – Меню додавання замовлень

Після того як клієнта було додано (рис. 3.17), у вікні «Переглянути клієнтів» можна побачити всю наявну інформацію про поточних клієнтів

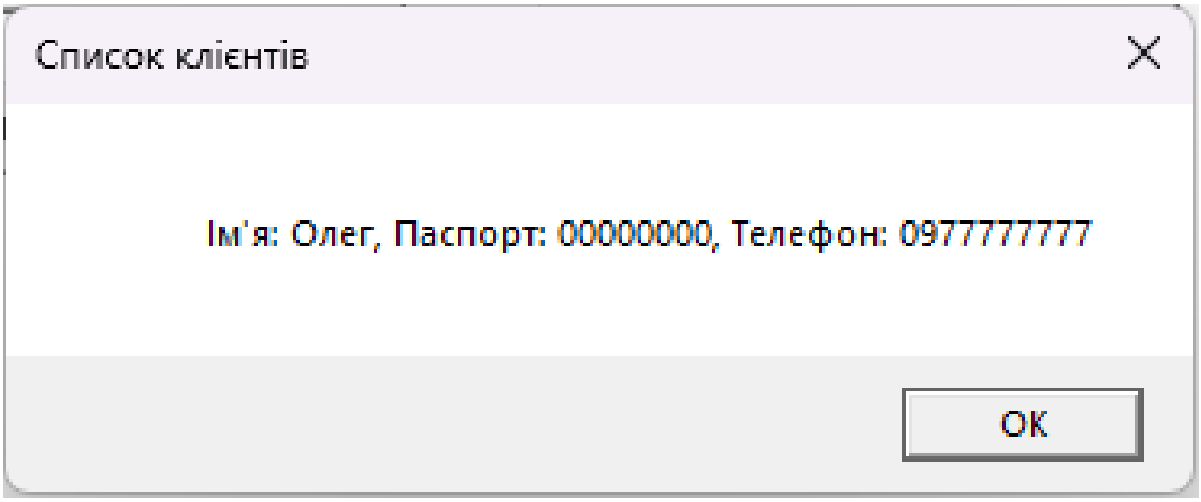


Рисунок 3.17 – Вікно списку клієнтів

Після того як клієнтів було додано, йде етап створення замовлення, по проходженню якого, у вікні «Переглянути замовлення» можна побачити інформацію (рис. 3.18) про усі наявні замовлення. Також відображена інформація про вартість замовлення та таймер повернення спорядження клієнтом.

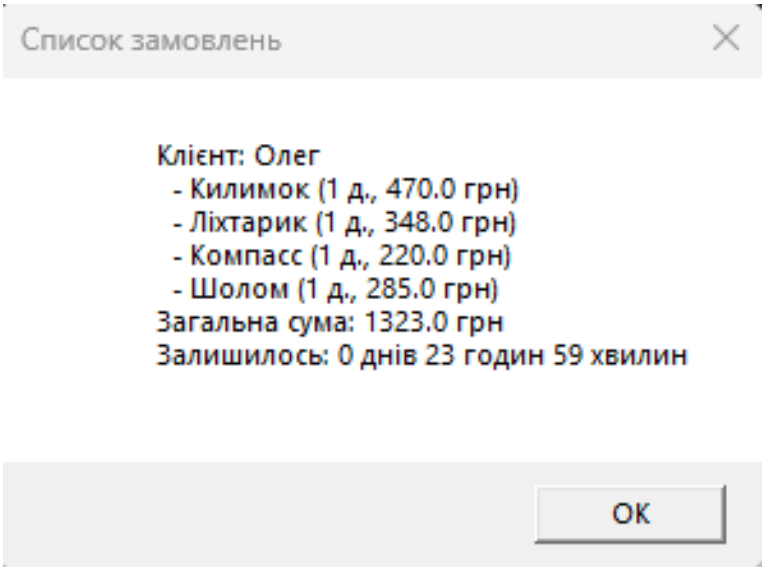


Рисунок 3.18 – Вікно перегляду замовлень

При натисканні кнопки «Зберегти дані» користувач бачитиме інформуюче вікно (рис. 3.19), із повідомленням про те, що усі дані було успішно збережено у файл.

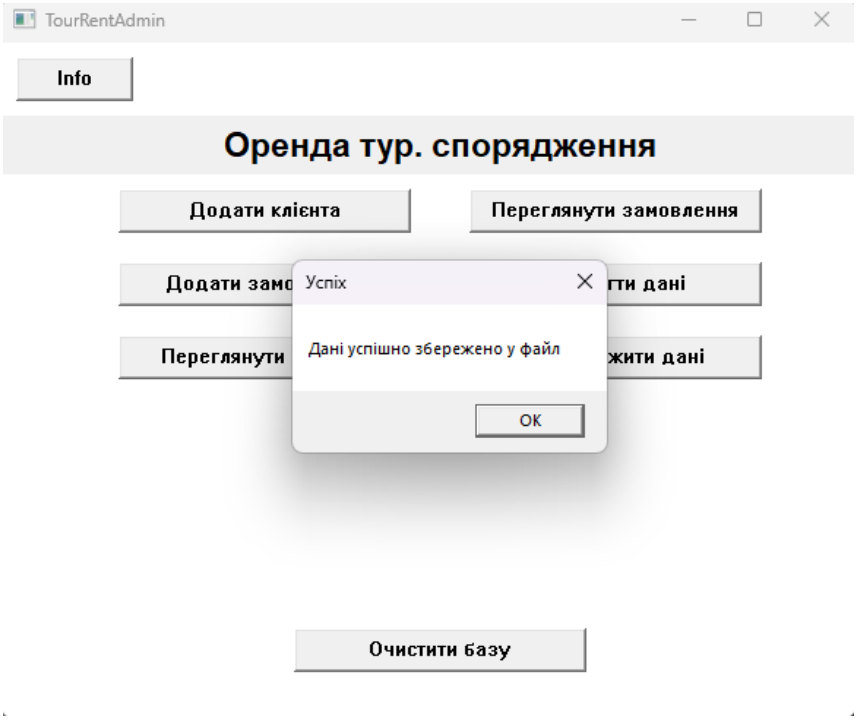


Рисунок 3.19 – Збереження даних

Після того як додаток був відкритий та адміністратору потрібно підвантажити базу клієнтів та замовлень, потрібно натиснути на кнопку «Завантажити дані». По натисканню користувач побачить діалогове вікно (рис. 3.20) яке сповістить що дані було успішно завантажено.

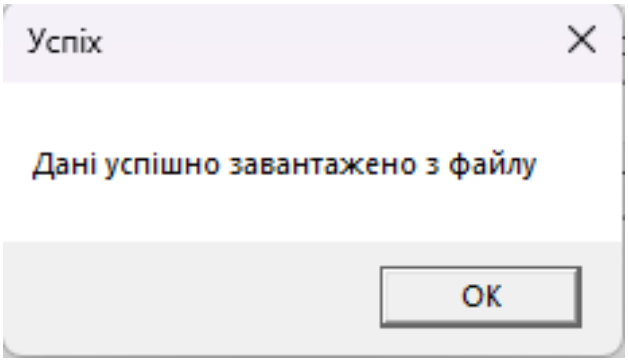


Рисунок 3.20 – Екран «Про програму»

За потреби адміністратором очистити наявну базу даних потрібно натиснути на кнопку «Очистити базу». Після цього база буде очищена та користувач побачить діалогове вікно із підтвердженням цієї дії (рис. 3.21).

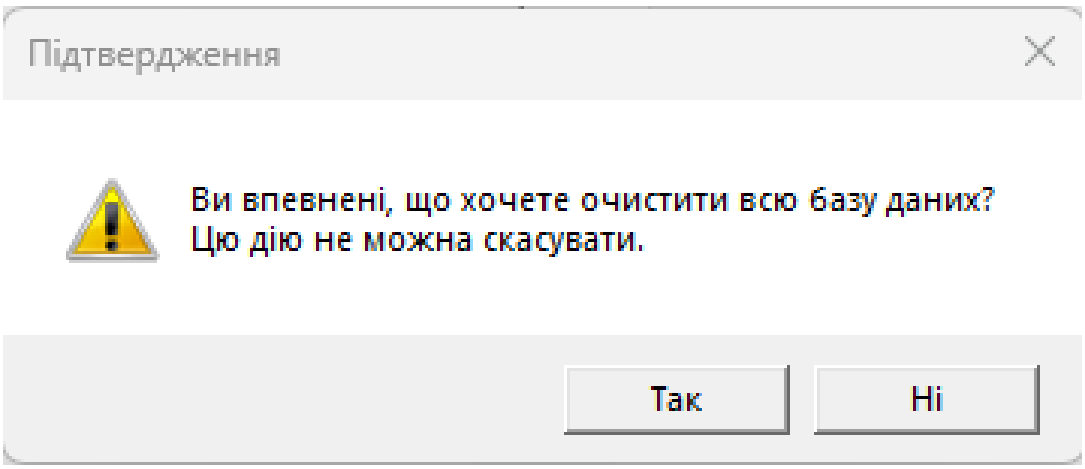


Рисунок 3.21 – Підтвердження очищення бази даних

Якщо відповідь адміністратора була схвальною, діалогове вікно (рис. 3.22) сповістить його про те, що базу даних було успішно очищено.

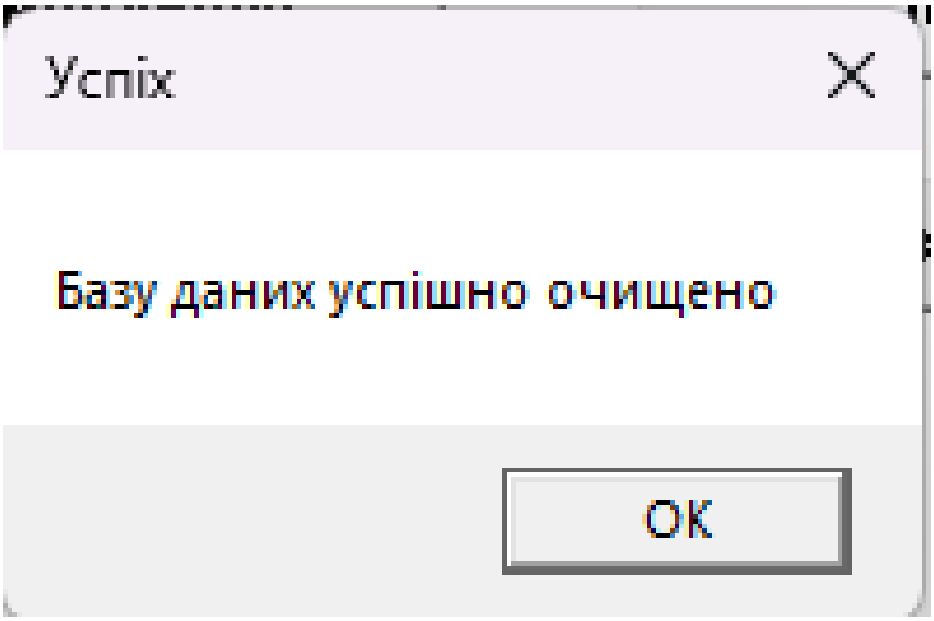


Рисунок 3.22 – Успішне очищення бази даних

Також доступна функція (рис. 3.23) перегляду інформації про програмний застосунок, яку можна побачити при натисканні кнопки «Info».

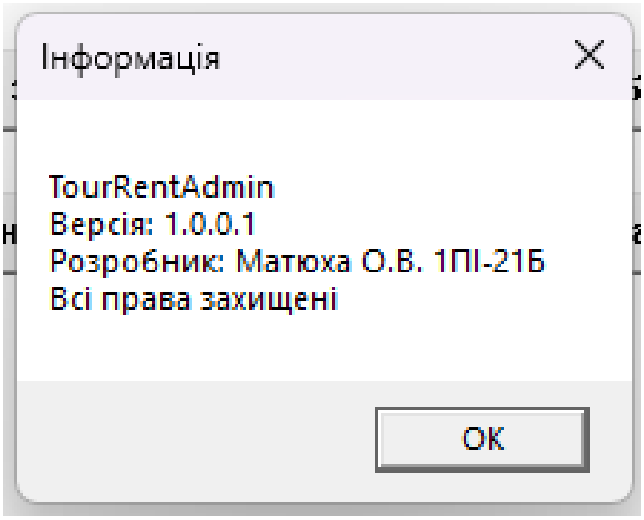


Рисунок 3.23 – Info

За потреби адміністратора дізнатись попит на певне туристичне спорядження, при натисканні відповідної кнопки відкривається вікно «Прогноз попиту» (рис. 3.24).

