НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_____Комп'ютерних наук_____

(назва кафедри)

_____ ___Голуб Б. Л.___

(підпис) (ПІБ)

"_2_" __червня__ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

"Програмне забезпечення інформаційної системи управління перевезеннями пасажирів таксі"

Спеціальність F2 – «Інженерія програмного забезпечення»

Гарант освітньої програми

_____к.н.т., доцент_____ _____ __Вайганг Г.О._____

(наукова ступінь та вчене звання) (підпис) (ПІБ)

Керівник Бакалаврської кваліфікаційної роботи

_____ст. викладач_____ _____ _Міловідов Ю. О.

(наукова ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Ткаченко Дмитро Олександрович

(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

Комп'ютерних наук

(назва кафедри)

_____ Голуб Б.Л.

(підпис) (ПІБ)

"16" грудня 2024 р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Ткаченку Дмитру Олександровичу

(прізвище, ім'я, по батькові)

Спеціальність F2 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи Програмне забезпечення Інформаційної системи управління первезенням пасажирів таксі

Затверджена наказом ректора НУБіП України від "16" грудня 2024 р. №2249 "С"

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Опис предмету дослідження, опис програмного забезпечення

Перелік питань, які потрібно розробити:

Системний аналіз предметної області

Аналіз предметної області

Розробка програмного забезпечення

Рекомендації щодо впровадження та експлуатації системи

Дата видачі завдання "16" грудня 2024 р.

Керівник Бакалаврської кваліфікаційної роботи

ст. викладач _____ Міловідов Ю. О.

(наукова ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання

_____ Ткаченко Дмитро Олександрович

(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

– API – інтерфейс прикладного програмування (Application Programming Interface).

– UI – інтерфейс користувача (User Interface).

– GUI – графічний інтерфейс користувача (Graphical User Interface).

– HTTPS – захищений протокол передавання гіпертексту (Hypertext Transfer Protocol Secure).

– Docker – програмна платформа для контейнеризації додатків.

– .exe – виконуваний файл операційної системи Windows.

– .dll – динамічна бібліотека, що підключається під час виконання (Dynamic-Link Library).

– .txt – текстовий файл для зберігання структурованих або табличних даних.

– CRUD – набір базових операцій з даними: створення (Create), читання (Read), оновлення (Update), видалення (Delete).

– SQL – мова структурованих запитів до баз даних (Structured Query Language).

– JSON – формат обміну даними у вигляді тексту (JavaScript Object Notation).

– .yml – конфігураційний файл YAML, що застосовується для опису параметрів сервісів (у Docker).

– .ps1 – сценарій PowerShell для автоматизації операцій у Windows.

– .sh – скрипт оболонки Unix/Linux.

– GET, POST – основні методи HTTP-запитів: отримання та передавання даних.

– Admin.exe – виконуваний файл адміністративного інтерфейсу.

– Order.exe – виконуваний файл клієнтського інтерфейсу для створення замовлень.

– clients.txt, orders.txt, drivers.txt, ratings.txt, feedbacks.txt – текстові файли, що виконують роль локального сховища даних.

– LTS – версія з тривалою підтримкою (Long-Term Support).

ВСТУП

У контексті цифрової трансформації транспортної галузі дедалі більшої важливості набуває автоматизація процесів управління пасажирськими перевезеннями, зокрема у сфері таксомоторного обслуговування. Зростання попиту на мобільні сервіси, вимоги споживачів до швидкого реагування та доступності електронних каналів взаємодії зумовлюють необхідність розробки спеціалізованого програмного забезпечення. Такі системи мають забезпечувати централізоване керування замовленнями, моніторинг транспортних засобів, збереження історичних даних про перевезення, а також адміністрування клієнтської інформації в режимі реального часу.

Сучасні методи інженерії програмного забезпечення, зокрема об’єктно-орієнтований підхід та використання мови UML для моделювання системи, дозволяють підвищити якість архітектурних рішень та забезпечити модульність і масштабованість майбутнього застосунку [5]. Відповідно до стандартів ISO/IEC 12207 та ISO/IEC/IEEE 29148, формалізація вимог і побудова моделей системної поведінки є необхідними етапами життєвого циклу інформаційної системи [3], [10].

Програмна реалізація системи здійснюється мовою програмування C# із використанням технології Windows Forms (WinForms), яка забезпечує розробку інтерактивного графічного інтерфейсу користувача для операційної системи Windows. Використання WinForms дозволяє реалізувати модульну структуру інтерфейсу, інтегрувати функціональні компоненти управління замовленнями, адміністративними панелями та моніторингом. Для зберігання й обробки даних застосовується реляційна база даних SQL Server, що забезпечує цілісність, консистентність і високу швидкодію доступу до логістичної інформації.

Об’єкт дослідження: процес автоматизованого управління перевезенням пасажирів у рамках служби таксі.

Предмет дослідження: методи моделювання інформаційних систем, алгоритми обробки замовлень, архітектура та компоненти програмного забезпечення служби таксі.

Мета роботи: розробка та реалізація інформаційної системи управління перевезенням пасажирів таксі з використанням об’єктно-орієнтованого підходу та засобів C#, що забезпечує обробку замовлень, керування транспортом, взаємодію з клієнтами та адміністрування процесів у реальному часі.

Для досягнення мети необхідно вирішити такі **завдання:**

1. провести системний аналіз предметної області та сформулювати функціональні вимоги до інформаційної системи.

2. Побудувати UML-діаграми (класів, компонентів, прецедентів, послідовностей) для формалізації структури системи [5].

3. Розробити архітектуру інформаційної системи із виділенням основних модулів.

4. Реалізувати графічний інтерфейс користувача засобами SQL Server [9].

5. Спроєктувати та реалізувати реляційну базу даних для зберігання інформації про замовлення, клієнтів і водіїв.

6. Виконати тестування працездатності системи, перевірити відповідність функціональним вимогам.

7. Надати рекомендації щодо впровадження системи, масштабування та навчання персоналу.

Наукова новизна полягає в інтеграції об’єктно-орієнтованого підходу з принципами управління транспортними операціями, що дозволяє створити масштабовану, гнучку та підтримувану інформаційну систему для автоматизації процесів перевезення пасажирів таксі з урахуванням локальних потреб.

Практичне значення роботи: розроблена інформаційна система може бути впроваджена у реальні підприємства сфери таксомоторних перевезень для підвищення ефективності операційної діяльності, зменшення витрат на обслуговування замовлень, поліпшення клієнтського досвіду та забезпечення прозорості управлінських рішень. Система орієнтована на малий та середній бізнес і адаптована до специфіки українського ринку.

Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

– У першому розділі подано системний аналіз предметної області, визначено вимоги до програмного забезпечення, здійснено огляд аналогів та обґрунтовано доцільність розробки.

– У другому розділі здійснено моделювання системи за допомогою UML, розроблено архітектуру та логічну структуру даних.

– У третьому розділі наведено реалізацію інформаційної системи: розробку інтерфейсу, бази даних, алгоритмів обробки запитів, а також результати тестування та аналізу працездатності.

Робота завершується висновками та рекомендаціями щодо подальшого розвитку і впровадження системи.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Опис предметної області

У сфері пасажирських перевезень служби таксі виконують критично важливу роль у забезпеченні мобільності населення. Стрімке зростання попиту на цифрові сервіси, необхідність оперативного реагування на запити клієнтів, а також підвищення вимог до ефективності використання транспортних ресурсів обумовлюють актуальність автоматизації процесів управління службою таксі. В таких умовах програмне забезпечення має забезпечити централізоване керування всіма етапами обслуговування – від реєстрації замовлення до завершення поїздки з фіксацією її параметрів у базі даних.

Предметна область інформаційної системи управління перевезенням пасажирів таксі охоплює сукупність логічно взаємопов’язаних компонентів, які забезпечують інтегровану взаємодію між користувачем (пасажиром), водієм, адміністратором та внутрішніми модулями програмної системи. Основними процесами є ініціація замовлення клієнтом, обробка цього замовлення диспетчерською логікою, визначення координат і маршруту, призначення водія та передача статусу виконання поїздки.

Функціональна схема предметної області наведена на рисунку 1.1. У її центрі знаходиться інтерфейс користувача, який забезпечує точку взаємодії між системою та ключовими ролями – клієнтом, водієм і адміністратором. Саме через цей інтерфейс ініціюється передача даних між компонентами та здійснюється візуалізація поточного стану замовлень.

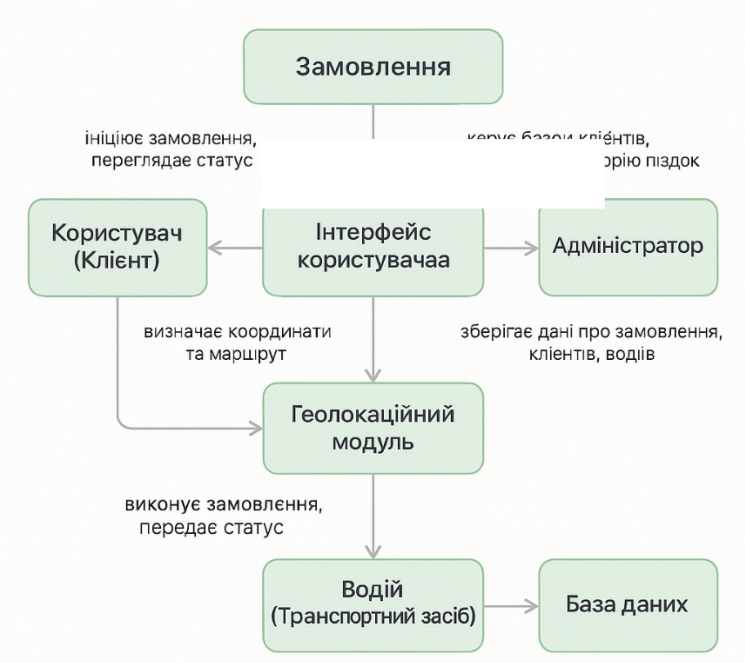


Рисунок 1.1 – Структурна схема предметної області служби таксі

Згідно з рисунком 1.1, у центрі функціональної моделі знаходиться інтерфейс користувача, через який здійснюється передача запитів від клієнта та адміністратора до інформаційної системи. Користувач ініціює замовлення, яке обробляється системою відповідно до поточного розташування клієнта та доступності транспортних засобів. Визначення координат, побудова маршруту та контроль переміщення водія виконується через геолокаційний модуль, який передає координати до інтерфейсу у реальному часі.

Адміністратор здійснює контроль за актуальністю даних у базі, керує реєстрацією водіїв, перевіряє статус виконання замовлень та формує статистичну інформацію. Водій, у свою чергу, отримує замовлення через мобільний або стаціонарний додаток, передає статус виконання й оновлює координати у геолокаційному модулі [9]. Відповідно до попередньої інформації доцільно буде розглянути таблицю 1.1 у якій представлені основні об'єкти предметної області та їхні функції.

Таблиця 1.1 – Основні об'єкти предметної області та їхні функції

№	Об'єкт	Функціональне призначення
1	Користувач (Клієнт)	Ініціює замовлення, переглядає статус, взаємодіє з інтерфейсом

Продовження таблиці 1.1

2	Водій	Отримує замовлення, виконує перевезення, передає статус виконання
3	Адміністратор	Керує базою клієнтів і водіїв, переглядає історію поїздок, генерує звіти
4	Інтерфейс користувача	Забезпечує доступ до функціональності системи для клієнтів та адміністраторів
5	Геолокаційний модуль	Визначає маршрут, координати, підтримує інтеграцію з картографічними сервісами
6	База даних	Зберігає інформацію про замовлення, користувачів, транспортні засоби

Предметна область управління перевезенням пасажирів у службі таксі включає розподілену систему з багатьма учасниками і компонентами. Основна складність полягає у необхідності забезпечити синхронну взаємодію між клієнтом, водієм та диспетчерською, обробку великого обсягу подій у режимі реального часу, а також підтримку збереження й обробки даних згідно з вимогами до надійності та цілісності [5], [9].

Розуміння структури предметної області є необхідним для подальшого моделювання системи та визначення функціональних вимог, що будуть сформовані у наступному підрозділі.

1.2 Аналіз джерел та існуючих рішень

Інформаційна система має забезпечувати автоматизовану обробку основних логістичних процесів, включаючи створення, виконання та моніторинг замовлень, взаємодію з клієнтами, обробку геолокаційних даних і адміністрування облікових записів. Формування вимог до функціональності системи доцільно здійснювати з урахуванням аналізу провідних рішень, присутніх на українському ринку, зокрема Uber, Bolt, Uklon та Opti.

Інтерфейс платформи Uber реалізований як швидкий механізм для введення місця посадки, призначення та вибору часу поїздки. З першого екрану користувач отримує доступ до основної функціональності, що дозволяє мінімізувати час взаємодії з системою (рис. 1.2) [20].

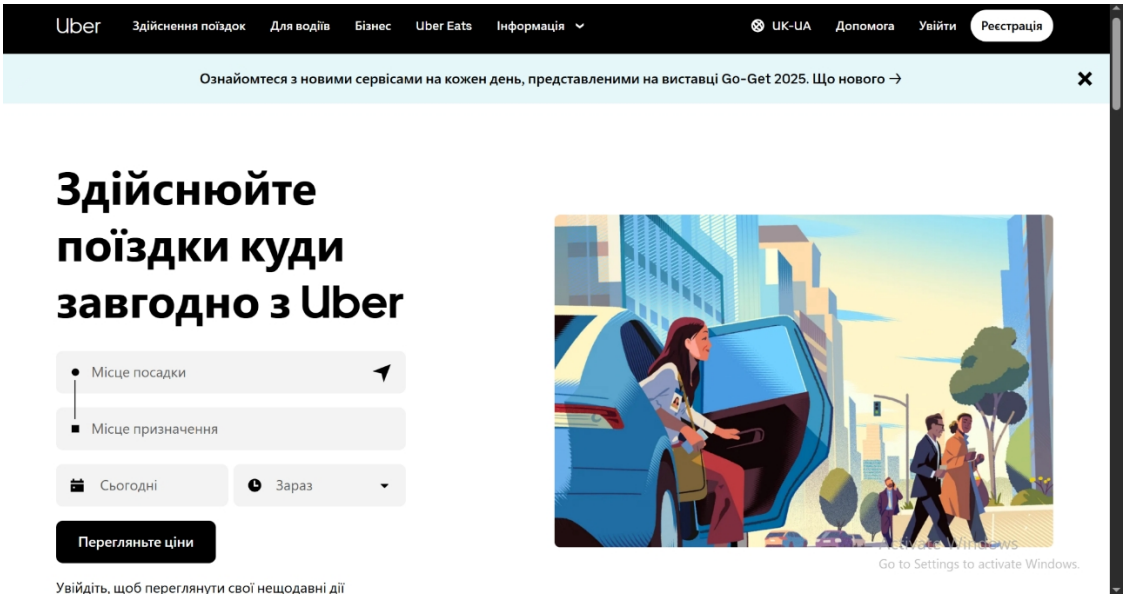


Рисунок 1.2 – Інтерфейс головної сторінки «Uber»

Підхід орієнтований на негайне створення замовлення та побудову маршруту в реальному часі. Такий принцип необхідно враховувати при формуванні вимог до клієнтського модуля проєктованої системи.