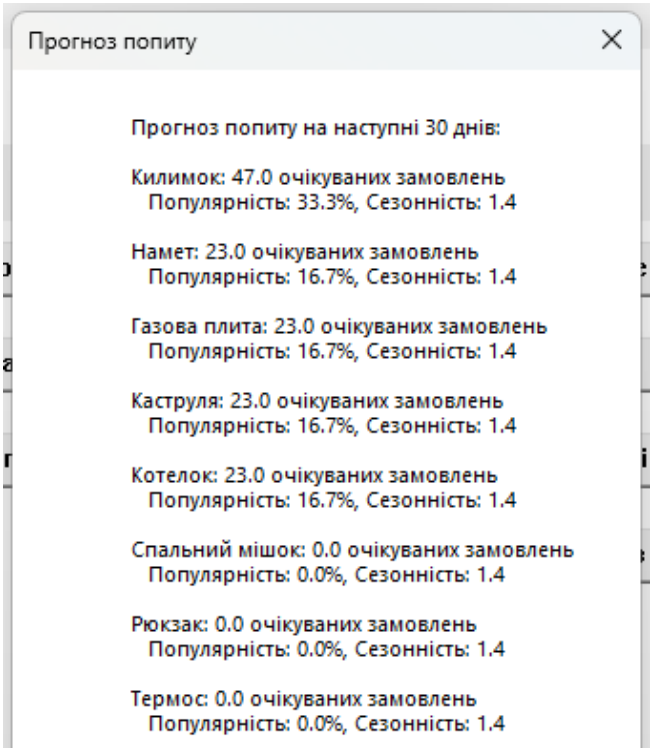


Рисунок 3.24 – Прогноз попиту

Розробка графічного інтерфейсу програмного застосунку була проведена у середовищі розробки Visual Studio з використанням технологій WinAPI та мови програмування C++.

### 3.6 Висновки

У третьому розділі було обґрунтовано вибір технологій і засобів, які були використані для розробки програмних модулів системи адміністрування пункту оренди туристичного спорядження. Було прийнято рішення реалізовувати систему з використанням мови програмування C++ у середовищі WinAPI, що дозволило створити нативне десктопне застосування з повноцінною взаємодією через графічний інтерфейс. Для реалізації збереження даних було обрано формат звичайного текстового файлу, що забезпечує простоту у використанні та зменшує потребу у зовнішніх залежностях.

У межах розділу детально описано реалізацію модулів додавання, завантаження, збереження та очищення інформації про клієнтів. Також були наведені фрагменти коду, що ілюструють функціонування цих модулів у рамках WinAPI, було розроблено структуру інтерфейсу програмного застосунку. Описані функції забезпечують плацдарм для майбутнього розширення функціональності системи та є важливими елементами для ефективного адміністрування бази клієнтів.

# 4 ТЕСТУВАННЯ ПРОГРАМИ

## 4.1 Тестування системи

Тестування програмного забезпечення – це процес перевірки програмного застосунку з метою виявлення різних помилок, дефектів або неполадок та перевірки відповідності вимогам замовника. Основна мета тестування – забезпечити якість та надійність програмного застосунку перед його випуском у виробництво або початком експлуатації [19].

Щоб розпочати тестування, слід встановити тестову версію програмного забезпечення на комп’ютер із операційною системою Windows, яка використовуватиметься у процесі перевірки. До початку тестування потрібно підготувати відповідну документацію, зокрема специфікацію вимог та тестовий план. Процес тестування охоплює кілька етапів і включає різні типи перевірок.

На першому етапі виконується функціональне тестування, метою якого є перевірка коректності роботи програмного забезпечення. До цього етапу входять перевірки введення та виведення даних, обробка вхідних значень, а також тестування відповідності поведінки програми визначеній бізнес-логіці.

Наступним етапом є тестування на відмовостійкість, яке має на меті виявлення збоїв і помилок у роботі програмного забезпечення. У межах цього процесу перевіряється здатність системи відновлювати свою функціональність після виникнення критичних ситуацій. До таких перевірок можуть входити навмисне введення некоректних даних, використання нестандартних або непередбачуваних вхідних параметрів, а також імітація роботи застосунку в умовах, наближених до екстремальних, зокрема — без доступу до зовнішніх компонентів чи сервісів системи [20].

Після завершення функціонального та відмовостійкого тестування виконується додатковий етап - оцінка продуктивності програмного забезпечення. На цьому етапі проводяться тести на швидкодію, метою яких є аналіз часу виконання конкретних операцій або завдань. Під час тестування

фіксуються показники часу реакції застосунку на різні запити чи дії з боку користувача.

Одним із додаткових етапів тестування є перевірка сумісності, яка дозволяє визначити, наскільки коректно програмне забезпечення функціонує в різних операційних системах, версіях програмного забезпечення та на різних апаратних платформах. Під час цього процесу оцінюється працездатність застосунку в різних конфігураціях середовища, а також виявляються можливі конфлікти або збої при взаємодії з іншими програмами [21].

Тестування системи адміністрування пункту оренди туристичного спорядження є невід'ємним етапом життєвого циклу програмного забезпечення, оскільки воно дозволяє переконатися в правильності роботи основних функціональних модулів, а також виявити потенційні помилки до початку реальної експлуатації.

Основною метою тестування є перевірка коректності функціонування системи, її стійкості до помилок, зручності у використанні та відповідності попередньо сформульованим вимогам. Перевірка роботи програми здійснювалась на комп'ютері з операційною системою Microsoft Windows 11, у локальному середовищі з використанням WinAPI та текстового файлу, що виконував роль бази даних.

На етапі функціонального тестування було перевірено основні функції програми.

В першу чергу було протестовано додавання нових клієнтів. Результати тестування показано на рисунку 4.1.

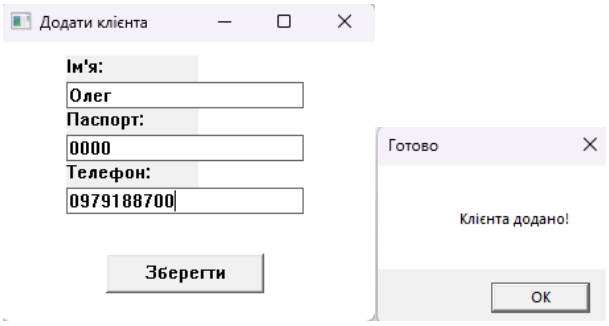


Рисунок 4.1 – Тестування додавання клієнтів

Наступним етапом було протестовано фунцію формування нових замовлень для існуючого клієнта. Результати тестування відображено на рисунку 4.2.

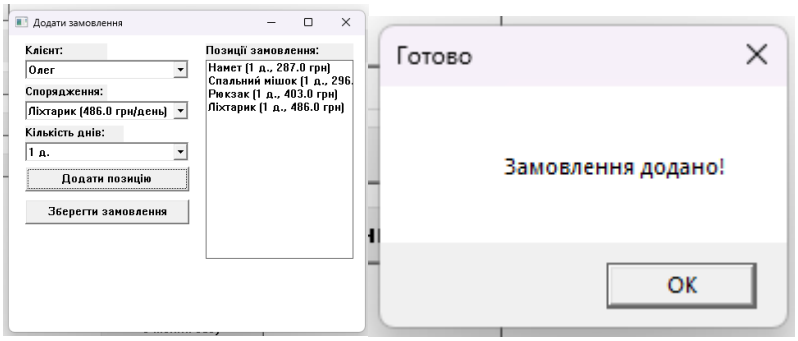


Рисунок 4.2 – Тестування додавання клієнтів

Дані про клієнтів та замовлення, при натисканні кнопки «Зберегти дані» зберігаються у текстовому файлі tour_rent_data.txt, показаному на рисунку 4.3.

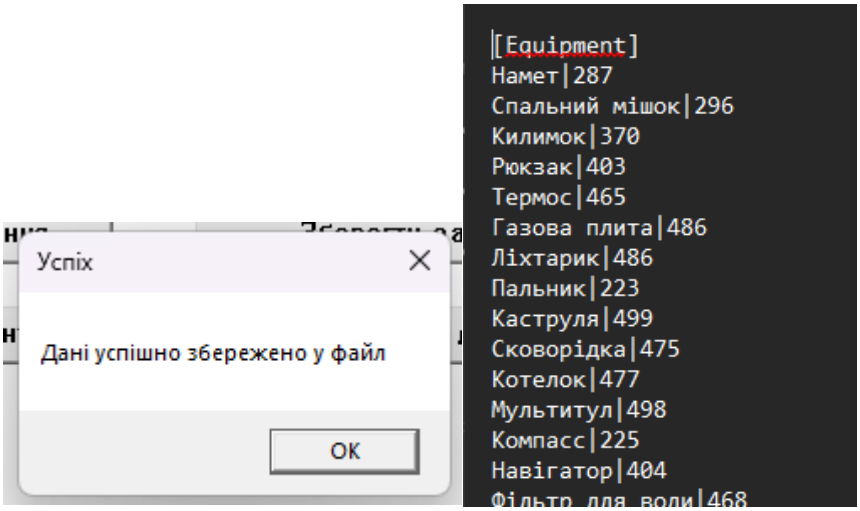


Рисунок 4.3 – Збереження даних у текстовий документ

Наступною протестованою функцією було очищення бази даних шляхом натискання кнопки «Очистити базу», яка після підтвердження очищує наявну юазу даних, що показано на рисунку 4.4.

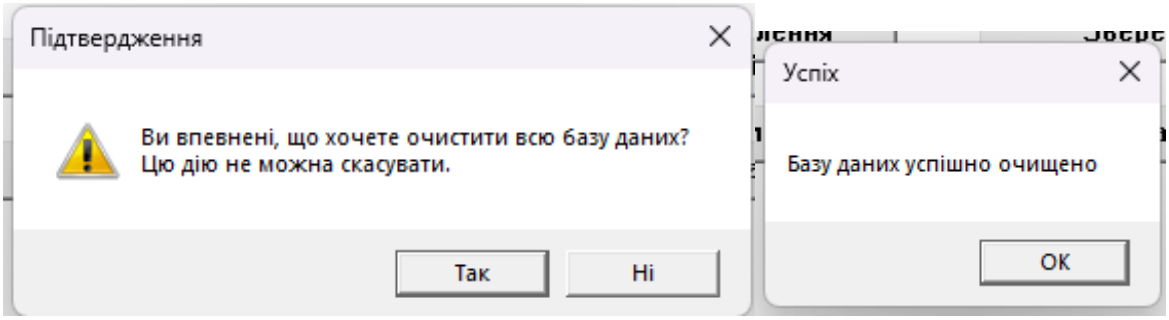


Рисунок 4.4 – Очищення бази даних

Останньою протестованою функцією, є функція підвантаження даних з бази при відкритті програми та натисканні кнопки «Завантажити дані», що відображено на рисунку 4.5.

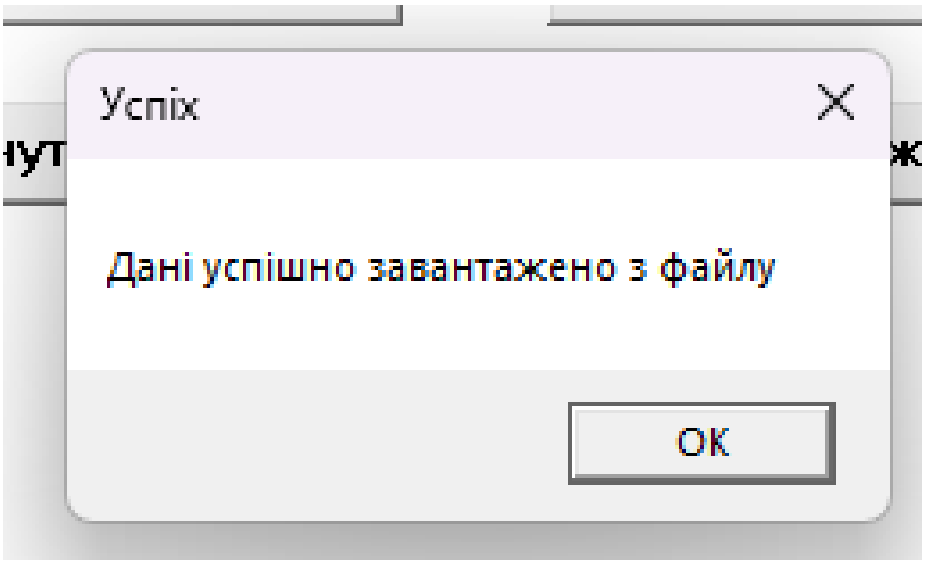


Рисунок 4.5 – Завантаження даних

При тестуванні відмов було протестовано декілька основних кейсів. Таких як створення клієнта із пустими полями та формування пустого замовлення. У всіх випадках система давала помилку, що відображено на рисунках 4.6 – 4.7

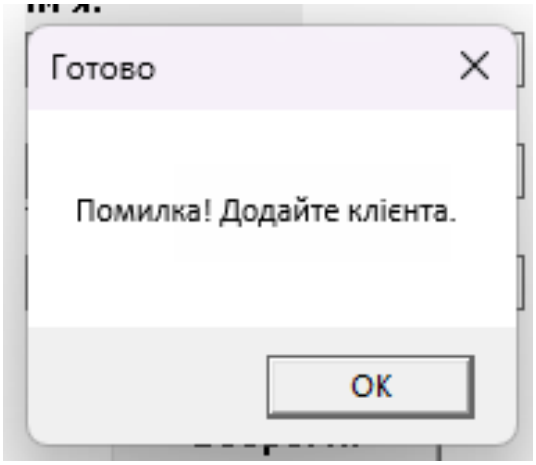


Рисунок 4.6 – Помилка додавання клієнта

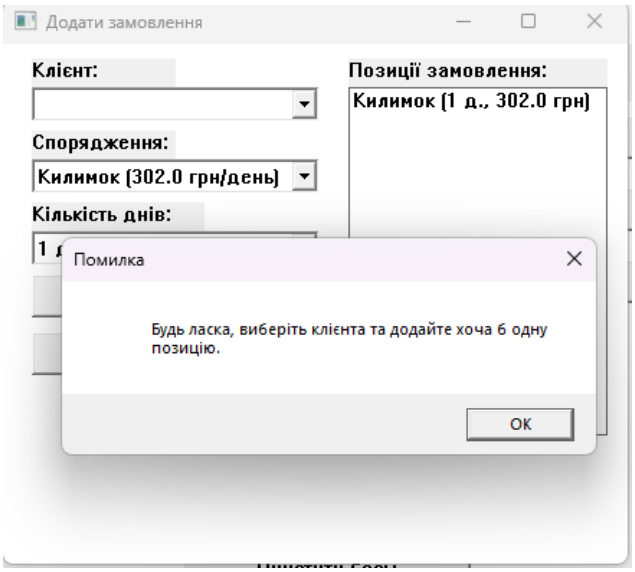


Рисунок 4.7 – Додавання пустого замовлення

Під час проведення тестування продуктивності було протестовано швидкодію системи при додаванні, збереженні та завантаженні великої кількості клієнтів (до 100 записів).

Результати:

- Час обробки великого обсягу даних був незначним (<1 сек).
- Затримок у графічному інтерфейсі не спостерігалось

Під час проведення тестування сумісності програму було перевірено на різних версіях Windows (Windows 8.1, Windows 10, Windows 11). Усі функції працювали стабільно. Проблем сумісності з файловою системою або графічним інтерфейсом не виявлено.

Тестування показало, що розроблене програмне забезпечення є стабільним, функціональним та зручним для використання. Воно правильно реагує на введення некоректних даних, стабільно працює при завантаженні великих обсягів інформації, а також сумісне з актуальними версіями операційних систем Windows. Проведені тести дозволяють рекомендувати систему для подальшого використання в реальних умовах.

### 4.2 Розробка інструкції користувача

У інструкції користувача наведено технічні вимоги для запуску програмного забезпечення. Інформація про мінімальні та рекомендовані параметри апаратного забезпечення представлена в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

| Вимоги до конфігурації | Рекомендовано | Мінімально |
|---|---|---|
| Процесор | Intel Core i3 | Intel Core i5 |
| Оперативна пам'ять | 8 ГБ | 4 ГБ |
| Вільний простір на диску | 2 ГБ | 1 ГБ |
| Відеокарта | інтегрована | інтегрована |
| Операційна система | Windows 10 | Windows 7 |
| Монітор | Розширення екрану 1920x1080 пікселів, діагональ 20" чи більше | Розширення екрану 1280x720 пікселів, діагональ 14" або більше |

Після інсталяції та запуску програмного забезпечення, користувач потрапляє у вікно головного екрану, де основні функції програмного застосунку. Головне меню дозволяє перейти до фундаментальних можливостей застосунку: перегляду, додавання, редагування, видалення спорядження, оформлення замовлень, перегляду історії, а також взаємодії з базою клієнтів.

Першим кроком роботи користувача зазвичай є авторизація або вхід до системи з правами адміністратора чи працівника пункту оренди. Після входу та натискання кнопки “Завантажити дані” система автоматично синхронізується з локальною базою даних та завантажує актуальну інформацію про доступне спорядження.

Для оформлення нового замовлення користувач переходить до вкладки «Оренда», де обирає спорядження, вказує терміни оренди, а також обирає клієнта із наявної бази або створює нового. Після підтвердження створюється запис у базі даних, стан замовлення змінюється на успішний.

У вкладці «Клієнти» користувач має можливість переглядати історію замовлень, редагувати контактні дані клієнтів та здійснювати пошук за ПІБ або номером телефону.

Для адміністративного управління доступні функції створення резервних копій бази даних, налаштування прав доступу.

Інтерфейс системи інтуїтивно зрозумілий, усі дії супроводжуються підказками та повідомленнями про успішне виконання операцій або можливі помилки. Приклад екрану успішного оформлення оренди відображено на рисунку 4.8.

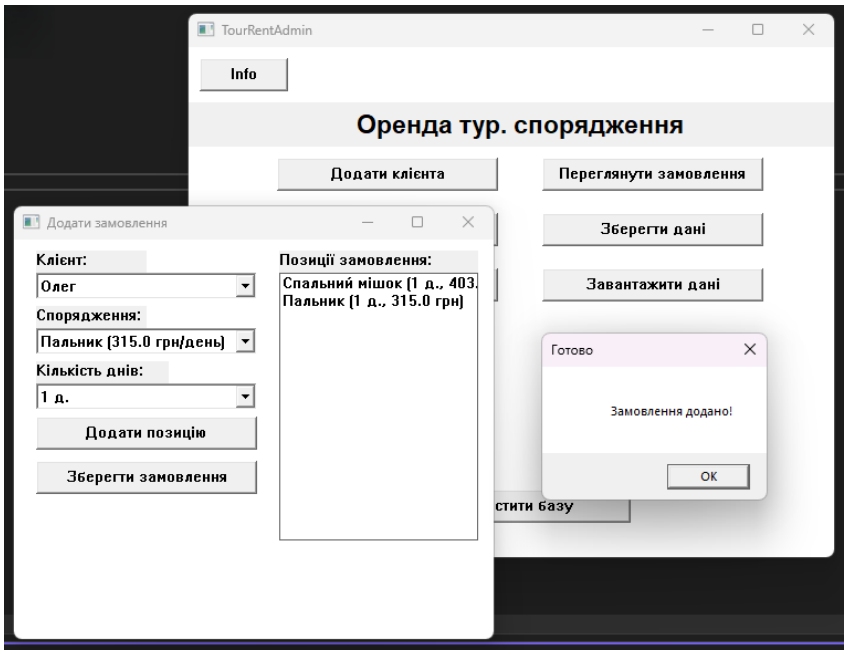


Рисунок 4.8 – Приклад екрану успішного оформлення оренди

### 4.3 Висновки

У четвертому розділі було проведено тестування функціональності програмних модулів системи для адміністрування пункту оренди туристичного спорядження. Здійснено перевірку коректності роботи основних функцій: облік спорядження, оформлення оренди, ведення бази клієнтів, експорт звітної інформації. Також було перевірено поведінку системи у випадку помилкових дій користувача або некоректного введення даних. Крім того, розроблено інструкцію користувача, яка містить опис технічних вимог, порядок запуску застосунку та сценарії роботи з основним функціоналом системи.

# ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи були розроблені модулі програмного забезпечення для адміністрування пункту оренди туристичного спорядження. Для реалізації програмного продукту використовувалося середовище розробки Microsoft Visual Studio з використанням мови програмування C++ та технології WinAPI. Розробка проводилась відповідно до методичних вказівок [12].

У ході виконання бакалаврської кваліфікаційної роботи було здійснено аналіз проблеми автоматизації обліку та обслуговування клієнтів у сфері оренди туристичного спорядження. Розглянуто наявні аналоги подібних програмних систем, виявлено їхні переваги та недоліки. На основі цього аналізу було сформульовано цілі і задачі бакалаврської кваліфікаційної роботи.

У ході розробки було обґрунтовано вибір мови програмування C++ як ефективного інструменту для створення настільних застосунків із мінімальними системними вимогами. Було також розроблено структуру програмного продукту з використанням базових засобів модульного програмування.

У межах бакалаврської кваліфікаційної оботи було виконано наступне:

- визначено функціональні вимоги до системи адміністрування оренди спорядження;
- реалізовано модулі для обліку туристичного спорядження, обробки замовлень, ведення клієнтської бази;
- створено зручний графічний інтерфейс користувача з інтуїтивною навігацією;
- проведено тестування програмного забезпечення у типових і граничних умовах;
- розроблено інструкцію користувача з технічними вимогами та описом основних сценаріїв взаємодії з програмним забезпеченням;

- побудовано UML-діаграми, що відображають архітектуру та логіку роботи системи.

Результати тестування підтвердили працездатність програмного продукту та відповідність функціоналу поставленим вимогам. Розроблений програмний застосунок може бути використано як основу для впровадження в реальних умовах діяльності пунктів оренди туристичного спорядження.

## ЛІТЕРАТУРА

1. Автоматизація технологічних процесів [Електронний ресурс] – Режим доступу до ресурсу: https://vukladach.pp.ua/MyWeb/manual/%D0%B5lektroenergetuka/Avtomotuzacia_tehnologihnuh_procesiv_i_sustemu_avtomatuhnogo_keryvanna/1/1.1.htm (дата звернення: 08.03.2025)

2. Автоматизація та комп'ютерно-інтегровані технології [Електронний ресурс] – Режим доступу до ресурсу: https://www.uzhnu.edu.ua/uk/program/39/avtomatizatsiya-ta-kompyuterno-integrovani-tehnologiji (дата звернення: 08.03.2025)

3. Проблема оренди туристичного спорядження [Електронний ресурс] – Режим доступу до ресурсу: https://dzvin-ski.com.ua/dodatkovo/pytannia-vidpovidi (дата звернення: 10.03.2025)

4. WinAPI [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/windows/win32/learnwin32/learn-to-program-for-windows (дата звернення: 15.03.2025)

5. Програмне забезпечення для комплексного управління підприємством [Електронний ресурс] – Режим доступу до ресурсу: https://www.netsoft.com.ua/articles-soft/bas-news/kompljeksnoje-upravljenije-prjedprijatijemU.html (дата звернення: 12.03.2025)

6. Booqable [Електронний ресурс] – Режим доступу до ресурсу: https://booqable.com/ (дата звернення: 16.03.2025)

7. Rentle [Електронний ресурс] – Режим доступу до ресурсу: https://admin.rentle.io/ (дата звернення: 16.03.2025)

8. OpenRMS [Електронний ресурс] – Режим доступу до ресурсу: https://qvistgaard.github.io/openrms/ (дата звернення: 17.03.2025)

9. Excel [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/excel (дата звернення: 17.03.2025)

10. Zoho Inventory [Електронний ресурс] – Режим доступу до ресурсу: https://www.zoho.com/inventory/ (дата звернення: 17.03.2025)

11. Графічний інтерфейс користувача [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/gui-tse/ (дата звернення: 19.03.2025)

12. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення». , уклад.: О.Н. Романюк, Г.О. Черноволик. – Вінниця: ВНТУ, 2022. – 51 с.

13. C++ [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/ (дата звернення: 21.03.2025)

14. UML diagram [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com/diagramming/what-is-a-uml-diagram/ (дата звернення: 23.03.2025)

15. Розробка алгоритмів [Електронний ресурс] – Режим доступу до ресурсу: https://iwanoff.inf.ua/programming_1_ua/LabTraining01.html (дата звернення: 25.03.2025)

16. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/ (дата звернення: 26.03.2025)

17. Draw IO [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/ (дата звернення: 01.04.2025)

18. СКБД [Електронний ресурс] – Режим доступу до ресурсу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%A1%D0%B0%D0%B2%D0%B0%D1%80%D0%B8%D0%BD_%D0%9B%D0%B5%D0%BF%D0%BA%D0%B8%D0%B9/teoretic/lec1.html (дата звернення: 03.04.2025)

19. Foundations of Software Testing ISTQB Certification: 4th Edition. Dorothy Graham, Rex Black, Erik van Veenendaal. 2019. 319p.

20. Failure Analysis. Fundamentals and Applications in Mechanical Components. Jose Luis Otegui. Springer. 2014. 411p.

21. Mastering Compatibility Testing. A Comprehensive Guide For Software QA Testing. Tailor Read. 2025. 284p.

22. Матюха О.В. ПРОГРАМНИЙ ЗАСТОСУНОК ДЛЯ АДМІНІСТРУВАННЯ ПУНКТУ ОРЕНДИ ТУРИСТИЧНОГО СПОРЯДЖЕННЯ / О.В. Матюха, Г.Б. Ракитянська // Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» , Вінниця, 2025 [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/24972

# ДОДАТКИ

# ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н.Романюк
"__" 03 2025 р.

**Технічне завдання**
**на бакалаврську кваліфікаційну роботу «Розробка програмного застосунку для адміністрування пункту оренди туристичного спорядження» за спеціальністю**
**121 Інженерія програмного забезпечення**

Керівник бакалаврської кваліфікаційної роботи:
_____к.т.н., доц. каф. ПЗ Г.Б. Ракитянська
"24 03 2025 р.
Виконав:
_____студент гр.1ПІ-21б О.В. Матюха
"24 "03 2025 р.

Вінниця – 2025 року

**1. Найменування та галузь застосування**

Бакалаврська кваліфікаційна робота: «Розробка програмного застосунку для адміністрування пункту оренди туристичного спорядження».

Галузь застосування – сфера надання послуг оренди туристичного спорядження.

**2. Підстава для розробки.**

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора ВНТУ про закріплення тем БКР.

**3. Мета та призначення розробки.**

Метою роботи є розробка та реалізація програмного застосунку для адміністрування пункту оренди туристичного спорядження.