Сервіс Bolt, аналогічно, реалізує модель запуску поїздки без зайвих кроків, надаючи мінімалістичний інтерфейс з підтримкою мобільної адаптації. Початковий екран орієнтований на моментальну геолокацію та вибір напряму руху (рис. 1.3) [21].

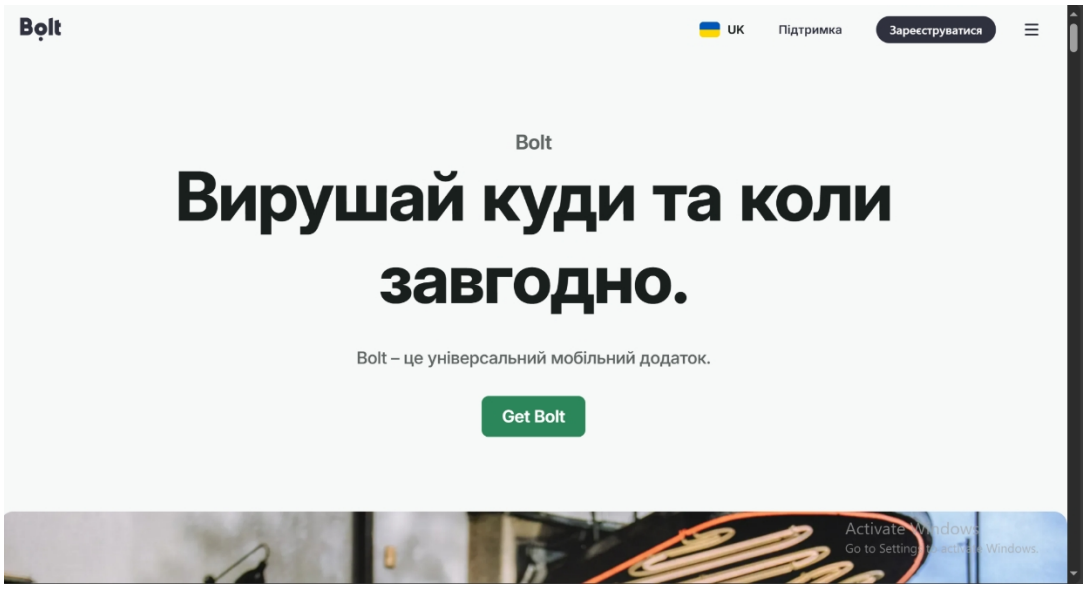


Рисунок 1.3 – Головнас сторінка сервісу «Bolt»

Для реалізації схожої зручності проєктована система має автоматично визначати координати клієнта та взаємодіяти з картографічним модулем у фоновому режимі.

Uklon демонструє іншу особливість — глибоку інтеграцію з локальними функціональностями: збереження декількох маршрутів, вибір адреси вручну або за допомогою карти, сплата поїздки через Google Pay / Apple Pay (рис. 1.4) [22].

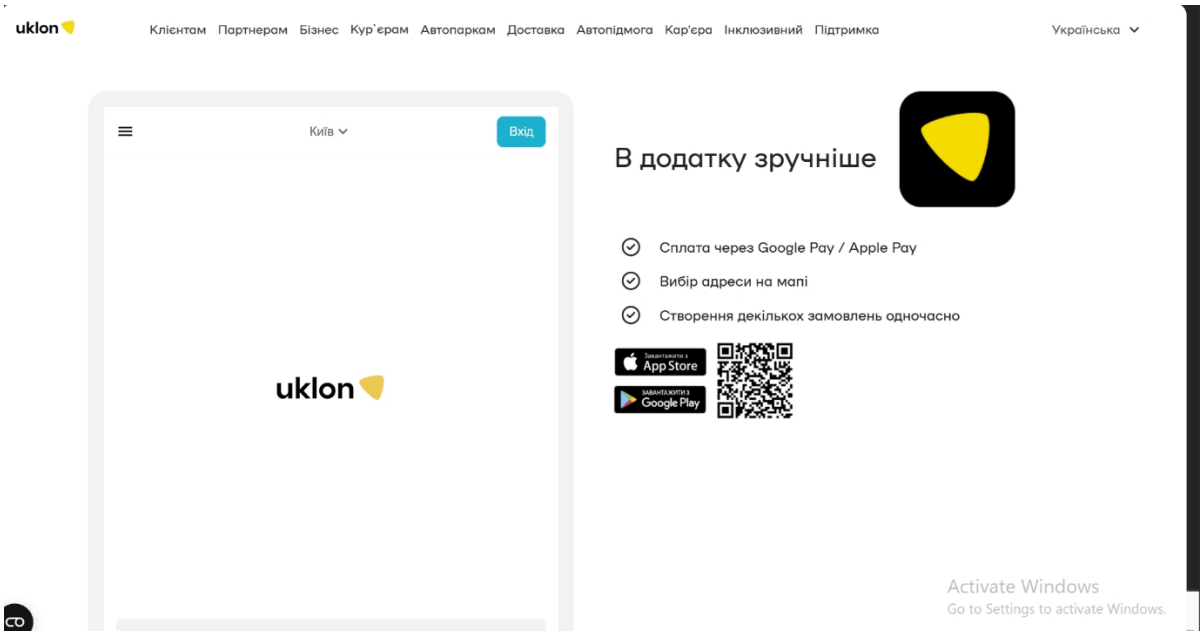


Рисунок 1.4 - Головна сторінка сервісу «Uklon»

На відміну від Uber і Bolt, які фокусуються на універсальності, Uklon акцентує на кастомізації сценарію замовлення, що є релевантним для українського ринку. З цієї причини функціональні вимоги до системи мають включати підтримку багаторазових шаблонів поїздок, швидке повторення замовлення та прив'язку платіжного методу.

Платформа Opti фокусується на емоційній складовій та простому UX-дизайні. Інтерфейс дозволяє миттєво перейти до замовлення через QR, застосунок або веб-форму, мінімізуючи етапи взаємодії (рис. 1.5) [23].

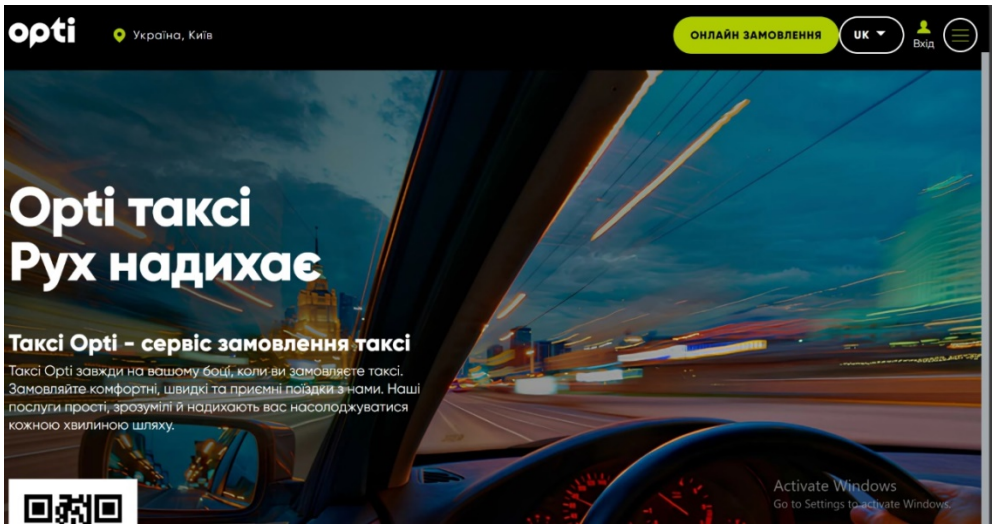


Рисунок 1.5 - Головна сторінка сервісу «Opti»

З точки зору функціональності, це визначає необхідність реалізації спрощеного сценарію створення замовлення з підтримкою мобільного додатку або окремої версії інтерфейсу для водіїв.

Усі розглянуті системи характеризуються високим ступенем оптимізації клієнтського досвіду, адаптивністю інтерфейсів, а також підтримкою автоматичного визначення місцеперебування. Проте жодна з них не є відкритою або адаптованою до інтеграції з внутрішніми системами малого або середнього бізнесу. Крім того, вони не передбачають автономного використання без постійного доступу до хмарної інфраструктури. Також у них відсутні механізми адаптації до конкретних локальних умов роботи (наприклад, невелика кількість водіїв, обмежена географія, офлайн-режими обліку тощо). Для обгрунтування попередньої інформації розглянемо таблицю 1.2, яка порівнює функціональність існуючих рішень і проєктованої

Таблиця 1.2 – Порівняння функціональності існуючих рішень і проєктованої системи

№	Характеристика	Uber	Bolt	Uklon	Opti	Проєктована система
1	Автоматичне визначення координат	+	+	+	+	+

Продовження таблиці 1.2

2	Веб-інтерфейс без мобільного застосунку	–	–	–	+	+
3	Підтримка локального розгортання	–	–	–	–	+
4	Повна автономія від	–	–	–	–	+

	зовнішніх API					
5	Платіжна інтеграція (Google Pay тощо)	+	+	+	–	– (опціонально)
6	Інтерфейс адміністратора	обмежений	обмежений	обмежений	частковий	повний контроль
7	Підтримка історії поїздок	+	+	+	+	+
8	Орієнтація на бізнес-користувача	–	–	–	–	+

Проєктована система не намагається дублювати комерційні сервіси, а пропонує вузькоспеціалізоване рішення для локального рівня, з мінімальними вимогами до інфраструктури, високим рівнем контролю над даними та зручним адмініструванням. Такий підхід дозволяє впровадити цифровізацію в компаніях, де традиційні сервіси є або надлишковими, або технічно непридатними для інтеграції.

1.3 Моделювання предметної області

Для формалізації логіки функціонування проєктованої системи використано засоби об’єктно-орієнтованого аналізу та нотації UML, які дозволяють структурувати функціональні сценарії взаємодії між користувачами та підсистемами. Моделювання предметної області здійснено шляхом побудови діаграм прецедентів і діаграм послідовностей для основних сценаріїв обробки замовлень.

На діаграмі прецедентів (рис. 1.6) відображено основні дії трьох типів користувачів: пасажира, водія та адміністратора.

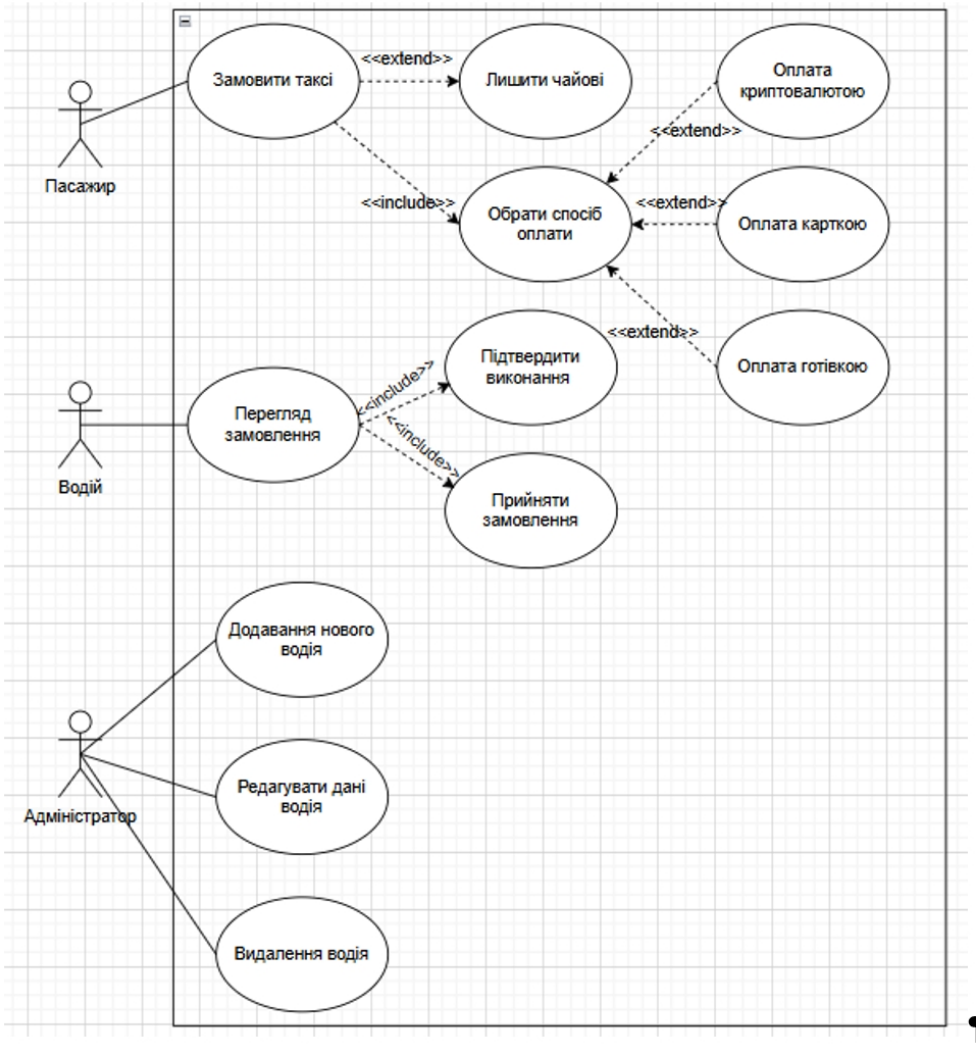


Рисунок 1.6 – Діаграма прецедентів розроблювальної системи

Пасажир може створити замовлення, обрати спосіб оплати, залишити чайові та підтвердити виконання поїздки. Сценарій «Замовити таксі» обов’язково включає дію «Обрати спосіб оплати», яка, в свою чергу, має варіанти реалізації: оплата готівкою, карткою або криптовалютою. Водій здійснює прийняття замовлення, його перегляд та підтвердження виконання. Адміністратор керує базою водіїв через операції додавання, редагування та

видалення записів. Модель прецедентів демонструє розмежування доступу відповідно до ролей та взаємозв’язки між функціональними сценаріями (рис. 1.6).

Для деталізації процесу створення та підтвердження замовлення побудовано діаграму послідовності (представлена на рисунку 1.7), яка відображає взаємодію між пасажиром, сервісом таксі, сервісом оплати та водієм.