Основними задачами є:

– проаналізувати предметну область та визначити вимоги до функціональності системи;

– розробити структуру бази даних, що відображає основні сутності орендного процесу;

– реалізувати графічний інтерфейс користувача засобами WinAPI;

– створити модулі для обробки замовлень, повернення спорядження, формування звітності;

– протестувати роботу застосунку в умовах моделювання реальної експлуатації;

– виконати економічне обґрунтування доцільності використання програмного продукту.

**4. Вихідні дані для проведення БДР**

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

**5. Технічні вимоги**

Середовища розробки – Visual Studio, мова розробки – C++, операційна система – Windows.

**6. Конструктивні вимоги**

Графічна та текстова документація повинна відповідати діючим стандартам України.

**7. Перелік технічної документації, що пред'являється по закінченню робіт:**

1. Пояснювальна записка до БКР;
2. Технічне завдання;
3. Лістинги програми.

**8. Вимоги до рівня уніфікації та стандартизації**

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

**9. Стадії та етапи розробки:**

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1 | Аналіз предметної області та визначення вимог до програмного забезпечення | 25.03.2025 – 05.04.2025 |
| 2 | Проєктування структури бази даних та моделі обліку інвентарю | 06.04.2025 – 17.04.2025 |
| 3 | Розробка користувацького інтерфейсу засобами WinAPI | 18.04.2025 – 01.05.2025 |
| 4 | Реалізація алгоритмів обробки оренди, повернення спорядження та формування звітів | 02.05.2025 – 13.05.2025 |
| 5 | Оформлення матеріалів до захисту БКР | 14.05.2025 – 30.05.2025 |

**10. Порядок контролю та прийняття**

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

## ДОДАТОК Б – ПЕРЕВІРКА НА ПЛАГІАТ
## ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка програмного застосунку для адміністрування пункту оренди туристичного спорядження»

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ Кафедра ПЗ, ФІТКІ, ІПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,7 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ракитянська Г.Б.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Матюха О.В.
(підпис) (прізвище, ініціали)