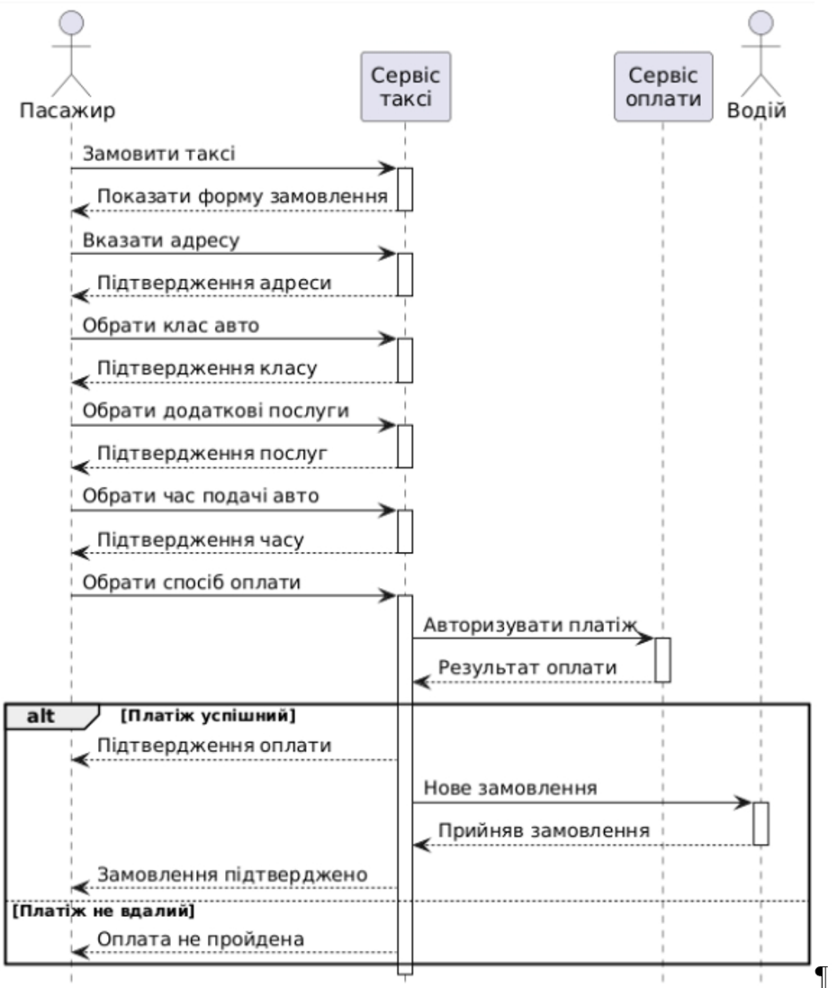


Рисунок 1.7 – Діаграма послідовності служби таксі

Сценарій починається з ініціації замовлення пасажиром, що викликає показ форми введення параметрів. Далі користувач вказує адресу, клас авто, додаткові послуги та бажаний час подачі. Після вибору способу оплати ініціюється запит до сервісу оплати, який повертає результат авторизації транзакції.

У випадку успішної оплати система підтверджує замовлення та надсилає його на розгляд водієві. Водій приймає замовлення, що завершує послідовність. У разі неуспішного платежу клієнт отримує відповідне повідомлення без переходу до етапу розсилки замовлення. Дана діаграма демонструє послідовність викликів функцій та залежності між підсистемами в межах одного запиту (рис. 1.7).

Деталізація логіки взаємодії між ролями системи, зокрема пасажиром, системним ядром і водієм, здійснена шляхом побудови UML-діаграми активності. Вона відображає

внутрішній механізм обробки замовлення, включаючи перевірку введених параметрів, обробку транзакцій, призначення виконавця та завершення маршруту.

Діаграма (рис. 1.8) складається з трьох функціональних доріжок (swimlanes), що відповідають ролям Passenger, System та Driver.

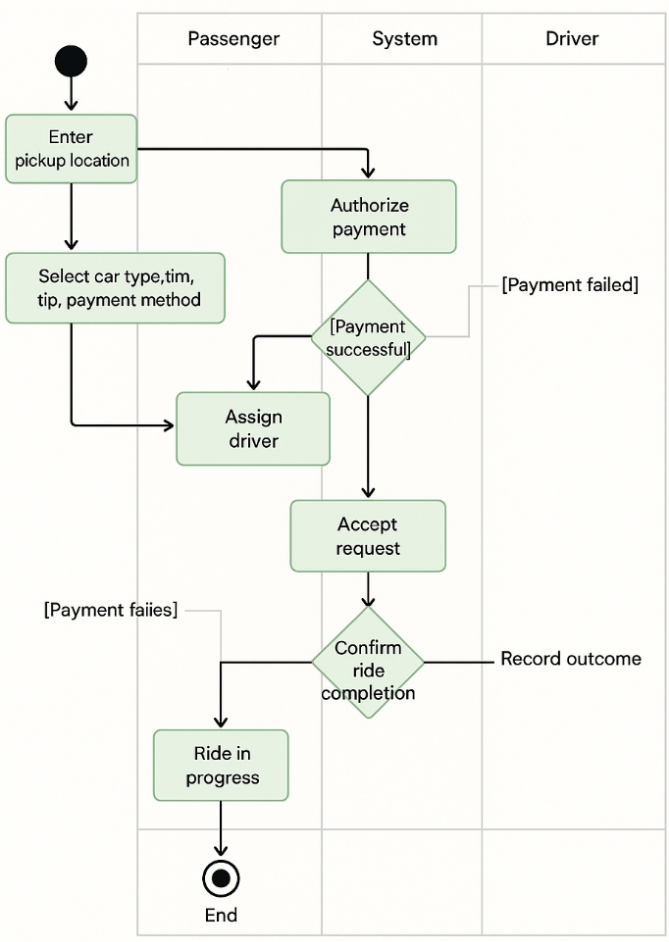


Рисунок 1.8 – Діаграма активності процесу створення та виконання замовлення

Сценарій розпочинається з дії Enter pickup location у ролі пасажира, яка ініціює процес створення замовлення. Далі користувач формує параметри поїздки через дію Select car type, time, tip, payment method, що містить конфігурацію класу автомобіля, додаткових сервісів, часу подачі та вибору платіжного інструменту.

На наступному етапі відбувається перехід керування до підсистеми System, яка виконує авторизацію оплати (Authorize payment). Результат цього етапу контролюється умовним вузлом [Payment successful]. У разі позитивного результату активність переходить до блоку Assign driver, що вказує на пошук вільного водія. Надалі система надсилає запит водію (Accept request), після чого верифікує завершення поїздки через блок Confirm ride completion.

Результат завершення маршруту фіксується дією Record outcome. Якщо на будь-якому етапі виникає відхилення (наприклад, [Payment failed]), процес переходить у гілку обробки

помилки та завершується. У випадку успішної обробки — контроль переходить до блоку Ride in progress і завершується через вузол End.

1.4 Постановка завдання і визначення вимог системи

Після виконаного аналізу предметної області, існуючих аналогів і побудови ключових моделей взаємодії в системі, на поточному етапі необхідно формалізувати вимоги, які визначають технічні, функціональні та поведінкові характеристики програмного забезпечення. Це дає змогу однозначно задати рамки майбутньої реалізації, уникнути суперечностей на рівні розробки, а також забезпечити відповідність очікуванням кінцевих користувачів системи.

Формування вимог здійснюється з позиції структурованого поділу на функціональні та нефункціональні. Функціональні вимоги описують конкретні дії, які система повинна виконувати в рамках своєї бізнес-логіки — це сценарії користувацької взаємодії, механізми прийому, обробки і завершення замовлення, робота з даними, доступ до інтерфейсу, авторизація і ролі. У свою чергу, нефункціональні вимоги задають обмеження або якісні характеристики системи, що стосуються продуктивності, масштабованості, платформи, безпеки, стабільності, часу реакції інтерфейсу та ін.

Вихідним базисом для розробки вимог слугують результати порівняльного аналізу аналогічних систем Uber, Bolt, Uklon та Opti (див. табл. 1.2), на підставі яких було визначено ключові операції, що забезпечують повний цикл обробки замовлення. При цьому враховано відмінності проєктованої системи — зокрема, орієнтацію на роботу у локальному середовищі, децентралізовану архітектуру без залежності від хмарних API, та необхідність забезпечення контролю з боку адміністратора.

У таблиці 1.3 подано перелік функціональних вимог, які мають бути реалізовані в рамках основного ядра програмного продукту. Кожна з них сформульована як завершений сценарій використання, що передбачає чітку вхідну умову, відповідальну роль (actor) і результат виконання.

Таблиця 1.3 – Функціональні вимоги до інформаційної системи

№	Вимога
1	Авторизація та реєстрація користувачів з розмежуванням ролей (пасажир, водій, адміністратор)
2	Створення замовлення із введенням координат, вибором типу авто, часу, опцій (чайові)
3	Інтеграція з геолокаційним модулем для фіксації місця подачі авто
4	Вибір способу оплати (готівка, банківська картка, криптовалюта)
5	Авторизація платежу з фіксацією результату (успішний/невдалий)
6	Призначення вільного водія за критеріями географічної близькості

7	Прийняття замовлення водієм, оновлення статусу виконання
8	Підтвердження завершення поїздки клієнтом або водієм
9	Ведення історії замовлень із доступом до детальної інформації (час, маршрут, сума)
10	Повний CRUD-функціонал для адміністратора щодо бази водіїв

Ці вимоги дозволяють реалізувати замкнений цикл функціонування служби таксі: від створення до завершення поїздки, включно з адмініструванням та контролем усіх ключових об'єктів предметної області.

У таблиці 1.4 наведено нефункціональні вимоги, які визначають контекст і характеристики експлуатації системи. Вони є критичними для забезпечення стабільності роботи, масштабованості та відповідності очікуваному рівню якості.

Таблиця 1.4 – Нефункціональні вимоги до інформаційної системи

№	Вимога
1	Підтримка операційної системи Windows 10 або вище
2	Інтерфейс користувача реалізується засобами WinForms
3	Зберігання даних у PostgreSQL з підтримкою ACID-транзакцій
4	Час відгуку GUI-компонентів не перевищує 1000 мс
5	Можливість автономної роботи без з’єднання з Інтернетом
6	Стійкість до збоїв: збереження сесій, відновлення після помилок
7	Підтримка не менше 100 одночасних з’єднань без деградації продуктивності

Продовження таблиці 1.4

8	Захист від некоректного вводу та обробка помилок на стороні клієнта і сервера
9	Повна локалізація інтерфейсу українською мовою
10	Логування усіх дій адміністратора і водія з мітками часу

Розроблена система має забезпечити реалізацію всіх ключових бізнес-сценаріїв замовлення й виконання поїздки в умовах обмеженого середовища експлуатації. Сформульовані вимоги є вичерпними щодо охоплення типових процесів та цілком достатніми для створення стабільного й масштабованого прикладного рішення. Вони також створюють основу для побудови архітектури, деталізації інтерфейсів та подальшого формального тестування. Визначення вимог завершує етап концептуального моделювання системи й забезпечує перехід до її структурного проєктування.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1Логічна модель даних

Проєктування бази даних є критичним етапом при створенні інформаційної системи управління перевезенням пасажирів таксі, оскільки забезпечує збереження, цілісність і зв’язність усіх ключових об'єктів предметної області. Для формалізації структури даних була розроблена логічна модель у вигляді сутнісно-зв’язної діаграми (ER-моделі), що відображає основні сутності, їхні атрибути та взаємозв’язки [9].

На рисунку 2.1 наведено логічну модель даних, що реалізує всі аспекти діяльності системи — від обробки замовлень до зберігання інформації про користувачів, водіїв, транспортні засоби, оцінювання сервісу та додаткові послуги.

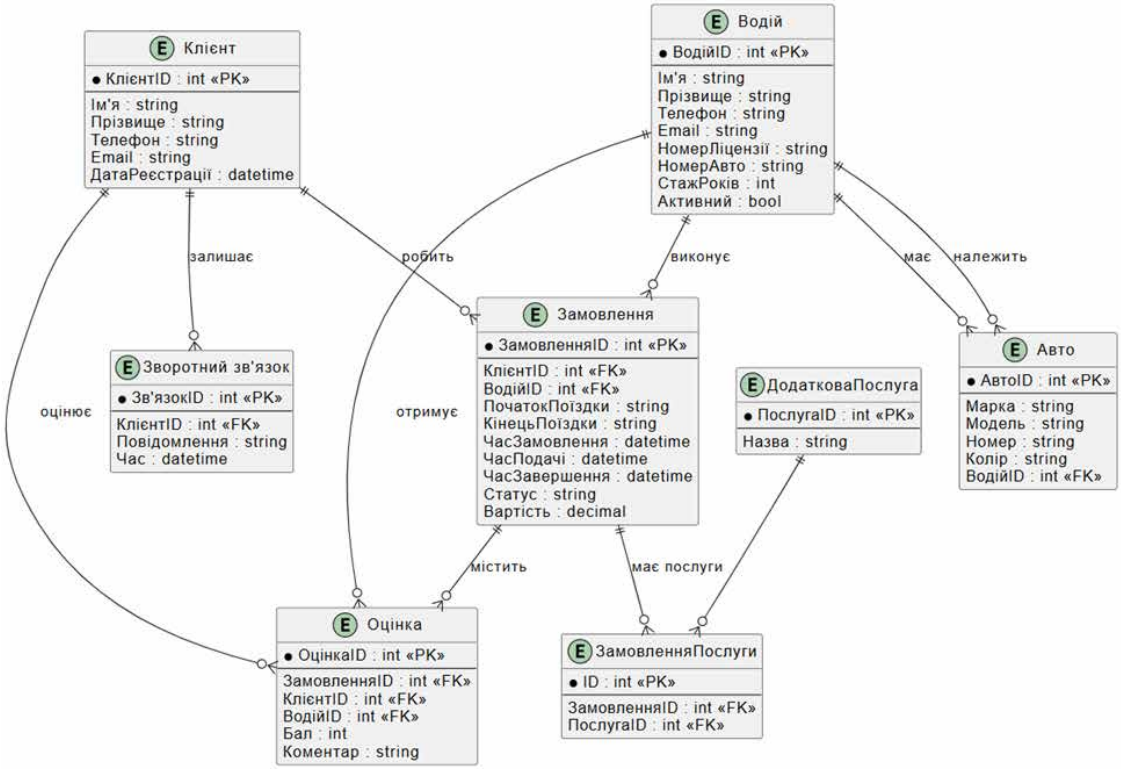


Рисунок 2.1 – Логічна модель даних інформаційної системи служби таксі

У логічній моделі виділено дев’ять ключових сутностей: Клієнт, Водій, Замовлення, Авто, Оцінка, Зворотний зв’язок, ДодатковаПослуга, ЗамовленняПослуги, які пов’язані між собою через відношення типу один-до-багатьох або багато-до-багатьох.

У таблиці 2.1 подано перелік сутностей логічної моделі з коротким описом їхніх атрибутів.

Таблиця 2.1 – Опис основних сутностей логічної моделі

№	Сутність	Основні атрибути	Призначення
1	Клієнт	КлієнтID, Ім’я, Прізвище, Телефон, Email, ДатаРеєстрації	Зберігає інформацію про користувачів системи
2	Водій	ВодійID, Ім’я, Прізвище, Email, НомерЛіцензії, НомерАвто, Стаж, Активний	Містить реєстраційні дані водіїв
3	Авто	АвтоID, Марка, Модель, Номер, Колір, ВодійID	Відображає транспортні засоби, прив’язані до водіїв
4	Замовлення	ЗамовленняID, КлієнтID, ВодійID, ПочатокПоїздки, КінецьПоїздки, ЧасПодачі, Статус	Основна таблиця, яка містить всі параметри здійснених поїздок
5	Оцінка	ОцінкаID, ЗамовленняID, КлієнтID, ВодійID, Бали, Коментар	Містить оцінки та коментарі після поїздки
6	Зворотний зв’язок	Зв’язокID, КлієнтID, Повідомлення, Час	Зберігає додаткові повідомлення клієнтів до адміністрації
7	ДодатковаПослуга	ПослугаID, Назва	Довідник доступних додаткових послуг
8	ЗамовленняПослуги	ID, ЗамовленняID, ПослугаID	Реалізує зв’язок типу багато-до-багатьох між замовленнями і послугами

Ключова сутність Замовлення є центральною у логіці системи. Вона поєднує Клієнта, Водія, масив Послуг, фіксує часові параметри поїздки, статус виконання та вартість. Зв’язки один-до-багатьох (Клієнт → Замовлення, Водій → Замовлення) дають змогу зберігати історію поїздок кожного учасника окремо.

Сутність Оцінка реалізує двосторонній аналіз якості: вона зв’язана з Клієнтом, Водієм та конкретним Замовленням. Такий підхід забезпечує можливість формування статистичних показників якості обслуговування. Сутність Зворотний зв’язок додатково дозволяє клієнтам надавати уточнення або скарги адміністрації, які фіксуються із міткою часу.

Особливу увагу приділено сутності ДодатковаПослуга — вона організована як довідник, а зв’язок із Замовленням реалізовано через проміжну таблицю ЗамовленняПослуги. Це дозволяє гнучко оперувати додатковими функціями, такими як «поїздка з кондиціонером», «перевезення тварин» тощо.

З метою нормалізації структури даних у моделі реалізовано:

- унікальні первинні ключі для всіх таблиць (ID),
- зовнішні ключі з обмеженням цілісності (FOREIGN KEY),
- використання відповідних типів даних (int, string, datetime, decimal),
- розділення довідників (Послуги, Авто) від основних транзакційних сутностей (Замовлення, Оцінка).

Побудована логічна модель повністю охоплює всі аспекти предметної області, забезпечує гнучкість розширення, ефективність запитів до бази даних і підтримку основних

функціональних вимог, сформульованих у розділі 1. У наступному підпункті буде розглянуто фізичне проєктування та реалізацію цієї структури засобами реляційної СУБД..

2.2Діаграма класів розроблювальної системи

Діаграма класів є одним із ключових структурних артефактів об’єктно-орієнтованого моделювання, що дозволяє формалізувати статичну структуру системи та відобразити взаємозв’язки між її об’єктами. Вона слугує основою для реалізації програмного коду, визначаючи властивості (атрибути) та поведінку (методи) кожного класу, а також зв’язки типу наслідування, асоціації та композиції.

На рисунку 2.2 представлено діаграму класів програмної системи управління перевезенням пасажирів у службі таксі. Вона побудована згідно з принципами UML та враховує основні логічні сутності, які беруть участь у бізнес-процесах замовлення, виконання та адміністрування поїздок.

Рисунок 2.2 – Діаграма класів програмної системи служби таксі

Ключовим елементом моделі є абстрактний клас Користувач, який інкапсулює загальні атрибути: id, ім’я, телефон, а також базові операції зареєструватися() і увійти(). Від нього успадковуються три спеціалізовані класи: Пасажир, Водій і Адміністратор, що реалізують власну поведінку відповідно до функціональних ролей у системі.

Клас Пасажир має атрибут історіяЗамовлень, який реалізується як список об’єктів типу Замовлення, що забезпечує доступ до архіву поїздок. Методи оформитиЗамовлення() та скасуватиЗамовлення() реалізують взаємодію з клієнтським інтерфейсом. В той час як клас Водій включає атрибут авто, що є екземпляром класу Автомобіль, а також методи

прийнятиЗамовлення() і завершитиЗамовлення(), які активуються у відповідь на дії клієнта або адміністратора.

І нарешті, клас Адміністратор реалізує адміністрування системи через методи: додатиВодія(), редагуватиДані() та видалитиВодія(), що забезпечують повний життєвий цикл керування обліковими записами водіїв.

Центральну роль у системі відіграє клас Замовлення, який містить атрибути:

– id – унікальний ідентифікатор;
– датаЧас – момент оформлення замовлення;
– статус – поточний стан поїздки;
– маршрут – зв'язок з класом Маршрут;
– спосібОплати – зв'язок із класом Оплата;
– послуги – список додаткових сервісів.

Клас Маршрут моделює логістику поїздки та містить інформацію про адресу початку, адресу призначення та відстань. Клас Оплата містить тип (готівка, карта), суму та метод здійснити() для ініціації транзакції.

І останньою є клас Послуга представляє додаткову функціональність, що може бути включена до поїздки (наприклад, «дитяче крісло», «перевезення тварин») та включає назву і ціну.

У таблиці 2.2 подано короткий опис кожного з класів та його функціонального призначення.

№	Назва класу	Атрибути та методи	Призначення
1	Користувач	id, ім'я, телефон; зареєструватися(), увійти()	Абстрактна сутність користувача, спільна для всіх типів ролей
2	Пасажир	історіяЗамовлень; оформитиЗамовлення(), скасуватиЗамовлення()	Реалізує клієнтський функціонал замовлення поїздки
3	Водій	статус, авто; прийнятиЗамовлення(), завершитиЗамовлення()	Відповідає за виконання замовлення
4	Адміністратор	–; додатиВодія(), редагуватиДані(), видалитиВодія()	Адмініструє базу водіїв та контроль системи
5	Замовлення	id, датаЧас, статус, маршрут, спосібОплати, послуги	Центральний клас, що фіксує всі параметри поїздки
6	Маршрут	адресаПочатку, адресаПризначення, відстань	Визначає просторові параметри поїздки
7	Оплата	тип, сума; здійснити()	Реалізує платіжну транзакцію
8	Послуга	назва, ціна	Моделює додаткові сервіси, що включаються до замовлення
9	Автомобіль	марка, модель, номер, клас	Описує транспортний засіб, прив'язаний до конкретного водія

Побудована модель демонструє дотримання принципів інкапсуляції, наслідування та композиції. Вона є структурною основою для реалізації класів у C# з подальшою інтеграцією

до інтерфейсних і базових модулів системи. Діаграма класів забезпечує основу для реалізації логіки взаємодії між модулями, тестування поведінки об’єктів та уніфікованого управління даними під час життєвого циклу програмного забезпечення.

2.4Діаграма пакетів

У межах структурного проєктування програмного забезпечення для системи управління перевезенням пасажирів критичним етапом є побудова діаграми пакетів, яка дозволяє візуалізувати розподіл функціональності системи на логічні модулі. Дана діаграма є високорівневим представленням архітектури, що демонструє взаємозв’язки між основними компонентами системи, організованими за функціональним принципом. Вона виконує роль інтеграційної карти, яка формалізує поділ відповідальностей, визначає точки зв’язку між підсистемами та закладає основу для модульного тестування і розгортання. На рисунку 2.3 представлено діаграму пакетів розроблюваної системи, яка охоплює шість основних модулів: «Користувачі», «Замовлення», «Оплата», «Керування персоналом», «Повідомлення» та «Інтеграція з платіжною системою».

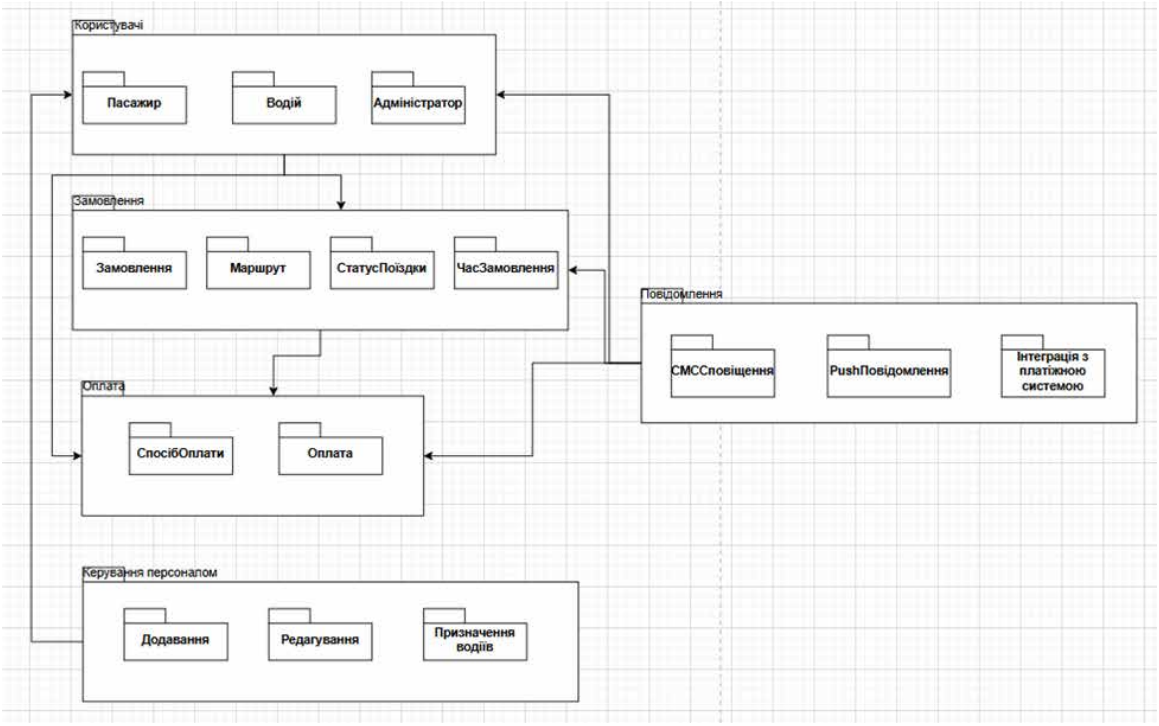


Рисунок 2.3 – Діаграма пакетів розроблювальної системи

Ці пакети відповідають логічним підсистемам, що реалізують окремі аспекти функціональності.

У структурі системи центральне місце посідає пакет «Замовлення», оскільки він забезпечує обробку основного бізнес-процесу – оформлення, супроводу та завершення

поїздки. Саме цей модуль пов’язує між собою інші компоненти через залежності до користувачів, системи оплати, засобів комунікації та адміністративного управління.

Взаємодія з користувачами реалізується через пакет «Користувачі», який включає три спеціалізовані ролі – пасажир, водій та адміністратор. Кожен із них має доступ до власного підмножини функціональності в рамках системи, відповідно до прав доступу. Пасажир ініціює замовлення, водій його виконує, а адміністратор здійснює контроль, модерацію та розподіл ресурсів. Пакет «Керування персоналом» реалізує адміністрування облікових записів водіїв, їх призначення до активних замовлень, а також забезпечує оновлення їхніх даних через функції додавання та редагування.

Пакет «Оплата» інтегрується до процесу замовлення для забезпечення обробки транзакцій. Він містить логіку вибору способу оплати (готівка, картка, криптовалюта), ініціацію платежу та фіксацію його результатів. Важливо, що цей модуль не є ізольованим: він має зовнішню залежність до пакета «Інтеграція», що забезпечує зв’язок з платіжними шлюзами та сторонніми API. Система сповіщень представлена пакетом «Повідомлення», до якого включено як SMS-, так і push-канали інформування, що активуються на подієвих тригерах – підтвердження оплати, зміна статусу замовлення, прибуття авто тощо. Цей механізм забезпечує двосторонню синхронізацію між клієнтом і системою в реальному часі.

Для систематизації архітектурної моделі доцільно узагальнити складові компонентів у табличній формі. У таблиці 2.3 наведено опис кожного пакета, його ключових класів та взаємозв’язків.

Таблиця 2.3 – Основні пакети та компоненти діаграми

№	Назва пакета	Основні компоненти	Взаємозв’язки
1	Користувачі	Пасажир, Водій, Адміністратор	Пов’язаний із Замовленням, Керуванням персоналом
2	Замовлення	Замовлення, Маршрут, СтатусПоїздки, Час	Центр зв’язку з Користувачами, Оплатою, Повідомленнями
3	Оплата	СпособиОплати, Оплата	Взаємодіє із Замовленням, залежить від Інтеграції
4	Керування персоналом	Додавання, Редагування, Призначення водіїв	Залежить від Користувачів
5	Повідомлення	СМСПовідомлення, PushПовідомлення	Отримує події з Замовлення
6	Інтеграція	Платіжна система	Взаємодіє з Оплатою

Згідно з рисунком 2.3, структура системи демонструє чітке розмежування функціональних зон та відповідає вимогам інкапсуляції і слабкого зв’язування, що забезпечує адаптивність до змін і спрощує масштабування. Використання пакетної архітектури дозволяє уникнути надмірної залежності між модулями, забезпечує модульність реалізації та відкриває можливості для інтеграції з іншими інформаційними системами,

зокрема через API рівня повідомлень або платіжних сервісів. Такий підхід є доцільним у розробці корпоративних інформаційних систем з чітким розподілом відповідальностей і розширюваною логікою бізнес-процесів.

2.5Діаграма компонентів

Діаграма компонентів визначає фізичну архітектуру системи, відображаючи взаємозв'язки між основними модулями, їхні інтерфейси та реалізації. На рисунку 2.4 представлено структуру інформаційної системи управління перевезенням пасажирів, яка поділена на два компоненти: клієнтський застосунок TaxiOrderingApp та бібліотеку бізнес-логіки Service.dll.

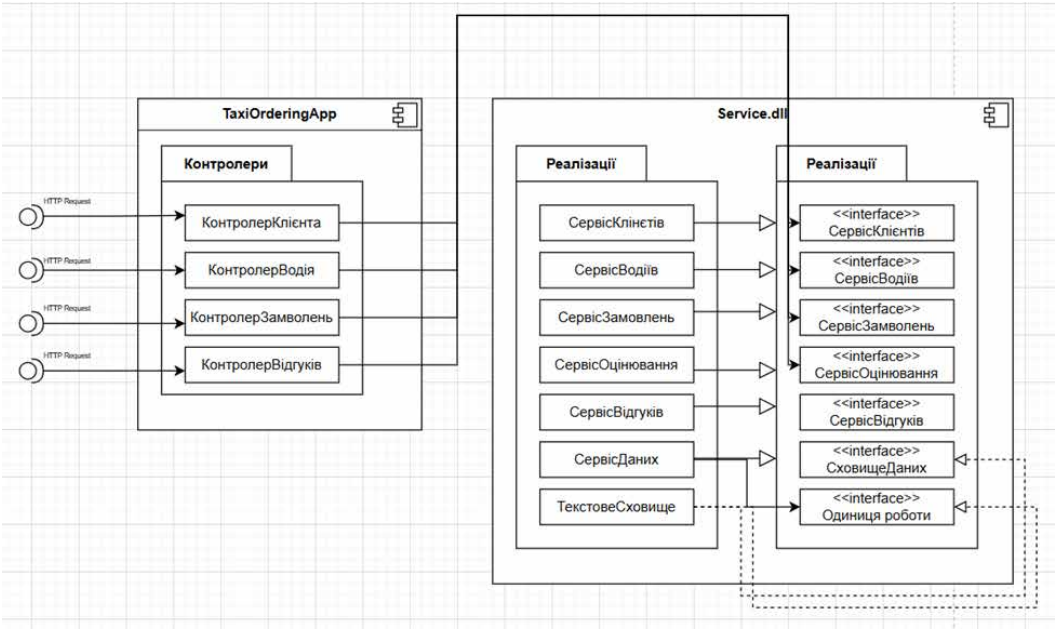


Рисунок 2.4 – Діаграма компонентів програмної системи

Клієнтська частина містить контролери, які обробляють HTTP-запити й спрямовують їх до відповідних сервісів. КонтролерКлієнта, КонтролерВодія, КонтролерЗамовлень та КонтролерВідгуків реалізують обробку запитів відповідно до ролей користувачів. Вони ізольовані від логіки обробки, яку реалізує Service.dll.