# ДОДАТОК В – ЛІСТИНГ ПРОГРАМИ

```
#include <windows.h>
#include <vector>
#include <string>
#include <cstdlib>
#include <ctime>
#include <sstream>
#include <iomanip>
#include <fstream>
#include <locale>
#include <codecvt>
#include <cmath>
#include <algorithm>
#include <map>

using namespace std;

// Допоміжні функції
void CenterWindow(HWND hwnd, int width, int height) {
    RECT rc;
    GetWindowRect(hwnd, &rc);

    int screenWidth = GetSystemMetrics(SM_CXSCREEN);
    int screenHeight = GetSystemMetrics(SM_CYSCREEN);

    int x = (screenWidth - width) / 2;
```

```
    int y = (screenHeight - height) / 2;


    SetWindowPos(hwnd, NULL, x, y, width, height, SWP_NOZORDER |
SWP_NOACTIVATE);

}


string WideStringToString(const wstring& wstr) {

    if (wstr.empty()) return string();

    int size_needed = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL,
0, NULL, NULL);

    string strTo(size_needed, 0);

    WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &strTo[0], size_needed,
NULL, NULL);

    return strTo;

}


wstring StringToWideString(const string& str) {

    if (str.empty()) return wstring();

    int size_needed = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);

    wstring wstrTo(size_needed, 0);

    MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstrTo[0], size_needed);

    return wstrTo;

}


void ShowCenteredMessageBox(HWND hwnd, const wstring& message, const wstring&
title, UINT uType = MB_OK) {

    RECT rcOwner, rcDlg, rc;

    GetWindowRect(hwnd, &rcOwner);
```

```
    HWND hTemp = CreateWindow(L"STATIC", message.c_str(), WS_POPUP, 0, 0, 0, 0,
hwnd, NULL, NULL, NULL);

    HDC hdc = GetDC(hTemp);

    HFONT hFont = (HFONT)SendMessage(hTemp, WM_GETFONT, 0, 0);

    SelectObject(hdc, hFont);


    SIZE size;

    GetTextExtentPoint32(hdc, message.c_str(), (int)message.length(), &size);

    ReleaseDC(hTemp, hdc);

    DestroyWindow(hTemp);


    rcDlg.left = rcDlg.top = 0;

    rcDlg.right = size.cx + 50;

    rcDlg.bottom = size.cy + 100;


    rc.left = rcOwner.left + (rcOwner.right - rcOwner.left - rcDlg.right) / 2;

    rc.top = rcOwner.top + (rcOwner.bottom - rcOwner.top - rcDlg.bottom) / 2;

    rc.right = rc.left + rcDlg.right;

    rc.bottom = rc.top + rcDlg.bottom;


    MSGBOXPARAMS mbp = { 0 };

    mbp.cbSize = sizeof(MSGBOXPARAMS);

    mbp.hwndOwner = hwnd;

    mbp.hInstance = NULL;

    mbp.lpszText = message.c_str();

    mbp.lpszCaption = title.c_str();

    mbp.dwStyle = uType | MB_USERICON;
```

```
    mbp.dwLanguageId = MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT);
    mbp.lpfnMsgBoxCallback = NULL;
    mbp.dwContextHelpId = 0;
    mbp.lpszIcon = NULL;
    mbp.dwStyle |= MB_SETFOREGROUND;

    MessageBoxIndirect(&mbp);
}

// Структури даних
struct Equipment {
    wstring name;
    double pricePerDay = 0.0;

    Equipment() : name(L""), pricePerDay(0.0) {}
};

struct Client {
    wstring name;
    wstring passport;
    wstring phone;

    Client() : name(L""), passport(L""), phone(L"") {}
};

struct OrderItem {
    Equipment equipment;
```

```
    int days = 0;

    OrderItem() : equipment(), days(0) {}
};

struct Order {
    Client client;
    vector<OrderItem> items;
    double totalPrice = 0.0;
    time_t startTime = 0;
    int durationDays = 0;
    bool isActive = false;

    Order() : client(), items(), totalPrice(0.0), startTime(0), durationDays(0), isActive(false) {}
};

struct DemandForecast {
    wstring equipmentName;
    double forecastValue;
    double popularityScore;
    double seasonalityFactor;
};

// Глобальні змінні
vector<Equipment> equipmentList;
vector<Client> clientList;
vector<Order> orderList;
```

```
vector<DemandForecast> forecastList;

// Елементи інтерфейсу
HWND hEquipmentCombo = nullptr;
HWND hClientCombo = nullptr;
HWND hDaysCombo = nullptr;
HWND hClientNameEdit = nullptr, hPassportEdit = nullptr, hPhoneEdit = nullptr;
HWND hOrderItemsList = nullptr;
HWND hAddItemBtn = nullptr;

// Ідентифікатори кнопок
#define BTN_ADD_CLIENT 2
#define BTN_SHOW_EQUIP 3
#define BTN_SHOW_CLIENT 4
#define BTN_SAVE_CLIENT 6
#define BTN_ADD_ORDER 7
#define BTN_SHOW_ORDERS 8
#define BTN_SAVE_ORDER 9
#define BTN_ADD_ITEM 10
#define BTN_SAVE_DATA 11
#define BTN_LOAD_DATA 12
#define BTN_INFO 13
#define BTN_CLEAR_DATA 14
#define BTN_SHOW_FORECAST 15

// Прототипи функцій
LRESULT CALLBACK MainWindowProc(HWND, UINT, WPARAM, LPARAM);
```

```
LRESULT CALLBACK AddClientProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK AddOrderProc(HWND, UINT, WPARAM, LPARAM);
void RefreshOrderCombos(HWND hwnd);
void AddItemToOrder(HWND hwnd);
void UpdateOrderTotal(HWND hwnd);
void SaveDataToFile(const wstring& filename);
void LoadDataFromFile(const wstring& filename);
void UpdateOrdersStatus();
void ClearData();
void ShowClients(HWND hwnd);
void ShowOrders(HWND hwnd);
vector<DemandForecast> CalculateDemandForecast(int forecastDays = 30);
void ShowForecast(HWND hwnd);

// Функції для відображення вікон
void ShowAddClientWindow(HINSTANCE hInstance) {
    HWND hwnd = CreateWindowEx(0, L"AddClientWindowClass", L"Додати клієнта",
        WS_OVERLAPPEDWINDOW & ~WS_THICKFRAME,
        0, 0, 300, 250,
        NULL, NULL, hInstance, NULL);
    if (hwnd) {
        CenterWindow(hwnd, 300, 250);
        ShowWindow(hwnd, SW_SHOW);
        UpdateWindow(hwnd);
    }
}
```

```
void ShowAddOrderWindow(HINSTANCE hInstance) {
    HWND hwnd = CreateWindowEx(0, L"AddOrderWindowClass", L"Додати замовлення",
        WS_OVERLAPPEDWINDOW & ~WS_THICKFRAME,
        0, 0, 450, 400,
        NULL, NULL, hInstance, NULL);
    if (hwnd) {
        CenterWindow(hwnd, 450, 400);
        ShowWindow(hwnd, SW_SHOW);
        UpdateWindow(hwnd);
    }
}

// Ініціалізація списку спорядження
void InitEquipmentList() {
    srand((unsigned)time(NULL));
    vector<wstring> names = {
        L"Намет", L"Спальний мішок", L"Килимок", L"Рюкзак", L"Термос",
        L"Газова плита", L"Ліхтарик", L"Пальник", L"Каструля", L"Сковорідка",
        L"Котелок", L"Мультитул", L"Компасс", L"Навігатор", L"Фільтр для води",
        L"Трекінгові палиці", L"Лижі", L"Шолом", L"Кішки", L"Мішок для сміття"
    };
    for (const auto& name : names) {
        Equipment eq;
        eq.name = name;
        eq.pricePerDay = 200 + rand() % 301;
        equipmentList.push_back(eq);
```

```
    }
}

// Оновлення статусів замовлень
void UpdateOrdersStatus() {
    time_t now = time(nullptr);
    for (auto& ord : orderList) {
        if (ord.isActive) {
            time_t endTime = ord.startTime + ord.durationDays * 24 * 3600;
            if (now >= endTime) {
                ord.isActive = false;
            }
        }
    }
}

// Оновлення комбо-боксів
void RefreshOrderCombos(HWND hwnd) {
    SendMessage(hClientCombo, CB_RESETCONTENT, 0, 0);
    for (const auto& cl : clientList) {
        SendMessage(hClientCombo, CB_ADDSTRING, 0, (LPARAM)cl.name.c_str());
    }

    SendMessage(hEquipmentCombo, CB_RESETCONTENT, 0, 0);
    for (const auto& eq : equipmentList) {
        wstringstream ss;
        ss << fixed << setprecision(1) << eq.pricePerDay;
```

```
        wstring entry = eq.name + L" (" + ss.str() + L" грн/день)";
        SendMessage(hEquipmentCombo, CB_ADDSTRING, 0, (LPARAM)entry.c_str());
    }

    SendMessage(hDaysCombo, CB_RESETCONTENT, 0, 0);
    for (int i = 1; i <= 7; i++) {
        wstring daysStr = to_wstring(i) + L" д.";
        SendMessage(hDaysCombo, CB_ADDSTRING, 0, (LPARAM)daysStr.c_str());
    }
    SendMessage(hDaysCombo, CB_SETCURSEL, 0, 0);
}

// Перегляд клієнтів
void ShowClients(HWND hwnd) {
    wstring list;
    for (const auto& cl : clientList) {
        list += L"Ім'я: " + cl.name + L", Паспорт: " + cl.passport +
            L", Телефон: " + cl.phone + L"\n";
    }
    if (list.empty()) list = L"Немає клієнтів.";
    ShowCenteredMessageBox(hwnd, list.c_str(), L"Список клієнтів", MB_OK);
}

// Перегляд замовлень
void ShowOrders(HWND hwnd) {
    wstring list;
    time_t now = time(nullptr);
```

```
    for (const auto& ord : orderList) {
        wstringstream ss;
        ss << fixed << setprecision(1) << ord.totalPrice;

        list += L"Клієнт: " + ord.client.name + L"\n";
        for (const auto& item : ord.items) {
            ss.str(L"");
            ss << fixed << setprecision(1) << item.equipment.pricePerDay * item.days;
            list += L"  - " + item.equipment.name + L" (" + to_wstring(item.days) + L" д., " +
ss.str() + L" грн)\n";
        }
        ss.str(L"");
        ss << fixed << setprecision(1) << ord.totalPrice;
        list += L"Загальна сума: " + ss.str() + L" грн\n";

        if (ord.isActive) {
            time_t endTime = ord.startTime + ord.durationDays * 24 * 3600;
            if (now < endTime) {
                time_t remaining = endTime - now;
                int days = static_cast<int>(remaining / (24 * 3600));
                remaining = remaining % (24 * 3600);
                int hours = static_cast<int>(remaining / 3600);
                remaining = remaining % 3600;
                int minutes = static_cast<int>(remaining / 60);

                wstring daysText = (days == 1) ? L"день" : (days > 1 && days < 5) ? L"дні" :
L"днів";
```

```
            wstring hoursText = (hours == 1) ? L"година" : (hours > 1 && hours < 5) ? L"години" : L"годин";
            wstring minutesText = (minutes == 1) ? L"хвилина" : (minutes > 1 && minutes < 5) ? L"хвилини" : L"хвилин";

            list += L"Залишилось: " + to_wstring(days) + L" " + daysText + L" " +
                to_wstring(hours) + L" " + hoursText + L" " +
                to_wstring(minutes) + L" " + minutesText + L"\n";
        }
        else {
            list += L"Час оренди вийшов!\n";
            const_cast<Order&>(ord).isActive = false;
        }
    }
    else {
        list += L"Замовлення завершено\n";
    }
    list += L"\n";
  }

  if (list.empty()) list = L"Немає замовлень.";
  ShowCenteredMessageBox(hwnd, list.c_str(), L"Список замовлень", MB_OK);
}

// Збереження даних у файл
void SaveDataToFile(const wstring& filename) {
  ofstream outFile(filename);
  if (!outFile) {
```

```
        ShowCenteredMessageBox(NULL, L"Помилка при відкритті файлу для запису",
L"Помилка", MB_OK | MB_ICONERROR);
        return;
    }

    outFile << "[Equipment]\n";
    for (const auto& eq : equipmentList) {
        outFile << WideStringToString(eq.name) << "|" << eq.pricePerDay << "\n";
    }

    outFile << "[Clients]\n";
    for (const auto& cl : clientList) {
        outFile << WideStringToString(cl.name) << "|"
            << WideStringToString(cl.passport) << "|"
            << WideStringToString(cl.phone) << "\n";
    }

    outFile << "[Orders]\n";
    for (const auto& ord : orderList) {
        outFile << WideStringToString(ord.client.name) << "|"
            << WideStringToString(ord.client.passport) << "|"
            << WideStringToString(ord.client.phone) << "\n";

        outFile << ord.items.size() << "\n";

        for (const auto& item : ord.items) {
            outFile << WideStringToString(item.equipment.name) << "|"
```

```
            << item.equipment.pricePerDay << "|"
            << item.days << "\n";
        }

        outFile << ord.totalPrice << "\n";
        outFile << ord.startTime << "|" << ord.durationDays << "|" << ord.isActive << "\n";
    }

    outFile.close();
    ShowCenteredMessageBox(NULL, L"Дані успішно збережено у файл", L"Успіх",
MB_OK | MB_ICONINFORMATION);
}

// Завантаження даних з файлу
void LoadDataFromFile(const wstring& filename) {
    ifstream inFile(filename);
    if (!inFile) {
        ShowCenteredMessageBox(NULL, L"Помилка при відкритті файлу для читання",
L"Помилка", MB_OK | MB_ICONERROR);
        return;
    }

    equipmentList.clear();
    clientList.clear();
    orderList.clear();

    string line;
    string section;
```

```
while (getline(inFile, line)) {
    if (line.empty()) continue;

    if (line == "[Equipment]") {
        section = "Equipment";
    }
    else if (line == "[Clients]") {
        section = "Clients";
    }
    else if (line == "[Orders]") {
        section = "Orders";
    }
    else {
        if (section == "Equipment") {
            size_t pos = line.find('|');
            if (pos != string::npos) {
                Equipment eq;
                eq.name = StringToWideString(line.substr(0, pos));
                eq.pricePerDay = stod(line.substr(pos + 1));
                equipmentList.push_back(eq);
            }
        }
        else if (section == "Clients") {
            vector<string> parts;
            size_t start = 0;
            size_t end = line.find('|');
            while (end != string::npos) {
```

```
            parts.push_back(line.substr(start, end - start));
            start = end + 1;
            end = line.find('|', start);
        }
        parts.push_back(line.substr(start));

        if (parts.size() >= 3) {
            Client cl;
            cl.name = StringToWideString(parts[0]);
            cl.passport = StringToWideString(parts[1]);
            cl.phone = StringToWideString(parts[2]);
            clientList.push_back(cl);
        }
    }
    else if (section == "Orders") {
        vector<string> clientParts;
        size_t start = 0;
        size_t end = line.find('|');
        while (end != string::npos) {
            clientParts.push_back(line.substr(start, end - start));
            start = end + 1;
            end = line.find('|', start);
        }
        clientParts.push_back(line.substr(start));

        if (clientParts.size() >= 3) {
            Order ord;
```

```
ord.client.name = StringToWideString(clientParts[0]);
ord.client.passport = StringToWideString(clientParts[1]);
ord.client.phone = StringToWideString(clientParts[2]);

getline(inFile, line);
int itemCount = stoi(line);

for (int i = 0; i < itemCount; i++) {
    getline(inFile, line);
    vector<string> itemParts;
    start = 0;
    end = line.find('|');
    while (end != string::npos) {
        itemParts.push_back(line.substr(start, end - start));
        start = end + 1;
        end = line.find('|', start);
    }
    itemParts.push_back(line.substr(start));

    if (itemParts.size() >= 3) {
        OrderItem item;
        item.equipment.name = StringToWideString(itemParts[0]);
        item.equipment.pricePerDay = stod(itemParts[1]);
        item.days = stoi(itemParts[2]);
        ord.items.push_back(item);
    }
}
```

```
getline(inFile, line);
ord.totalPrice = stod(line);

getline(inFile, line);
vector<string> orderParts;
start = 0;
end = line.find('|');
while (end != string::npos) {
    orderParts.push_back(line.substr(start, end - start));
    start = end + 1;
    end = line.find('|', start);
}
orderParts.push_back(line.substr(start));

if (orderParts.size() >= 3) {
    ord.startTime = stol(orderParts[0]);
    ord.durationDays = stoi(orderParts[1]);
    ord.isActive = stoi(orderParts[2]);
}
else {
    ord.startTime = time(nullptr);
    ord.durationDays = 0;
    ord.isActive = true;
}

orderList.push_back(ord);
```

```
            }
        }
    }
}

    inFile.close();
    ShowCenteredMessageBox(NULL, L"Дані успішно завантажено з файлу", L"Успіх",
MB_OK | MB_ICONINFORMATION);
}

// Очищення бази даних
void ClearData() {
    clientList.clear();
    orderList.clear();