Бібліотека Service.dll структурована у вигляді набору інтерфейсів та їх реалізацій. Кожен сервіс — СервісКлієнтів, СервісВодіїв, СервісЗамовлень, СервісОцінювання, СервісВідгуків — реалізує окремий функціональний блок системи. Сховище реалізовано у вигляді компонента ТекстовеСховище, що відповідає інтерфейсу СховищеДаних, і забезпечує гнучкість при зміні джерела зберігання. Компонент Одиниця роботи підтримує цілісність транзакцій між модулями.

Реалізована архітектура забезпечує чітке розмежування рівнів контролю, логіки та зберігання, відповідаючи принципам інверсії залежностей та модульності. Такий підхід спрощує тестування, підтримку та масштабування системи.

Для кращого розуміння призначення кожного з компонентів діаграми наведемо таблицю 2.4, яка структурує їхню функціональну роль та взаємозв’язки.

Таблиця 2.4 – Основні компоненти та їх призначення

№	Назва компонента	Призначення
1	КонтролерКлієнта	Обробка запитів клієнтів, передача даних у СервісКлієнтів
2	КонтролерВодія	Взаємодія з даними про водіїв, перенаправлення запитів до СервісВодіїв
3	КонтролерЗамовлень	Створення, оновлення, завершення замовлень через СервісЗамовлень
4	КонтролерВідгуків	Збір і обробка відгуків користувачів через СервісВідгуків
5	Сервіси (Клієнтів тощо)	Бізнес-логіка обробки сутностей, реалізація контрактів інтерфейсів
6	ТекстовеСховище	Реалізація локального зберігання даних відповідно до інтерфейсу Сховища
7	Інтерфейси	Анотація функціональних контрактів і забезпечення інверсії залежностей

Побудована діаграма компонентів формалізує фізичну структуру системи, демонструє її модульну організацію та сприяє реалізації принципів масштабованої і підтримуваної архітектури для прикладного програмного забезпечення у сфері перевезень.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління базою даних

Для зберігання структурованих даних у розроблюваній інформаційній системі використовується реляційна система керування базами даних Microsoft SQL Server. Цей вибір обґрунтовано високою швидкодією при виконанні транзакцій, підтримкою ACID-властивостей, розширеною системою безпеки, широкими можливостями для оптимізації запитів і зручним інструментарієм адміністрування. SQL Server дозволяє реалізувати складні реляційні структури, зовнішні ключі, обмеження цілісності, тригери та індексацію для підвищення продуктивності.

Фізична реалізація бази даних базується на логічній моделі, поданій у розділі 2. Основними сутностями є: Clients, Drivers, Orders, Cars, Payments, Ratings, Feedback, Services, OrderServices. Для забезпечення коректності роботи всі таблиці мають первинні ключі, визначені типи даних, а також зовнішні ключі для зв’язку з іншими таблицями.

Нижче наведено приклад створення таблиці Clients, яка зберігає дані про користувачів системи на рисунку 3.1.

```
CREATE TABLE Clients (
    ClientID INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Phone NVARCHAR(20),
    Email NVARCHAR(100),
    RegistrationDate DATETIME DEFAULT GETDATE()
);
```

Рисунок 3.1 – Лістинінг коду створення таблиці Clients

Кожен клієнт автоматично отримує унікальний ідентифікатор, а дата реєстрації встановлюється системною функцією. Зв’язок між замовленням і клієнтом реалізується через зовнішній ключ, як показано на рисунку 3.2.

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY IDENTITY(1,1),
    ClientID INT FOREIGN KEY REFERENCES Clients(ClientID),
    DriverID INT FOREIGN KEY REFERENCES Drivers(DriverID),
    StartTime DATETIME,
    EndTime DATETIME,
    OrderStatus NVARCHAR(20),
    TotalAmount DECIMAL(10,2)
);
```

Рисунок 3.2 – Лістинінг запиту представлення зв'язку між замовленням і клієнтом і відповідне створення таблиці

У структурі таблиці Orders зберігаються часові параметри, поточний статус і фінансова інформація. Для підтримки гнучкості й масштабованості впроваджено таблицю зв'язку OrderServices для реалізації відношення багато-до-багатьох між замовленнями та додатковими сервісами, що продемонстровано на рис. 3.3.

```
CREATE TABLE OrderServices (
    ID INT PRIMARY KEY IDENTITY(1,1),
    OrderID INT FOREIGN KEY REFERENCES Orders(OrderID),
    ServiceID INT FOREIGN KEY REFERENCES Services(ServiceID)
);
```

Рисунок 3.3 – Лістинінг запиту створення замовлення-додаткові сервіси

Для забезпечення аналітичного доступу до інформації використовуються SQL-запити із застосуванням агрегатних функцій. Наприклад, для отримання середньої оцінки для кожного водія формується такий запит, як показано на рис.3.4.

```
SELECT DriverID, AVG(RatingValue) AS AverageRating
FROM Ratings
GROUP BY DriverID;
```

Рисунок 3.4 – Запит отримання середньої оцінки для кожного водія

Цей запит дозволяє формувати рейтинги водіїв на основі відгуків клієнтів. Також реалізовано запит для отримання списку замовлень певного клієнта, що продемонстровано на рис.3.5.

```
SELECT o.OrderID, o.StartTime, o.EndTime, o.TotalAmount
FROM Orders o
JOIN Clients c ON o.ClientID = c.ClientID
WHERE c.Email = 'user@example.com';
```

Рисунок 3.5 – Запит отримання списку замовлень

З метою оптимізації пошуку і підвищення швидкодії застосовано індекси для полів Phone, Email у таблиці Clients, а також для OrderStatus у таблиці Orders.

У таблиці 3.1 подано загальний перелік основних таблиць, їх призначення та ключові поля.

Таблиця 3.1 – Основні таблиці бази даних та їх призначення

№	Назва таблиці	Призначення	Ключові поля
1	Clients	Дані про користувачів	ClientID, Email, Phone
2	Drivers	Інформація про водіїв	DriverID, LicenseNumber
3	Orders	Замовлення поїздок	OrderID, ClientID, DriverID
4	Payments	Дані про оплати	PaymentID, OrderID, Amount
5	Ratings	Оцінки клієнтів	RatingID, DriverID, RatingValue
6	Feedback	Повідомлення та скарги	FeedbackID, ClientID, Message
7	Services	Список додаткових послуг	ServiceID, Name
8	OrderServices	Зв’язок замовлень і послуг	OrderID, ServiceID

Реалізація бази даних у середовищі SQL Server забезпечує повну підтримку операцій CRUD, цілісність транзакцій, контроль доступу та гнучке масштабування відповідно до обсягів даних і кількості користувачів системи. Схема дозволяє здійснювати глибокий аналіз, формувати звітність та інтегруватися з зовнішніми системами через стандартні SQL-запити та вбудовані засоби реплікації та резервного копіювання.

3.2 Розробка інформаційної бази

Фізична модель бази даних є формалізованим відображенням логічної структури предметної області у вигляді реляційної схеми, реалізованої в середовищі Microsoft SQL Server. Вона включає набір взаємопов’язаних таблиць, кожна з яких відповідає окремій сутності предметної області. Таблиці мають чітко визначену структуру: назви полів, типи даних, обмеження NULL/NOT NULL, первинні та зовнішні ключі, а також унікальні індекси та значення за замовчуванням. Основна мета фізичної моделі — забезпечити ефективне, надійне й цілісне зберігання даних із підтримкою транзакцій і можливістю масштабування.

На рисунку 3.3 представлено повну ілюстрацію фізичної моделі бази даних системи таксомоторного обслуговування. Діаграма включає таблиці Clients, Drivers, Orders, Payments, Ratings, Feedback, Services та OrderServices, між якими реалізовано зв’язки типу «один-до-багатьох» та «багато-до-багатьох» через проміжну таблицю. Усі зв’язки базуються на зовнішніх ключах, що забезпечує референційну цілісність.

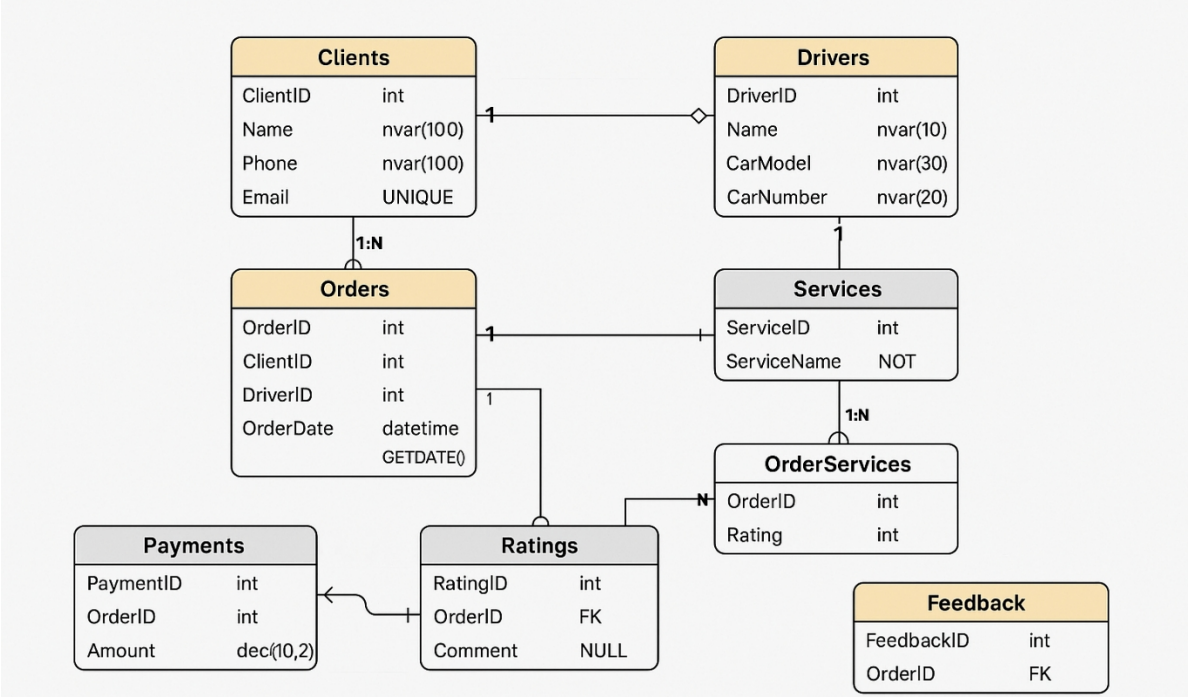


Рисунок 3.3 – Фізична модель бази даних системи таксомоторного обслуговування

У таблиці Clients зберігаються персональні дані користувачів, кожен з яких ідентифікується через унікальний ClientID. Поле Email має обмеження UNIQUE, що виключає дублювання. Аналогічно, у таблиці Drivers фіксуються дані про водіїв, включно з моделлю автомобіля, номером реєстрації, статусом активності (IsActive) та датою реєстрації, яка автоматично встановлюється за допомогою GETDATE().

Центральним елементом системи є таблиця Orders, яка зв’язує клієнтів із водіями, фіксує часові межі поїздки, статус та загальну суму. До неї прив’язуються записи оплат (Payments), оцінок (Ratings) та послуг (OrderServices). Додаткові сервіси, такі як кондиціонер або дитяче крісло, представлені окремою таблицею Services, яка зв’язується з замовленнями через проміжну таблицю.

Для демонстрації запитів, що використовуються на рівні вибірки, нижче наведено приклади SQL-інструкцій на рис.3.4, реалізованих у системі

```sql
-- Вибірка середньої оцінки для кожного водія
SELECT DriverID, AVG(RatingValue) AS AverageRating
FROM Ratings
GROUP BY DriverID;

-- Вибірка замовлень певного клієнта за email
SELECT o.OrderID, o.StartTime, o.EndTime, o.TotalAmount
FROM Orders o
JOIN Clients c ON o.ClientID = c.ClientID
WHERE c.Email = 'user@example.com';
```

Рисунок 3.4 – Реалізація запитів вибірки

Усі таблиці створено з дотриманням нормалізації: повторювані дані винесено в окремі довідники, забезпечено унікальність ключових атрибутів, що підвищує ефективність виконання запитів і зменшує надмірність.

У таблиці 3.4 наведено опис основних таблиць фізичної моделі бази даних, з переліком ключових полів та типів зв’язків.

Таблиця 3.4 – Опис таблиць фізичної моделі бази даних

№	Назва таблиці	Основні поля	Тип зв’язку
1	Clients	ClientID, Name, Phone, Email	1:N → Orders, Feedback
2	Drivers	DriverID, Name, CarModel, CarNumber	1:N → Orders, Ratings
3	Orders	OrderID, ClientID, DriverID, OrderDate	центральна, зв’язки з усіма
4	Payments	PaymentID, OrderID, Amount	1:1 → Orders
5	Ratings	RatingID, OrderID, RatingValue	1:N → Drivers, Orders
6	Feedback	FeedbackID, OrderID, Message	1:N → Clients
7	Services	ServiceID, ServiceName	1:N → OrderServices
8	OrderServices	OrderID, ServiceID	M:N → Orders × Services

Запропонована фізична модель є готовою до експлуатації в продуктивному середовищі, підтримує масштабування, розширення новими модулями та інтеграцію з клієнтськими додатками. Реалізація на базі SQL Server забезпечує сумісність з сучасними інструментами адміністрування, можливість резервного копіювання, логування змін і створення аналітичних запитів для формування звітності.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення відіграє ключову роль у забезпеченні структурної організації системи, її масштабованості, підтримуваності та надійності. Для

інформаційної системи управління перевезенням пасажирів побудовано трирівневу архітектуру, що реалізує розділення логіки на презентаційний шар (Presentation Layer), шар бізнес-логіки (Business Logic Layer) та шар доступу до даних (Data Access Layer). Такий підхід відповідає сучасним принципам побудови прикладного ПЗ і дозволяє гнучко розвивати систему без порушення цілісності її внутрішніх компонентів.

Архітектура побудована з використанням інверсії залежностей: зовнішній шар взаємодіє з внутрішнім лише через контракти (інтерфейси), що забезпечує слабке зв'язування та високу модульність. Основу реалізації становить клієнтський застосунок TaxiOrderingApp, який приймає вхідні запити користувачів через графічний інтерфейс або HTTP, після чого спрямовує їх до контролерів, що знаходяться у презентаційному шарі. Контролери делегують обробку до відповідних сервісів бізнес-логіки (СервісКлієнтів, СервісЗамовлень, тощо), які реалізують прикладну поведінку відповідно до вимог системи.

На рисунку 3.4 представлено повну ілюстрацію компонентної архітектури системи. Компоненти згруповані за логічними рівнями та взаємодіють через визначені інтерфейси, що забезпечує ізольованість змін і спрощення модульного тестування.

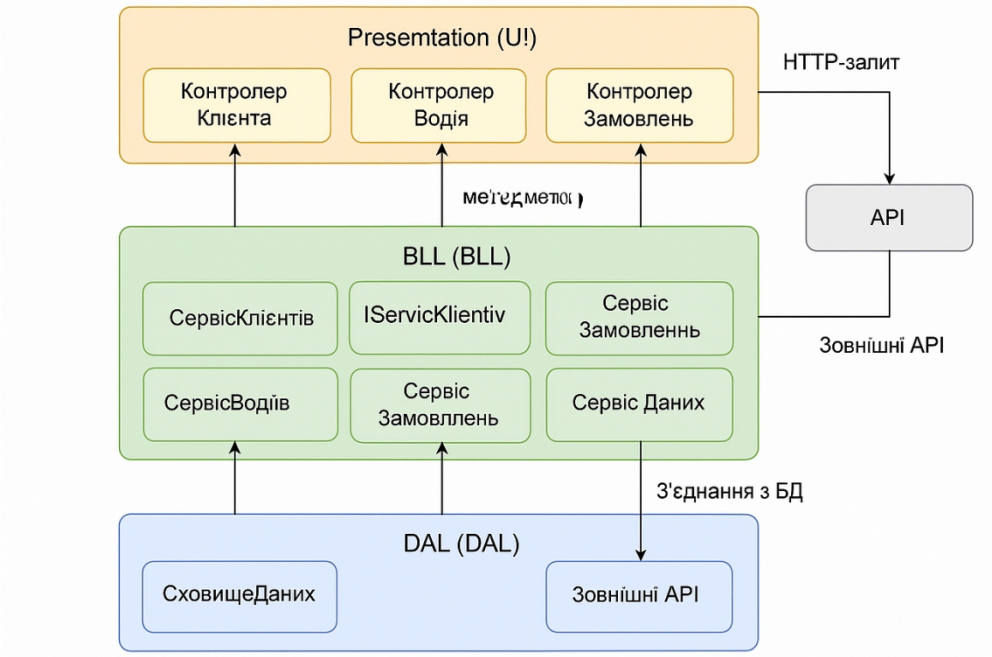


Рисунок 3.4 – Архітектура програмного забезпечення системи управління перевезенням пасажирів

У презентаційному шарі реалізовано чотири основні контролери: КонтролерКлієнта, КонтролерВодія, КонтролерЗамовлень, КонтролерВідгуків. Кожен контролер обробляє запити від відповідної ролі користувача, ініціює перевірку даних та взаємодіє із сервісами бізнес-рівня через чітко визначені інтерфейси (IServicКлієнтів, IServicВодіїв тощо). Шар бізнес-логіки містить реалізації сервісів, які інкапсулюють усі обчислення, перевірки, призначення ресурсів та взаємодію із зовнішніми компонентами. Усі сервіси залежні від

інтерфейсів сховищ (ISqlRepository, ITextStorage), що дозволяє легко змінювати реалізацію доступу до даних, зберігаючи логіку незмінною.

Шар доступу до даних відповідає за роботу з базою даних SQL Server. У стандартній конфігурації реалізовано сховище ТекстовеСховище, яке може бути замінене на SQL-реалізацію без зміни логіки роботи системи. Через патерн Одиниця роботи (Unit of Work) реалізується атомарність операцій та контроль транзакцій. Це забезпечує послідовність змін у системі при одночасному доступі з декількох джерел.

Для узагальнення архітектурної структури системи у таблиці 3.5 наведено опис трьох логічних рівнів архітектури, їх функцій та відповідних компонентів.

Таблиця 3.5 – Опис логічних рівнів архітектури ПЗ

Шар	Функціональне призначення	Основні компоненти
Презентаційний (UI)	Отримання запитів користувача, валідація, делегування до сервісів	КонтролерКлієнта, КонтролерВодія, КонтролерЗамовлень, КонтролерВідгуків
Бізнес-логіка (BLL)	Обробка логіки системи, розподіл навантаження, перевірка прав доступу	СервісКлієнтів, СервісЗамовлень, СервісОцінювання, СервісДаних
Доступ до даних (DAL)	Взаємодія з БД, виконання CRUD-операцій, реалізація транзакцій	ТекстовеСховище, СховищеSQL, Одиниця роботи, інтерфейси сховищ

Реалізована архітектура повністю відповідає принципам чистої архітектури (Clean Architecture) та DDD-підходу (Domain-Driven Design), що забезпечує ізоляцію бізнес-логіки від технологічних залежностей. Вона підтримує розширення (наприклад, додавання REST API, мобільного застосунку), повторне використання компонентів, легке тестування модулів і розподілення коду між командами розробників. У результаті така архітектура забезпечує високу гнучкість, керованість та адаптивність до змін у процесі подальшої еволюції системи.

3.4 Вибір інструментарію для створення прикладного програмного забезпечення

Правильний вибір інструментарію програмної реалізації інформаційної системи управління перевезенням пасажирів визначає ефективність розробки, рівень підтримуваності, продуктивність, безпеку та можливості масштабування проєкту. Вибір технологій та середовищ ґрунтувався на таких критеріях: сумісність із архітектурною моделлю, доступність документації, підтримка модульного та об’єктно-орієнтованого

підходу, стабільність при роботі з транзакціями та базами даних, зручність для GUI-розробки й можливість інтеграції з зовнішніми сервісами.

Програмне забезпечення реалізовано на основі мови програмування C# з використанням платформи .NET та архітектурної моделі WinForms у якості графічного інтерфейсу. Для побудови бізнес-логіки обрано розділення на контролери та сервіси із застосуванням шаблонів програмування (переважно dependency injection, repository, unit of work). Роботу з базою даних SQL Server реалізовано через ADO.NET і типізовані запити T-SQL. Для адміністрування БД використано Microsoft SQL Server Management Studio. У тестуванні застосовувались інструменти для логування, емуляції запитів та перевірки транзакційності.

Таблиця 3.6 – Обґрунтований вибір інструментальних засобів

№	Компонент системи	Інструмент / Технологія	Призначення та переваги
1	Мова програмування	C# (.NET 6/7)	Об’єктно-орієнтована, типобезпечна, підтримка WinForms, лямбда-виразів, LINQ
2	Графічний інтерфейс	WinForms	Швидка розробка форм, зручна робота з подієвою моделлю, підтримка розмітки UI
3	Бізнес-логіка	Класи сервісів (BLL)	Ізоляція логіки, гнучка модульна побудова, розширюваність

Продовження таблиці 3.6

4	Доступ до даних	ADO.NET, SqlClient	Прямий доступ до SQL Server, висока продуктивність, контроль транзакцій
5	СУБД	Microsoft SQL Server	Надійність, підтримка ACID, потужні засоби адміністрування
6	Адміністрування БД	SQL Server Management Studio (SSMS)	Візуальне створення таблиць, керування ключами, виконання запитів
7	Організація проєкту	Visual Studio 2022 / 2023	Повний цикл розробки, підтримка .NET, інтеграція з NuGet, профілювання
8	Взаємодія з API	HttpClient, System.Net.Http	Обробка HTTP-запитів, взаємодія з RESTful-сервісами
9	Зовнішні сервіси	SMS/Push API (модульно)	Інтеграція з повідомленнями через стандартні API
10	Логування та налагодження	System.Diagnostics, Debug.WriteLine()	Просте трасування логіки, контроль виконання методів, фіксація помилок

Використаний інструментарій забезпечив ефективну побудову програмної архітектури з дотриманням принципів SOLID, дозволив реалізувати функціональність без надмірної складності, забезпечив високу продуктивність при роботі з великими обсягами

транзакційних даних та створив підґрунтя для масштабування проєкту (включно з можливістю перенесення логіки на ASP.NET Core чи WPF у майбутньому). Комбінація C#, SQL Server і WinForms була обрана як оптимальна для настільної системи середнього рівня складності з розширеним обліково-операційним функціоналом.

3.5 Алгоритми функціонування ключових сценаріїв програми

Алгоритмічна складова прикладного програмного забезпечення для системи таксомоторного обслуговування реалізує послідовну, перевірену й детерміновану логіку взаємодії користувача із системою. Кожна функціональна операція – авторизація, створення замовлення, обробка оплати, голосове введення або залишення чайових – оформлена у вигляді окремого алгоритму, що забезпечує надійність, передбачуваність та контрольованість виконання. Розробка алгоритмів базується на використанні класичних структур управління (розгалуження, перевірки умов, цикли), а їх реалізація здійснена через подієву модель платформи WinForms із застосуванням мови програмування C#.