    ofstream outFile("tour_rent_data.txt", ios::trunc);
    outFile.close();

    ShowCenteredMessageBox(NULL, L"Базу даних успішно очищено", L"Успіх",
MB_OK | MB_ICONINFORMATION);
}

// Додавання позиції до замовлення
void AddItemToOrder(HWND hwnd) {
    int equipIndex = static_cast<int>(SendMessage(hEquipmentCombo, CB_GETCURSEL,
0, 0));
    int daysIndex = static_cast<int>(SendMessage(hDaysCombo, CB_GETCURSEL, 0, 0));
    int days = daysIndex + 1;
```

```
    if (equipIndex >= 0 && days > 0) {
        OrderItem item;
        item.equipment = equipmentList[equipIndex];
        item.days = days;

        wstringstream ss;
        ss << fixed << setprecision(1) << item.equipment.pricePerDay * item.days;
        wstring entry = item.equipment.name + L" (" + to_wstring(item.days) + L" д., " +
ss.str() + L" грн)";

        SendMessage(hOrderItemsList, LB_ADDSTRING, 0, (LPARAM)entry.c_str());
        SendMessage(hDaysCombo, CB_SETCURSEL, 0, 0);
    }
    else {
        ShowCenteredMessageBox(hwnd, L"Будь ласка, виберіть спорядження",
L"Помилка", MB_OK);
    }
}

// Оновлення загальної суми замовлення
void UpdateOrderTotal(HWND hwnd) {
    int count = static_cast<int>(SendMessage(hOrderItemsList, LB_GETCOUNT, 0, 0));
    if (count > 0) {
        EnableWindow(GetDlgItem(hwnd, BTN_SAVE_ORDER), TRUE);
    }
    else {
        EnableWindow(GetDlgItem(hwnd, BTN_SAVE_ORDER), FALSE);
```

```
    }
}

// Прогнозування попиту
vector<DemandForecast> CalculateDemandForecast(int forecastDays) {
    vector<DemandForecast> forecasts;
    time_t now = time(nullptr);
    struct tm timeinfo;
    localtime_s(&timeinfo, &now);
    int currentMonth = timeinfo.tm_mon + 1; // 1-12

    // Визначення сезонних коефіцієнтів для кожного місяця
    map<int, double> seasonalityFactors = {
        {1, 0.6},  // Січень - низький сезон
        {2, 0.7},  // Лютий
        {3, 0.8},  // Березень
        {4, 1.1},  // Квітень
        {5, 1.4},  // Травень - початок високого сезону
        {6, 1.6},  // Червень
        {7, 1.8},  // Липень - пік сезону
        {8, 1.7},  // Серпень
        {9, 1.3},  // Вересень
        {10, 1.0}, // Жовтень
        {11, 0.7}, // Листопад
        {12, 0.5}  // Грудень - низький сезон
    };
```

```
// Аналіз історичних даних
map<wstring, int> itemCounts;
map<wstring, double> itemDays;
int totalOrders = 0;

for (const auto& order : orderList) {
    for (const auto& item : order.items) {
        itemCounts[item.equipment.name]++;
        itemDays[item.equipment.name] += item.days;
        totalOrders++;
    }
}

// Розрахунок прогнозу для кожного типу спорядження
for (const auto& eq : equipmentList) {
    DemandForecast forecast;
    forecast.equipmentName = eq.name;

    // Популярність (частка в загальній кількості замовлень)
    double popularity = totalOrders > 0 ?
        (double)itemCounts[eq.name] / totalOrders : 0.0;

    // Середня тривалість оренди
    double avgDays = itemCounts[eq.name] > 0 ?
        itemDays[eq.name] / itemCounts[eq.name] : 3.0;

    // Сезонний фактор
```

```
        double seasonality = seasonalityFactors[currentMonth];

        // Прогнозована кількість замовлень
        double baseDemand = popularity * 100; // Базовий попит
        forecast.forecastValue = round(baseDemand * seasonality * (forecastDays / 30.0));
        forecast.popularityScore = popularity;
        forecast.seasonalityFactor = seasonality;

        forecasts.push_back(forecast);
    }

    // Сортування за прогнозованим попитом (за зменшенням)
    sort(forecasts.begin(), forecasts.end(),
        [](const DemandForecast& a, const DemandForecast& b) {
            return a.forecastValue > b.forecastValue;
        });

    return forecasts;
}

void ShowForecast(HWND hwnd) {
    forecastList = CalculateDemandForecast(30); // Прогноз на 30 днів
    wstring forecastText = L"Прогноз попиту на наступні 30 днів:\n\n";

    for (const auto& forecast : forecastList) {
        wstringstream ss;
        ss << fixed << setprecision(1);
```

```
        ss << forecast.equipmentName << L": " << forecast.forecastValue
           << L" очікуваних замовлень\n";
        ss << L"   Популярність: " << forecast.popularityScore * 100 << L"%, "
           << L"Сезонність: " << forecast.seasonalityFactor << L"\n\n";
        forecastText += ss.str();
    }

    ShowCenteredMessageBox(hwnd, forecastText, L"Прогноз попиту", MB_OK);
}

// Головна функція програми
int WINAPI wWinMain(_In_ HINSTANCE hInstance, _In_opt_ HINSTANCE
hPrevInstance, _In_ PWSTR pCmdLine, _In_ int nCmdShow) {
    InitEquipmentList();

    WNDCLASS wc = {};
    wc.lpfnWndProc = MainWindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = L"MainWindowClass";
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    RegisterClass(&wc);

    WNDCLASS wcClient = {};
    wcClient.lpfnWndProc = AddClientProc;
    wcClient.hInstance = hInstance;
    wcClient.lpszClassName = L"AddClientWindowClass";
    wcClient.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
```

```
    RegisterClass(&wcClient);

    WNDCLASS wcOrder = {};
    wcOrder.lpfnWndProc = AddOrderProc;
    wcOrder.hInstance = hInstance;
    wcOrder.lpszClassName = L"AddOrderWindowClass";
    wcOrder.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    RegisterClass(&wcOrder);

    HWND hwnd = CreateWindowEx(0, L"MainWindowClass", L"TourRentAdmin",
        WS_OVERLAPPEDWINDOW, 0, 0, 600, 500,
        NULL, NULL, hInstance, NULL);
    if (hwnd) {
        CenterWindow(hwnd, 600, 500);
        ShowWindow(hwnd, nCmdShow);
        UpdateWindow(hwnd);
    }

    MSG msg = {};
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return 0;
}

// Вікно додавання клієнта
```

```
LRESULT CALLBACK AddClientProc(HWND hwnd, UINT uMsg, WPARAM wParam,
LPARAM lParam) {

   switch (uMsg) {

   case WM_CREATE:

      CreateWindow(L"STATIC", L"Ім'я:", WS_VISIBLE | WS_CHILD,

         50, 10, 100, 20, hwnd, NULL, NULL, NULL);

      hClientNameEdit = CreateWindow(L"EDIT", NULL, WS_VISIBLE | WS_CHILD |
WS_BORDER,

         50, 30, 180, 20, hwnd, NULL, NULL, NULL);



      CreateWindow(L"STATIC", L"Паспорт:", WS_VISIBLE | WS_CHILD,

         50, 50, 100, 20, hwnd, NULL, NULL, NULL);

      hPassportEdit = CreateWindow(L"EDIT", NULL, WS_VISIBLE | WS_CHILD |
WS_BORDER,

         50, 70, 180, 20, hwnd, NULL, NULL, NULL);



      CreateWindow(L"STATIC", L"Телефон:", WS_VISIBLE | WS_CHILD,

         50, 90, 100, 20, hwnd, NULL, NULL, NULL);

      hPhoneEdit = CreateWindow(L"EDIT", NULL, WS_VISIBLE | WS_CHILD |
WS_BORDER,

         50, 110, 180, 20, hwnd, NULL, NULL, NULL);



      CreateWindow(L"BUTTON", L"Зберегти", WS_VISIBLE | WS_CHILD,

         80, 160, 120, 30, hwnd, (HMENU)BTN_SAVE_CLIENT, NULL, NULL);

      return 0;



   case WM_COMMAND:

      if (LOWORD(wParam) == BTN_SAVE_CLIENT) {

         wchar_t name[100] = { 0 };
```

```
            wchar_t passport[100] = { 0 };
            wchar_t phone[100] = { 0 };
            GetWindowText(hClientNameEdit, name, 100);
            GetWindowText(hPassportEdit, passport, 100);
            GetWindowText(hPhoneEdit, phone, 100);

            Client cl;
            cl.name = name;
            cl.passport = passport;
            cl.phone = phone;

            clientList.push_back(cl);
            ShowCenteredMessageBox(hwnd, L"Клієнта додано!", L"Готово", MB_OK);
            DestroyWindow(hwnd);
        }
        return 0;
    }
    return DefWindowProc(hwnd, uMsg, wParam, lParam);
}


// Вікно додавання замовлення
LRESULT CALLBACK AddOrderProc(HWND hwnd, UINT uMsg, WPARAM wParam,
LPARAM lParam) {
    static vector<OrderItem> currentItems;

    switch (uMsg) {
```

```
    case WM_CREATE:

        CreateWindow(L"STATIC", L"Клієнт:", WS_VISIBLE | WS_CHILD,

            20, 10, 100, 20, hwnd, NULL, NULL, NULL);

        hClientCombo = CreateWindow(L"COMBOBOX", NULL, WS_VISIBLE |
WS_CHILD | CBS_DROPDOWNLIST,

            20, 30, 200, 200, hwnd, NULL, NULL, NULL);



        CreateWindow(L"STATIC", L"Спорядження:", WS_VISIBLE | WS_CHILD,

            20, 60, 100, 20, hwnd, NULL, NULL, NULL);

        hEquipmentCombo = CreateWindow(L"COMBOBOX", NULL, WS_VISIBLE |
WS_CHILD | CBS_DROPDOWNLIST,

            20, 80, 200, 200, hwnd, NULL, NULL, NULL);



        CreateWindow(L"STATIC", L"Кількість днів:", WS_VISIBLE | WS_CHILD,

            20, 110, 120, 20, hwnd, NULL, NULL, NULL);

        hDaysCombo = CreateWindow(L"COMBOBOX", NULL, WS_VISIBLE | WS_CHILD
| CBS_DROPDOWNLIST,

            20, 130, 200, 200, hwnd, NULL, NULL, NULL);



        hAddItemBtn = CreateWindow(L"BUTTON", L"Додати позицію", WS_VISIBLE |
WS_CHILD,

            20, 160, 200, 30, hwnd, (HMENU)BTN_ADD_ITEM, NULL, NULL);



        CreateWindow(L"STATIC", L"Позиції замовлення:", WS_VISIBLE | WS_CHILD,

            240, 10, 180, 20, hwnd, NULL, NULL, NULL);

        hOrderItemsList = CreateWindow(L"LISTBOX", NULL, WS_VISIBLE | WS_CHILD |
WS_BORDER | WS_VSCROLL | LBS_NOTIFY,

            240, 30, 180, 250, hwnd, NULL, NULL, NULL);
```

```
        CreateWindow(L"BUTTON", L"Зберегти замовлення", WS_VISIBLE | WS_CHILD |
WS_DISABLED,
            20, 200, 200, 30, hwnd, (HMENU)BTN_SAVE_ORDER, NULL, NULL);
        return 0;

    case WM_ACTIVATE:
        if (wParam != WA_INACTIVE) {
            RefreshOrderCombos(hwnd);
            currentItems.clear();
            SendMessage(hOrderItemsList, LB_RESETCONTENT, 0, 0);
            UpdateOrderTotal(hwnd);
        }
        return 0;

    case WM_COMMAND:
        switch (LOWORD(wParam)) {
        case BTN_ADD_ITEM:
            AddItemToOrder(hwnd);
            UpdateOrderTotal(hwnd);
            break;

        case BTN_SAVE_ORDER: {
            int clientIndex = static_cast<int>(SendMessage(hClientCombo, CB_GETCURSEL,
0, 0));
            int itemCount = static_cast<int>(SendMessage(hOrderItemsList, LB_GETCOUNT,
0, 0));

            if (clientIndex >= 0 && itemCount > 0) {
```

```
Order ord;
ord.client = clientList[clientIndex];
ord.totalPrice = 0;
ord.startTime = time(nullptr);
ord.isActive = true;
ord.durationDays = 0;

for (int i = 0; i < itemCount; i++) {
    wchar_t buffer[256] = { 0 };
    SendMessage(hOrderItemsList, LB_GETTEXT, i, (LPARAM)buffer);

    for (const auto& eq : equipmentList) {
        if (wstring(buffer).find(eq.name) != wstring::npos) {
            OrderItem item;
            item.equipment = eq;
            size_t pos = wstring(buffer).find(L" (");
            if (pos != wstring::npos) {
                size_t endPos = wstring(buffer).find(L" д.", pos);
                if (endPos != wstring::npos) {
                    wstring daysStr = wstring(buffer).substr(pos + 2, endPos - pos - 2);
                    item.days = _wtoi(daysStr.c_str());
                }
            }
            ord.items.push_back(item);
            ord.totalPrice += item.equipment.pricePerDay * item.days;

            if (item.days > ord.durationDays) {
```

```
                    ord.durationDays = item.days;
                }
                break;
            }
        }
    }

    orderList.push_back(ord);
    ShowCenteredMessageBox(hwnd, L"Замовлення додано!", L"Готово",
MB_OK);
    DestroyWindow(hwnd);
}
else {
    ShowCenteredMessageBox(hwnd, L"Будь ласка, виберіть клієнта та додайте
хоча б одну позицію.",
        L"Помилка", MB_OK);
}
break;
}
}
return 0;
}
return DefWindowProc(hwnd, uMsg, wParam, lParam);
}

// Головне вікно програми
LRESULT CALLBACK MainWindowProc(HWND hwnd, UINT uMsg, WPARAM
wParam, LPARAM lParam) {
```

```
static HINSTANCE hInst;
switch (uMsg) {
case WM_CREATE: {
    hInst = ((LPCREATESTRUCT)lParam)->hInstance;
    SetTimer(hwnd, 1, 60000, NULL);

    CreateWindow(L"BUTTON", L"Info", WS_VISIBLE | WS_CHILD,
        10, 10, 80, 30, hwnd, (HMENU)BTN_INFO, hInst, NULL);

    CreateWindow(L"STATIC", L"Оренда тур. спорядження",
        WS_VISIBLE | WS_CHILD | SS_CENTER | SS_CENTERIMAGE,
        0, 50, 600, 40, hwnd, NULL, hInst, NULL);

    HFONT hFont = CreateFont(28, 0, 0, 0, FW_BOLD, FALSE, FALSE, FALSE,
        DEFAULT_CHARSET, OUT_OUTLINE_PRECIS, CLIP_DEFAULT_PRECIS,
        CLEARTYPE_QUALITY, VARIABLE_PITCH, L"Arial");
    HWND hTitle = GetDlgItem(hwnd, NULL);
    SendMessage(hTitle, WM_SETFONT, (WPARAM)hFont, TRUE);

    int buttonWidth = 200;
    int buttonHeight = 30;
    int margin = 20;
    int startY = 100;
    int column1 = (600 / 2) - buttonWidth - margin;
    int column2 = (600 / 2) + margin;

    CreateWindow(L"BUTTON", L"Додати клієнта", WS_VISIBLE | WS_CHILD,
```

```
        column1, startY, buttonWidth, buttonHeight, hwnd, (HMENU)BTN_ADD_CLIENT,
hInst, NULL);

    CreateWindow(L"BUTTON", L"Додати замовлення", WS_VISIBLE | WS_CHILD,

        column1, startY + 50, buttonWidth, buttonHeight, hwnd,
(HMENU)BTN_ADD_ORDER, hInst, NULL);

    CreateWindow(L"BUTTON", L"Переглянути клієнтів", WS_VISIBLE | WS_CHILD,

        column1, startY + 100, buttonWidth, buttonHeight, hwnd,
(HMENU)BTN_SHOW_CLIENT, hInst, NULL);



    CreateWindow(L"BUTTON", L"Переглянути замовлення", WS_VISIBLE |
WS_CHILD,

        column2, startY, buttonWidth, buttonHeight, hwnd,
(HMENU)BTN_SHOW_ORDERS, hInst, NULL);

    CreateWindow(L"BUTTON", L"Зберегти дані", WS_VISIBLE | WS_CHILD,

        column2, startY + 50, buttonWidth, buttonHeight, hwnd,
(HMENU)BTN_SAVE_DATA, hInst, NULL);

    CreateWindow(L"BUTTON", L"Завантажити дані", WS_VISIBLE | WS_CHILD,

        column2, startY + 100, buttonWidth, buttonHeight, hwnd,
(HMENU)BTN_LOAD_DATA, hInst, NULL);



    CreateWindow(L"BUTTON", L"Очистити базу", WS_VISIBLE | WS_CHILD,

        200, 400, 200, 30, hwnd, (HMENU)BTN_CLEAR_DATA, hInst, NULL);



    // Додавання кнопки для відображення прогнозу

    CreateWindow(L"BUTTON", L"Показати прогноз", WS_VISIBLE | WS_CHILD,

        column2, startY + 150, buttonWidth, buttonHeight, hwnd,
(HMENU)BTN_SHOW_FORECAST, hInst, NULL);



    return 0;

}
```

```
    case WM_TIMER:
        UpdateOrdersStatus();
        break;

    case WM_COMMAND:
        switch (LOWORD(wParam)) {
        case BTN_INFO:
            ShowCenteredMessageBox(hwnd,
                L"TourRentAdmin\nВерсія: 1.0.0.1\nРозробник: Матюха О.В. 1ПІ-21Б\nВсі права захищені",
                L"Інформація",
                MB_OK | MB_ICONINFORMATION);
            break;
        case BTN_ADD_CLIENT:
            ShowAddClientWindow(hInst);
            break;
        case BTN_ADD_ORDER:
            ShowAddOrderWindow(hInst);
            break;
        case BTN_SHOW_CLIENT:
            ShowClients(hwnd);
            break;
        case BTN_SHOW_ORDERS:
            ShowOrders(hwnd);
            break;
        case BTN_SAVE_DATA:
            SaveDataToFile(L"tour_rent_data.txt");
```

```
            break;
        case BTN_LOAD_DATA:
            LoadDataFromFile(L"tour_rent_data.txt");
            break;
        case BTN_CLEAR_DATA:
            if (MessageBox(hwnd, L"Ви впевнені, що хочете очистити всю базу даних?\nЦю дію не можна скасувати.",
                L"Підтвердження", MB_YESNO | MB_ICONWARNING) == IDYES) {
                ClearData();
            }
            break;
        case BTN_SHOW_FORECAST:
            ShowForecast(hwnd);
            break;
        }
        return 0;
    }
    return DefWindowProc(hwnd, uMsg, wParam, lParam);
                                                    }
```