Один із базових алгоритмів — авторизація користувача — починається з введення логіна й пароля та завершується успішним входом або повідомленням про помилку (Лістинінг коду представлений у ДОДАТОК А). Система зчитує дані з локального файлу users.txt, аналізує наявність відповідного користувача та звіряє правильність пароля. У разі відповідності — система ініціює відкриття головного вікна (наприклад, адміністративного інтерфейсу). У разі помилки відображається відповідне повідомлення. Детальний логічний перебіг процесу показано на рисунку 3.5 у вигляді блок-схеми, яка відображає усі можливі гілки виконання.

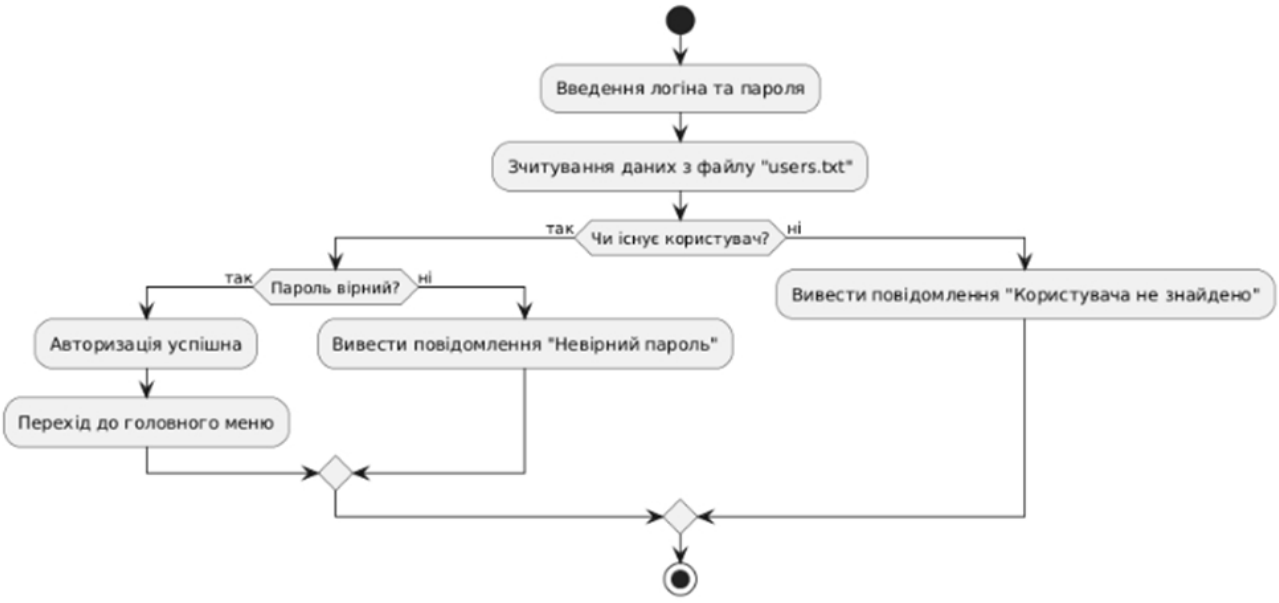


Рисунок 3.5 – Блок-схема авторизації користувача

Відповідний фрагмент коду забезпечує покрокову реалізацію цієї логіки — зчитування даних з файлу, розділення рядків, перевірку умов, виклик методів переходу до головного вікна. Обробка винятків забезпечує стійкість у разі недоступності файлу або помилки у форматі.

Другим ключовим алгоритмом є оформлення замовлення таксі, яке охоплює повну взаємодію користувача з інтерфейсом програми (лістинінг коду представлений у ДОДАТОК Б). Користувач послідовно вводить ім’я, адресу прибуття й призначення, обирає клас авто та спосіб оплати. За необхідності активуються додаткові сервіси: кур’єрська доставка, перевезення тварин, дитяче крісло, перевезення осіб з інвалідністю, замовлення мінівена або додатковий багаж. Також реалізовано розгалуження, що залежить від типу оплати – якщо обрано криптовалюту, відбувається перехід на зовнішній платіжний сервіс; інакше – обробляється стандартна готівкова/безготівкова оплата. Завершальним етапом є перевірка заповнення полів та виведення повідомлення «Таксі замовлено». Цей сценарій ілюструється на рисунку 3.6, який чітко структурує логіку оформлення замовлення.

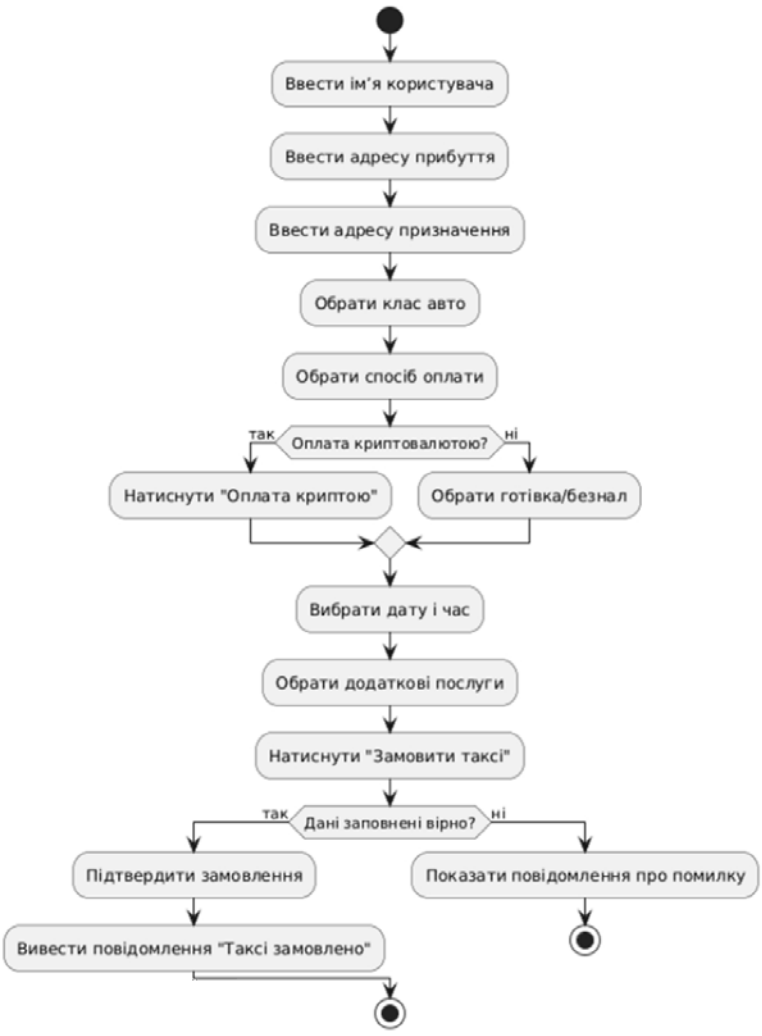


Рисунок 3.6 – Блок-схема оформлення замовлення таксі

Програмна реалізація даного алгоритму включає функцію btnOrderTaxi_Click, що викликається при натисканні кнопки замовлення. У межах цієї функції перевіряються обов’язкові поля, зчитуються значення з усіх елементів інтерфейсу, формується об’єкт замовлення типу OrderDetails і ініціюється збереження замовлення у текстовий файл. Додаткові можливості, як-от голосове введення імені (через інтеграцію з вебсервісом розпізнавання мови) або залишення чайових (введення суми й підтвердження через вікно), реалізовані через окремі обробники подій.

Розгалуження в логіці, наприклад, вибір оплати криптовалютою, обробляється через виклик методу Process.Start() із відкриттям браузера за посиланням на Binance або PayPal. Система не зберігає платіжні дані, що відповідає сучасним вимогам безпеки.

Крім двох основних сценаріїв — авторизації та замовлення, — у програмі реалізовано низку додаткових алгоритмів: збереження замовлення у файл, очищення форми, перевірка вхідних даних, обробка помилок, підтвердження суми чайових, тощо. Вони забезпечують повноцінний користувацький досвід, узгоджений з бізнес-логікою та структурою системи.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

На заключному етапі розробки прикладного програмного забезпечення для інформаційної системи управління перевезенням пасажирів було проведено комплексне тестування функціональних модулів з метою виявлення помилок, перевірки стійкості до некоректних даних та підтвердження відповідності реалізованої логіки технічним вимогам. Основна увага приділялась працездатності ключових сценаріїв: авторизації користувачів, створення замовлень, відображення результатів, введення додаткових параметрів (у т.ч. оплати криптовалютою, голосового введення, підтвердження чайових), а також адмініструванню водіїв.

Тестування здійснювалося вручну в інтерактивному середовищі Windows Forms з візуальним контролем дій користувача. До кожного елемента інтерфейсу та функціонального сценарію було застосовано позитивні та негативні вхідні дані, після чого фіксувалися реакції системи, відповідність повідомлень, робота з виключеннями, збереження та відображення інформації.

Таблиця 4.1 – План функціонального тестування основних модулів

№	Назва сценарію	Вхідні умови	Очікуваний результат	Фактичний результат
1	Успішна авторизація адміністратора	Вірні логін і пароль	Вхід до вікна WindowAdmin	Успішно
2	Авторизація з неправильним паролем	Вірний логін, помилковий пароль	Повідомлення "Невірний пароль"	Успішно
3	Авторизація з неіснуючим логіном	Несуществующий логін	Повідомлення "Користувача не знайдено"	Успішно
4	Створення замовлення (успішне)	Заповнено всі поля, обрано авто, оплату, дату	Повідомлення "Таксі замовлено", збережено у файл	Успішно

Продовження таблиці 4.1

5	Створення замовлення (неповні дані)	Не заповнено адресу призначення	Повідомлення про помилку, замовлення не створено	Успішно

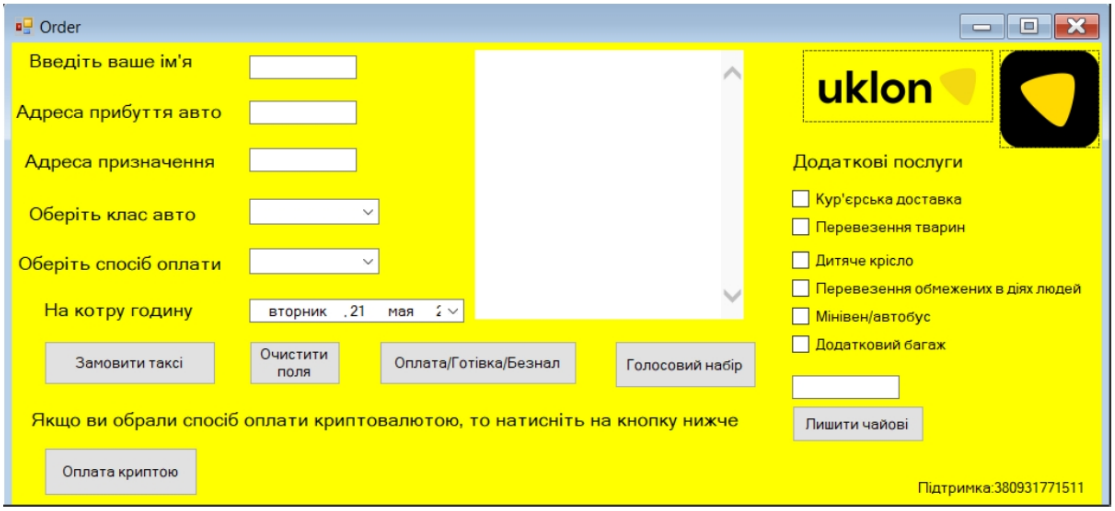
6	Оплата криптовалютою	Обрано відповідний спосіб, натиснуто кнопку	Відкриття сайту Binance у браузері	Успішно
7	Голосовий ввід	Активовано вебпереглядач, отримано текст	Поле імені автоматично заповнено	Успішно
8	Введення і підтвердження чайових	Введено валідну суму (напр., 50 грн)	Повідомлення "Ви залишили 50 грн чайових"	Успішно
9	Помилка введення чайових	Введено текст "п’ятдесят" замість числа	Повідомлення "Введіть правильну суму чайових"	Успішно
10	Робота вікна адміністратора	Введено нові дані про водія, збережено запис	Дані з’явились у таблиці з підтвердженням	Успішно

У результаті виконання тестів були зафіксовані скриншоти, що демонструють поведінку системи при взаємодії з користувачем.

Інтерфейс авторизації адміністратора (рис. 4.1) підтверджує правильну роботу перевірки логіна й пароля. У разі успішного введення — перехід до вікна адміністрування; у разі помилки — відповідне повідомлення без аварійного завершення програми.

Рисунок 4.1 – Інтерфейс авторизації адміністратора

Головна форма замовлення таксі (рис. 4.2) демонструє реалізовану послідовність взаємодії з користувачем: введення даних, вибір послуг, вибір способу оплати, натискання на кнопку замовлення. Було перевірено як коректні, так і неповні сценарії.



Ілюстрація представлена на рис.4.3, порожньої таблиці водіїв до початку взаємодії з базою. Усі поля доступні для введення вручну або імпорту з бази.

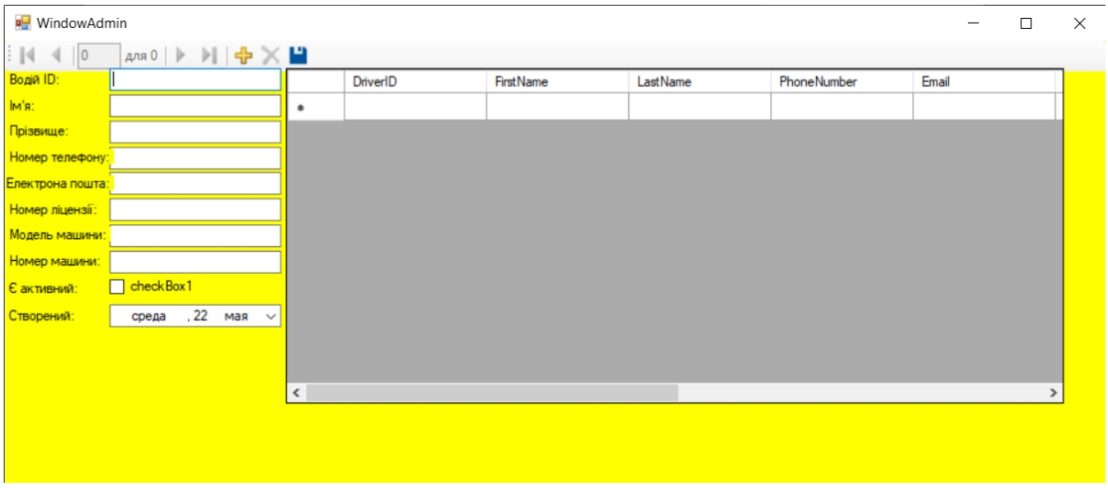


Рисунок 4.3 – Форма адміністратора до додавання даних

Результат успішного тестування додавання водія. У таблиці з'являється новий запис із вказаними значеннями: ID, ім'я, прізвище, телефон, email тощо представлений на рис.4.4

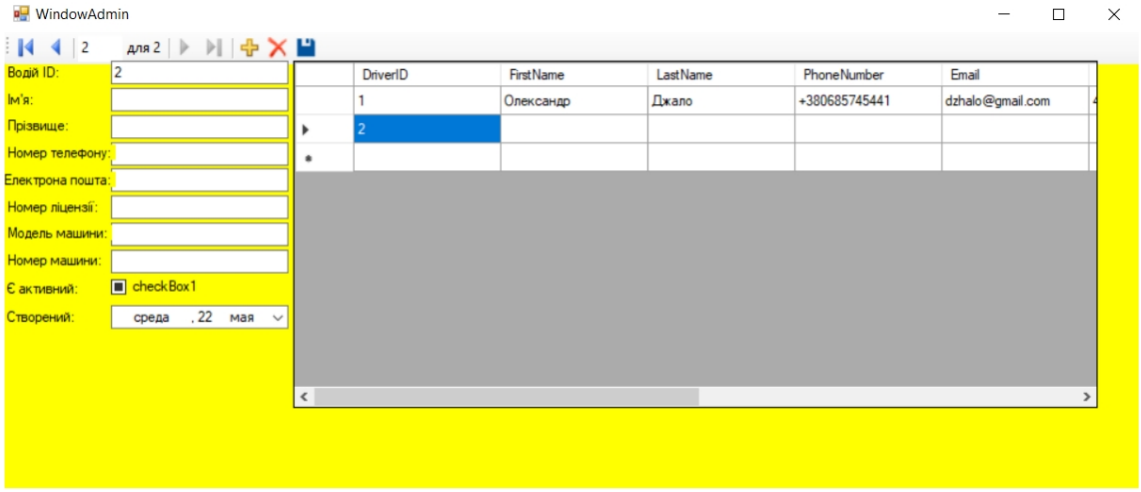


Рисунок 4.4 – Стан системи після збереження нового водія

У наступній серії тестів було перевірено коректність роботи додаткового функціоналу прикладного програмного забезпечення: взаємодії з чекбоксами додаткових послуг, обробки вводу суми чайових, відображення повного підтвердження замовлення та збереження замовлення у файл. Ці функції забезпечують користувацьку зручність, прозорість транзакцій і операційний контроль.

Під час тестування у вікні оформлення поїздки було вибрано кілька опцій додаткових послуг, зокрема перевезення тварин і встановлення дитячого крісла, а також введено суму чайових. На рис. 4.5 показано візуальний стан елементів інтерфейсу з активованими опціями та полем для введення суми чайових.

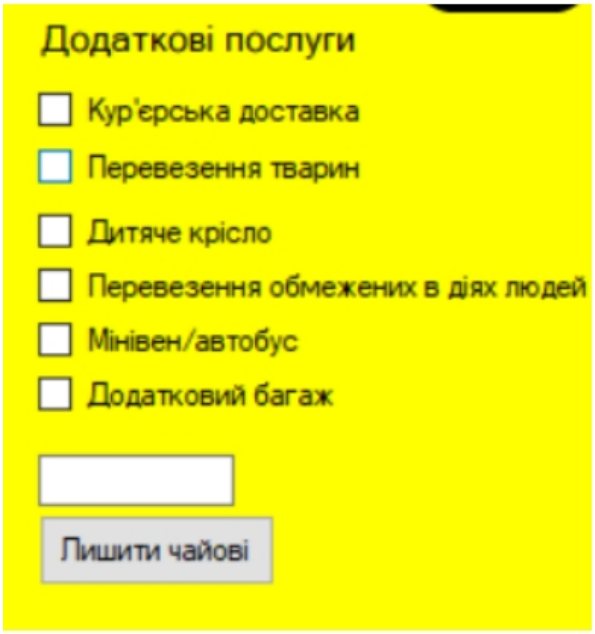


Рисунок 4.5 – Інтерфейс вибору додаткових послуг і введення чайових

Після натискання кнопки підтвердження чайових система автоматично виводить повідомлення про успішне збереження суми, що видно на рис. 4.6. Це підтверджує правильну обробку числового значення та наявність зворотного зв'язку з користувачем.

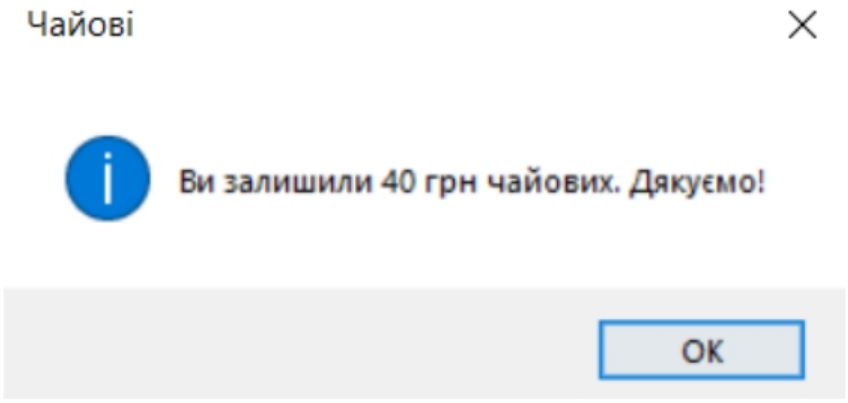


Рисунок 4.6 – Повідомлення про успішне залишення чайових

Формування фінального повідомлення про замовлення з повною деталізацією (ім’я, маршрут, дата, клас авто, тип оплати, обрані послуги) продемонстровано на рис. 4.7. Система інтегрує всі вхідні дані у форматоване інформаційне вікно, що дозволяє користувачу перевірити правильність замовлення до його підтвердження.

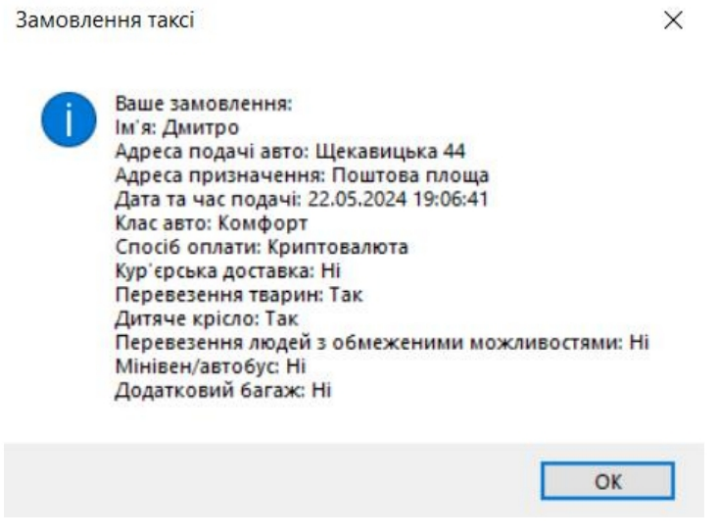


Рисунок 4.7 – Повідомлення про сформоване замовлення

Крім того, автоматичне збереження замовлення у текстовий файл у форматі .txt представлено на рис. 4.8. Файл містить усі введені параметри, включаючи суму чайових (навіть якщо вона дорівнює нулю), що є підтвердженням надійності механізму генерації звітної інформації.

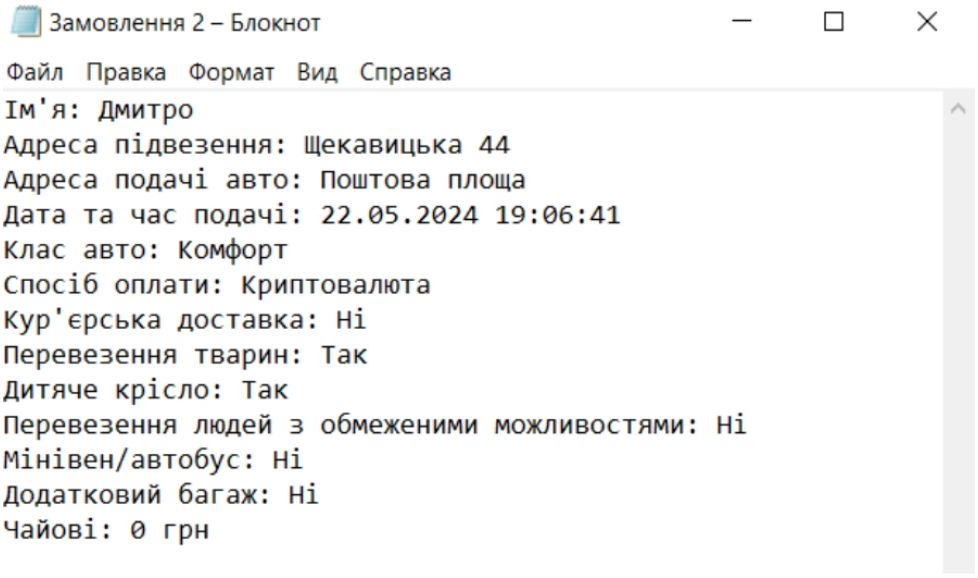


Рисунок 4.8 – Текстовий файл із деталями замовлення

Проведене тестування інформаційної системи управління перевезенням пасажирів підтвердило працездатність її основних функціональних компонентів у повному обсязі. Усі протестовані сценарії — від авторизації користувачів до створення замовлення, вибору

додаткових послуг, обробки оплати, залишення чайових і адміністративного керування даними — успішно виконували свої функції відповідно до заданих технічних вимог.