# ДОДАТОК Г – ІЛЮСТРАТИВНА ЧАСТИНА

## ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АДМІНІСТРУВАННЯ ПУНКТУ ОРЕНДИ ТУРИСТИЧНОГО СПОРЯДЖЕННЯ

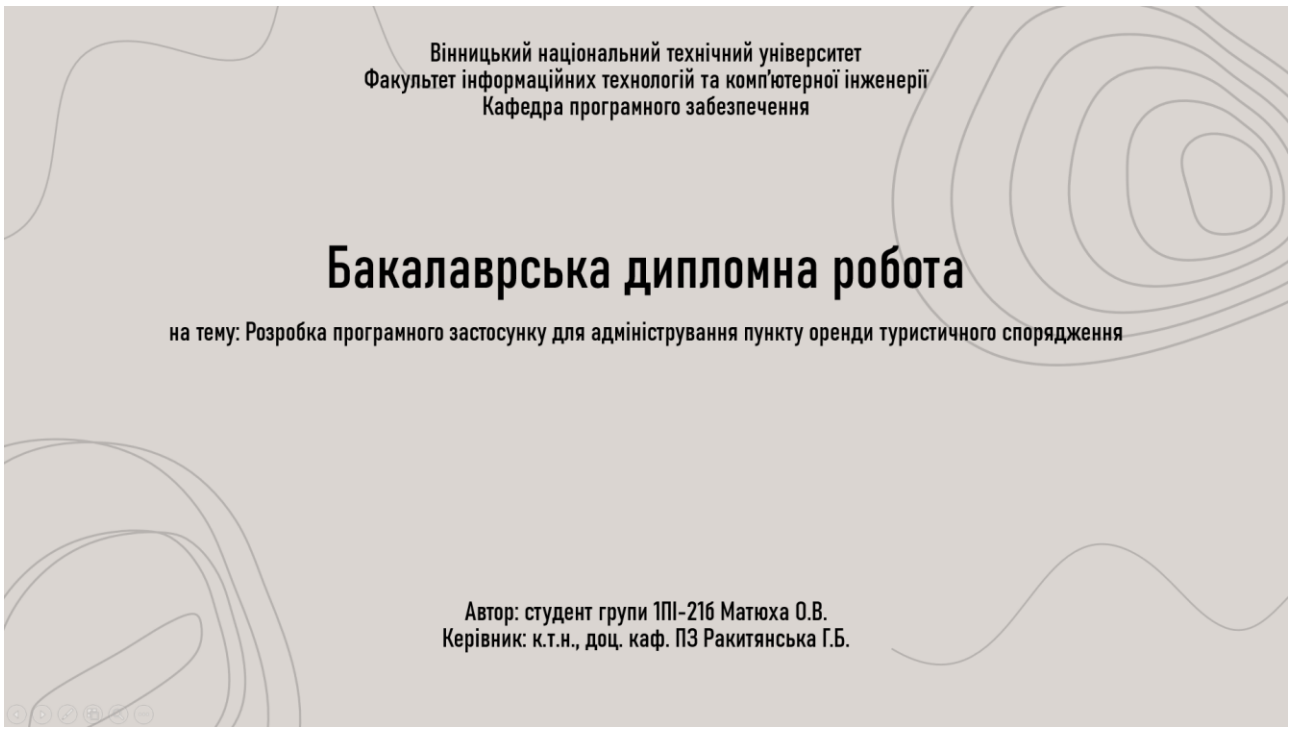


Рисунок Г.1 – Титульний слайд

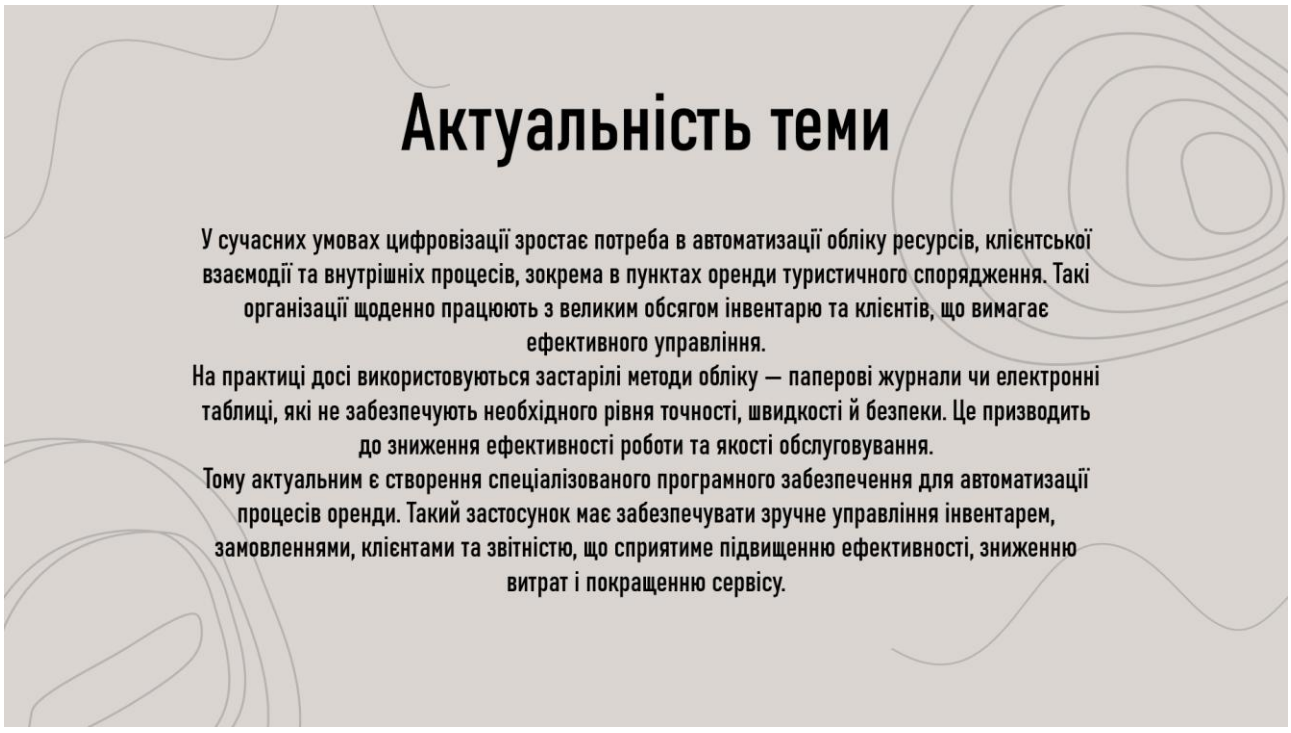


Рисунок Г.2 – Актуальність теми

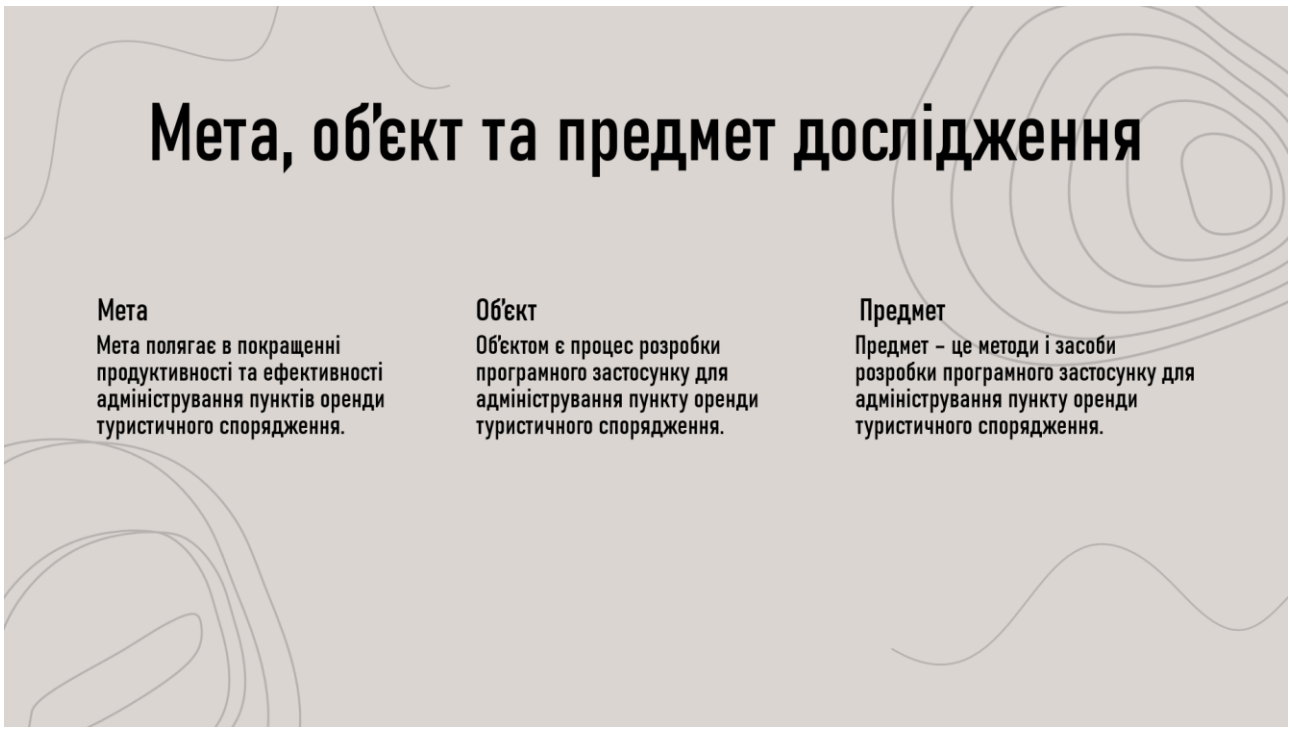


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

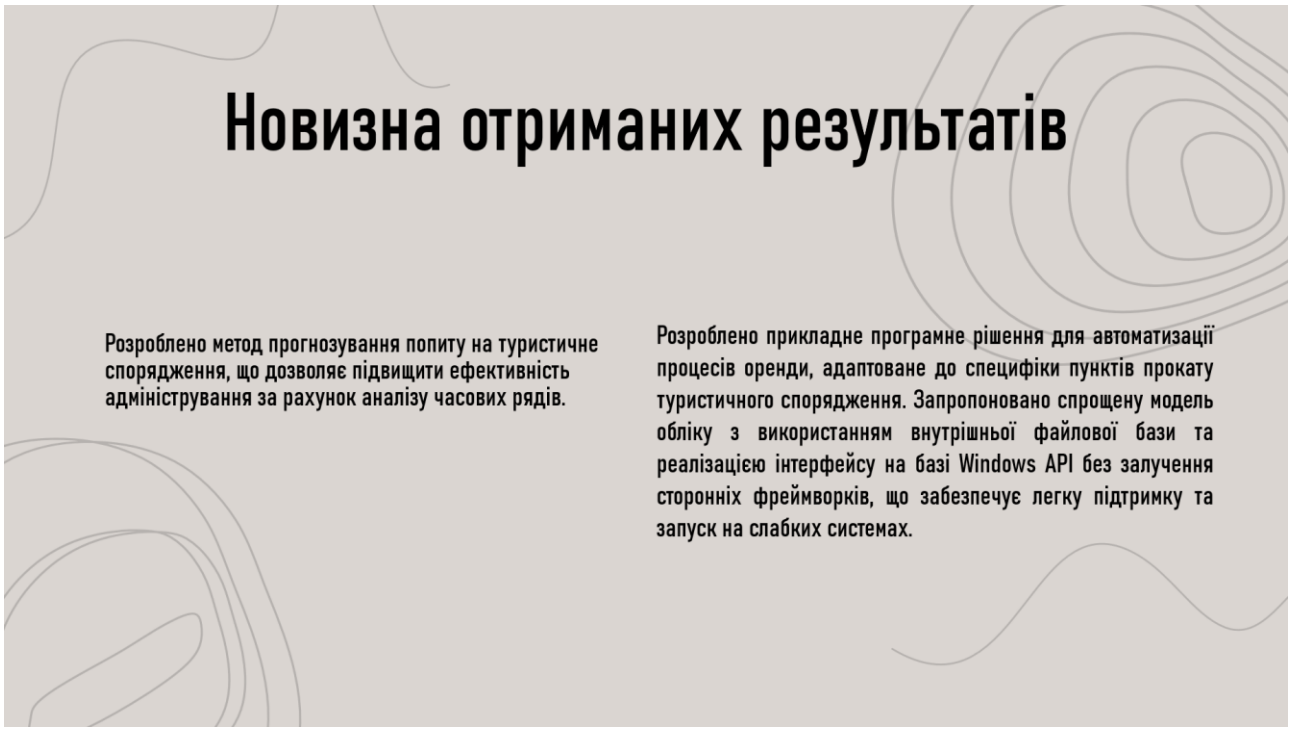


Рисунок Г.4 – Новизна отриманих результатів

## Порівняльний аналіз аналогів

| Критерії | Booqable | Rentle | OpenRMS | Excel | Zoho Inventory | Власна розробка |
|---|---|---|---|---|---|---|
| Автоматизація процесів оренди | + | + | + | - | - | + |
| Підтримка обліку спорядження | + | + | + | + | + | + |
| Зручність інтерфейсу | - | - | - | + | + | + |
| Можливість онлайн-роботи | + | + | - | - | + | + |
| Наявність української локалізації | - | - | - | + | + | + |
| Можливість кастомізації | - | - | - | + | - | + |
| Орієнтованість на туристичне спорядження | + | + | + | - | - | + |
| Загальна оцінка | 70% | 60% | 50% | 40% | 55% | 100% |

Рисунок Г.5 – Порівняльний аналіз аналогів

## Розробка алгоритмів роботи системи

На рисунку представлено загальний алгоритм роботи програмного застосунку для адміністрування пункту оренди туристичного спорядження

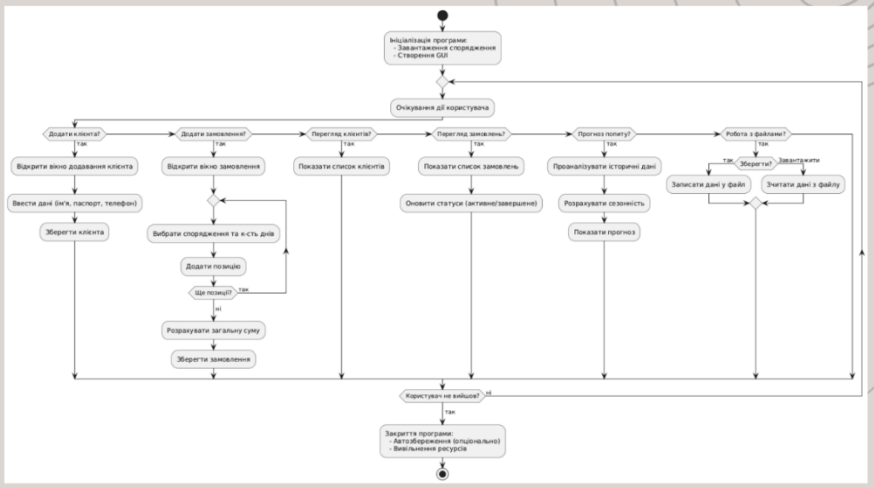


Рисунок Г.6 – Розробка алгоритмів системи – Загальний алгоритм роботи програмного застосунку

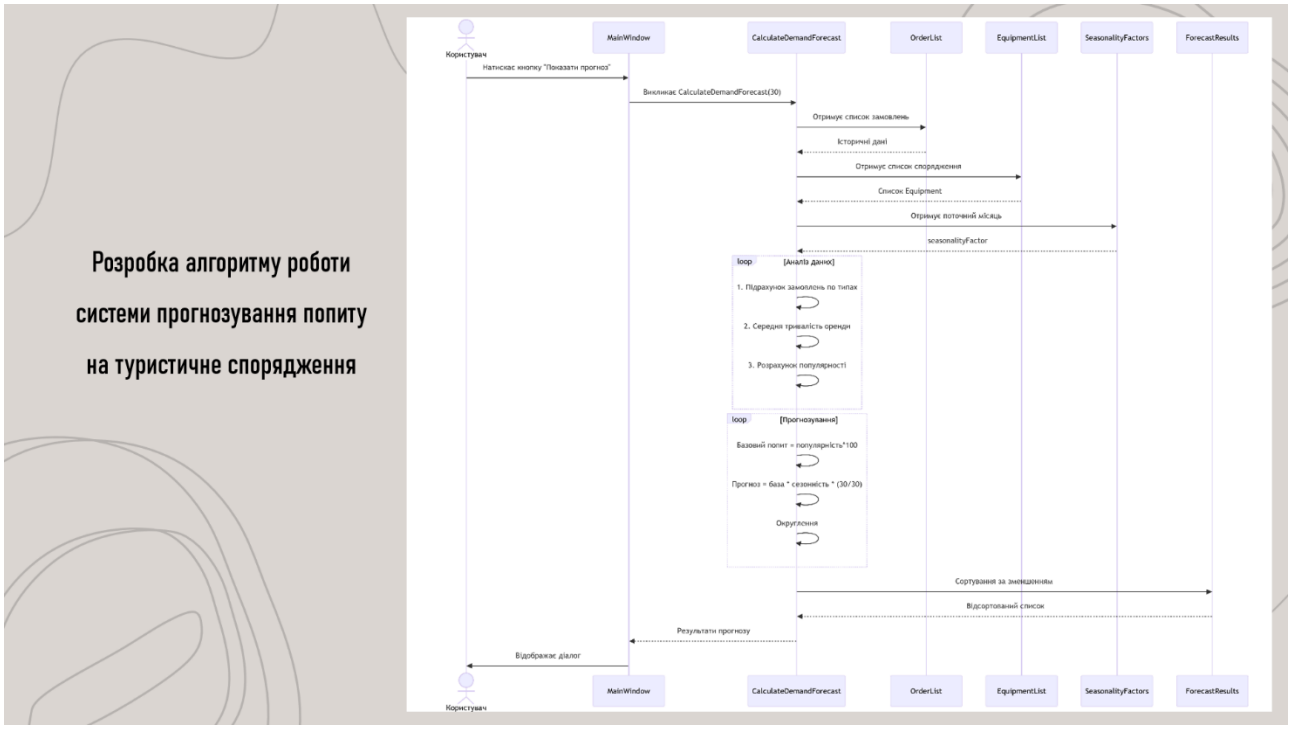


Рисунок Г.7 – Розробка алгоритмів системи – Алгоритм роботи системи прогнозування попиту на туристичне спорядження

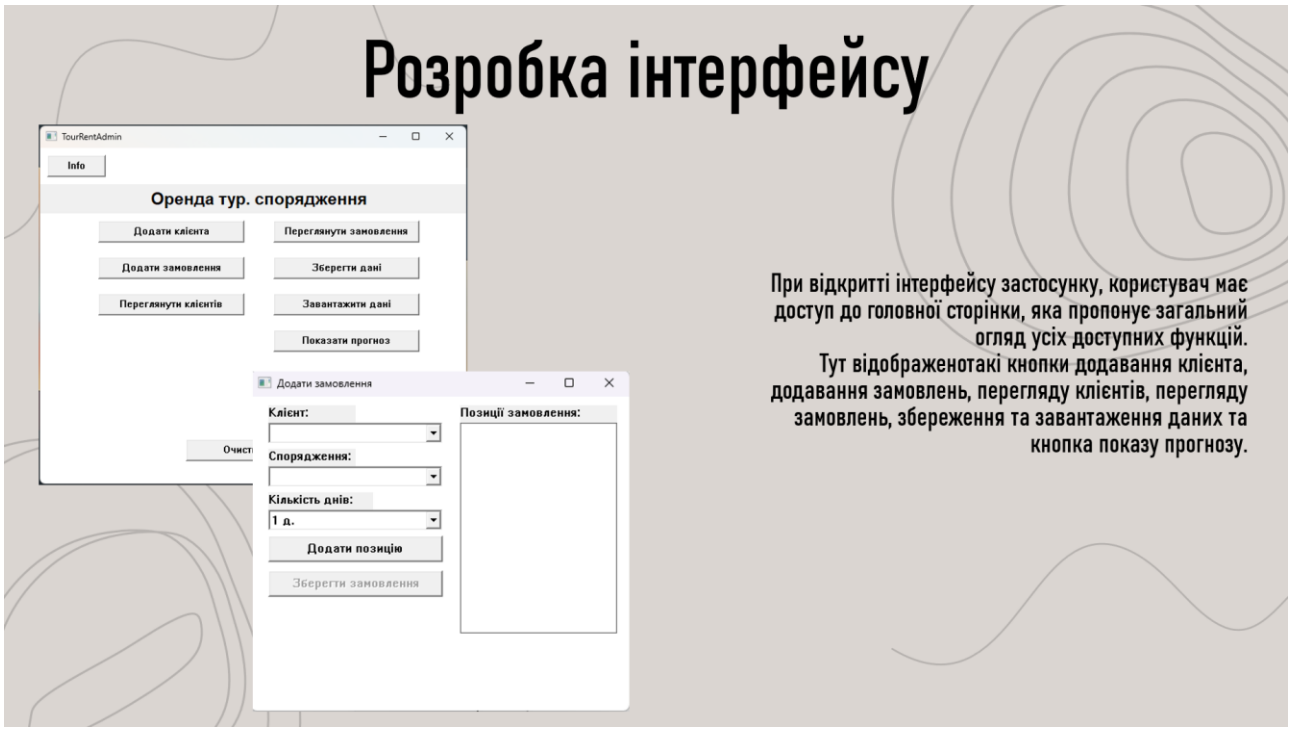


Рисунок Г.8 – Розробка інтерфейсу

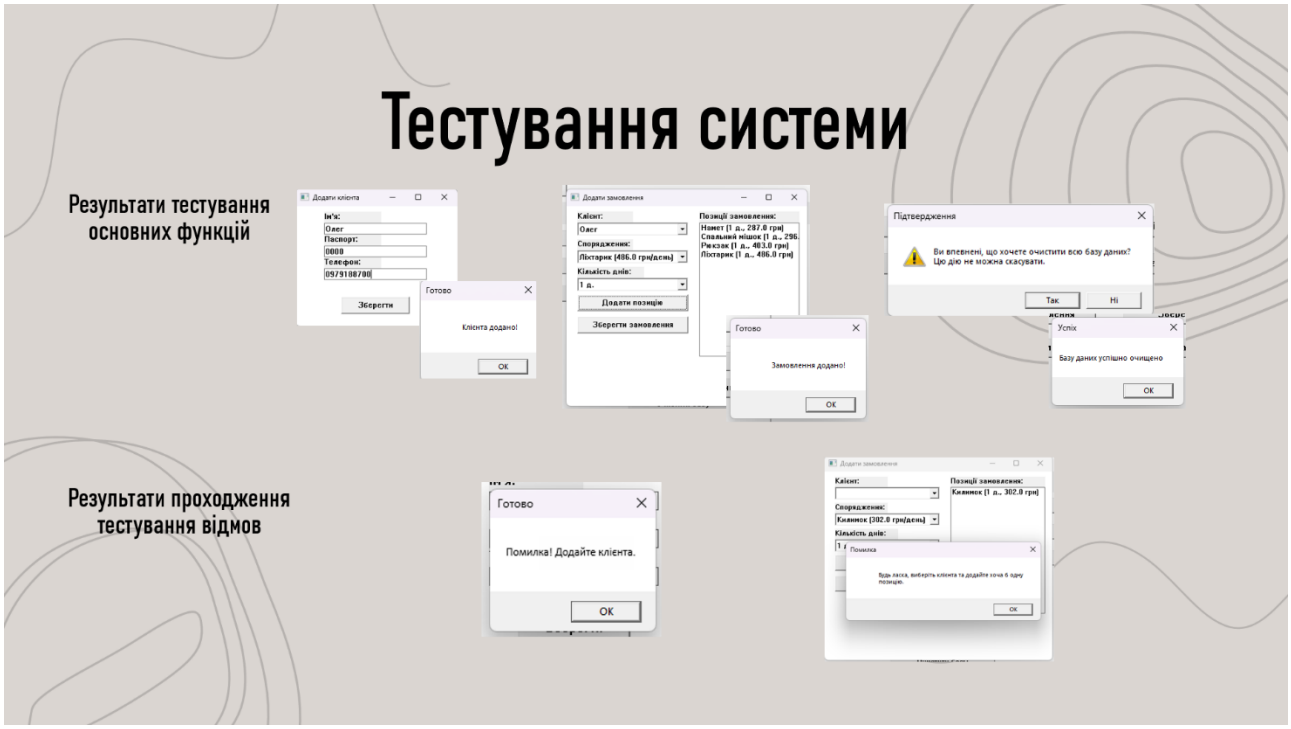


Рисунок Г.9 – Тестування системи

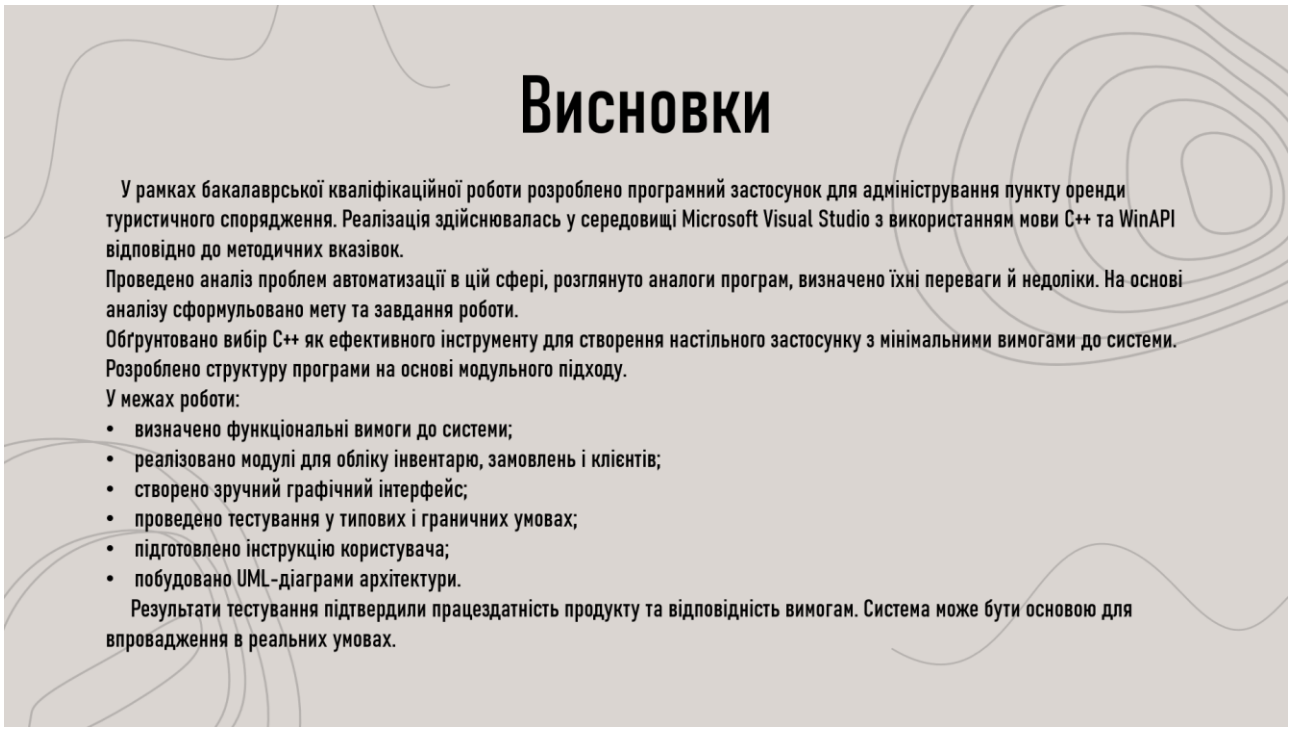


Рисунок Г.10 – Висновки