Система продемонструвала стійкість до некоректного вводу, коректне реагування на виняткові ситуації, наявність інформативних повідомлень для користувача, а також цілісність збереження інформації у внутрішніх структурах та зовнішніх файлах. Візуальна складова графічного інтерфейсу дозволяє інтуїтивно взаємодіяти з програмою, що робить її придатною для використання як адміністраторами, так і кінцевими користувачами.

Функціональне тестування засвідчило відповідність реалізованого прикладного ПЗ очікуваній логіці роботи, його готовність до практичного використання та перспективність подальшого вдосконалення з можливістю масштабування або доповнення новими сервісами.

4.2 Вимоги до апаратного та програмного забезпечення

Реалізація, тестування та розгортання прикладного програмного забезпечення для інформаційної системи управління перевезенням пасажирів потребує відповідного апаратного і програмного забезпечення, яке забезпечує стабільну роботу, високу доступність і взаємодію всіх компонентів системи у клієнт-серверній архітектурі. Згідно з логікою побудови, представлено на рисунку 4.9, програмний комплекс працює у середовищі Docker, що містить окремі сервіси: бекенд-додаток TaxiOrderingApp.exe, API-сервіс для роботи з файловим сховищем TxtStorage API, а також клієнтські файли, що обслуговуються через HTTPS.

Фізичні та віртуальні компоненти взаємодіють за моделлю «робоча станція – сервер – сховище». На стороні користувача достатньо наявності браузера з підтримкою HTTPS-запитів. Серверна частина виконує обробку логіки, маршрутизацію запитів, маніпуляції з файлами clients.txt, drivers.txt, orders.txt, ratings.txt та feedbacks.txt, що розміщені у файловій системі.

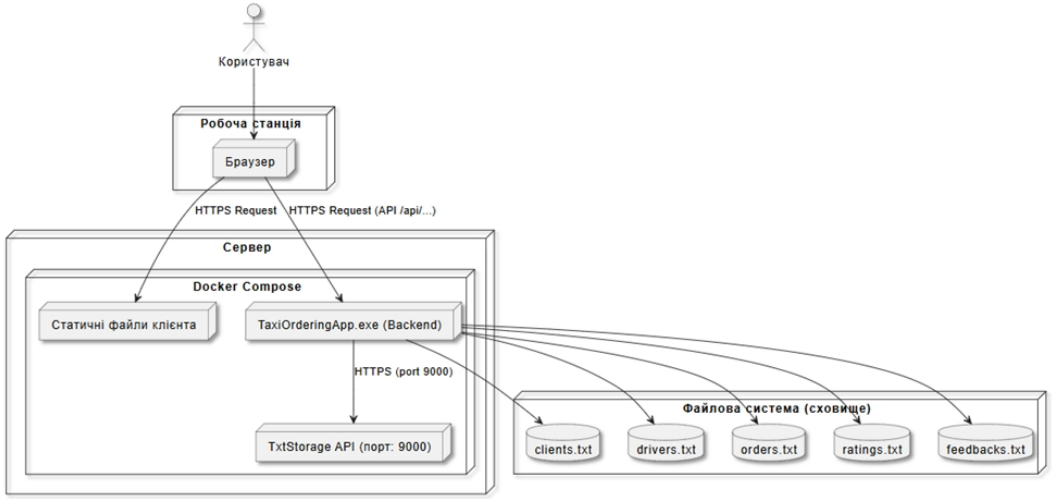


Рисунок 4.9 – Схема розгортання інформаційної системи в середовищі Docker

З метою забезпечення ефективного функціонування програмного забезпечення у різних середовищах, визначено перелік мінімальних і рекомендованих вимог до апаратного та програмного забезпечення. Вимоги поділено на дві категорії: для клієнтської сторони (робоча станція користувача) та для серверної частини (розміщення backend, API та файлового сховища).

Таблиця 4.2 – Вимоги до апаратного та програмного забезпечення

Компонент середовища	Параметр	Мінімальні вимоги	Рекомендовані вимоги
Клієнт	Операційна система	Windows 7 / Linux Ubuntu 18.04+	Windows 10+ / Ubuntu 22.04 LTS
	Браузер	Google Chrome / Mozilla Firefox	Останні версії Chrome / Edge / Firefox
	Оперативна пам’ять	2 ГБ	4 ГБ
	Роздільна здатність дисплея	1024×768	1920×1080
	Підключення до мережі	Локальна мережа або інтернет (HTTPS)	Інтернет ≥ 20 Мбіт/с

Продовження таблиці 4.2

Сервер	Операційна система	Ubuntu Server 20.04	Ubuntu Server 22.04 або Docker-compatible
	Процесор	2 ядра (x86_64, ARM64)	4 ядра, підтримка віртуалізації
	Оперативна пам’ять	4 ГБ	8 ГБ і більше
	Накопичувач	SSD 20 ГБ	SSD 50+ ГБ (з резервуванням)
	Програмне забезпечення	Docker Engine 20.x, Docker Compose, .NET 6	Docker Engine, Docker Compose, Traefik
	Порти	HTTPS (443, 9000), API	Конфігуровані через reverse proxy

У середовищі розгортання серверної частини рекомендовано використання Linux-серверів або хмарної інфраструктури з підтримкою контейнеризації (Docker). Застосування контейнерної архітектури дозволяє швидко масштабувати систему, оновлювати окремі компоненти та забезпечувати ізольовану роботу кожного модуля без впливу на загальну інфраструктуру.

Клієнтська частина функціонує незалежно від операційної системи завдяки використанню вебтехнологій, що підвищує гнучкість і кросплатформеність програмного забезпечення. Таким чином, сформульовані технічні вимоги гарантують надійність, масштабованість та сумісність системи у різноманітних обчислювальних середовищах.

4.3 Склад інсталяційного пакету

Для забезпечення повноцінного розгортання та коректного функціонування інформаційної системи управління перевезенням пасажирів сформовано інсталяційний пакет, який містить усі необхідні компоненти програмного забезпечення, конфігураційні файли, сховище даних, допоміжні інструменти й супровідну документацію. Склад пакету визначено відповідно до обраної архітектури (рис. 4.9) з урахуванням розділення на клієнтську та серверну частину, контейнеризації середовища та ізоляції доступу до даних.

Усі елементи пакету згруповані відповідно до функціонального призначення та включені до репозиторію або архіву, який може бути розгорнутий локально або на сервері. Нижче наведено повну структурну специфікацію інсталяційного пакету.

Таблиця 4.3 – Структура інсталяційного пакету інформаційної системи

№	Назва компонента	Тип файлу	Призначення
1	`TaxiOrderingApp.exe`	.exe	Основний серверний модуль (бекенд), що обробляє логіку системи
2	`TxtStorage.dll`	.dll	API-компонент для ізольованого доступу до текстового сховища
3	`clients.txt`, `drivers.txt`, `orders.txt`, `ratings.txt`, `feedbacks.txt`	.txt	Файли даних, що зберігають інформацію про клієнтів, водіїв, замовлення тощо
4	`Order.exe`, `Admin.exe`	.exe	Графічні інтерфейси клієнта і адміністратора на основі Windows Forms
5	`docker-compose.yml`	.yml	Файл конфігурації для контейнеризації системи через Docker
6	`run.sh`, `install.ps1`	.sh / .ps1	Сценарії запуску, перевірки залежностей, налаштування середовища
7	`README.md`, `manual.pdf`	.md / .pdf	Інструкції користувача та адміністратора щодо встановлення та використання
8	`/assets/`	Каталог	Статичні файли інтерфейсу (іконки, логотипи, стилі)
9	`config.json`	.json	Налаштування параметрів запуску, портів, шляхи до сховища

Інсталяційний пакет розроблений із дотриманням принципів модульності, кросплатформеності та масштабованості. Завдяки контейнерній ізоляції всі компоненти можуть бути запущені як у локальному середовищі Windows/Linux, так і у хмарній інфраструктурі. Сценарії інсталяції адаптовані для автоматичного налаштування мережевих портів, ініціалізації сховища та перевірки залежностей.

Сформований інсталяційний пакет забезпечує повну готовність системи до розгортання, включає всі технічні ресурси для запуску та підтримки роботи, а також надає супровідну документацію для користувачів і адміністраторів, що відповідає вимогам до сучасного прикладного ПЗ.

ВИСНОВОК

У результаті виконання дипломної роботи було розроблено, реалізовано та протестовано інформаційну систему управління перевезенням пасажирів, що забезпечує автоматизацію основних бізнес-процесів служби таксі, включаючи створення та обробку замовлень, адміністрування персоналу, інтеграцію з зовнішніми інтерфейсами та ведення зворотного зв’язку з користувачем. Створена система відповідає вимогам функціональності, масштабованості, доступності й забезпечує зручну взаємодію як для кінцевих користувачів (пасажирів), так і для операторів та адміністраторів служби.

У першому розділі було обґрунтовано актуальність теми, визначено предметну область, досліджено сучасні рішення в галузі автоматизації таксомоторного обслуговування та сформовано загальні вимоги до майбутньої системи. На основі цього було розроблено концептуальну модель програмного забезпечення, яка охоплює ключові сценарії та рольову модель взаємодії.

У другому розділі реалізовано проєктування системи відповідно до вимог користувачів. Побудовано діаграми прецедентів, класів, компонентів та активностей, що демонструють внутрішню логіку і структуру додатку. Визначено функціональні вимоги, архітектуру програмного забезпечення, склад модулів та механізм зберігання даних у вигляді фізичної моделі з використанням текстових файлових сховищ. Розроблено SQL-запити до системи управління базами даних для обробки інформації про водіїв, клієнтів, замовлення, оцінки та відгуки.

У третьому розділі було реалізовано прикладне програмне забезпечення на основі архітектурної моделі, що включає серверну частину (TaxiOrderingApp.exe), API для роботи з текстовими файлами та клієнтські інтерфейси для користувача і адміністратора. Розроблено алгоритмічні блок-схеми сценаріїв входу в систему, створення замовлення, голосового введення та залишення чайових. Реалізовано можливості гнучкого вибору послуг, оплату криптовалютою, підтримку додаткових сервісів і ведення локального журналу замовлень. Сформовано вимоги до інсталяції системи та описано склад інсталяційного пакету, що дозволяє швидке розгортання на сервері за допомогою Docker.

У четвертому розділі проведено повне функціональне тестування системи, яке підтвердило її коректну роботу в типових і граничних сценаріях. Перевірено обробку введення, виведення повідомлень, коректність запису в сховище, відображення інформації у графічному інтерфейсі. Підтверджено відповідність розробленого ПЗ встановленим функціональним і нефункціональним вимогам. Визначено апаратно-програмне забезпечення,

необхідне для ефективної експлуатації, та підтверджено сумісність з кросплатформенними середовищами.

Узагальнюючи результати, можна зробити висновок, що розроблена інформаційна система відповідає критеріям сучасного програмного забезпечення, є стабільною, масштабованою, зручною у використанні й придатною для подальшого впровадження у реальних умовах роботи служби таксі. Реалізовані архітектурні та технологічні рішення створюють надійну основу для розширення функціоналу, інтеграції з платіжними сервісами, мобільними клієнтами або геолокаційними API в майбутньому.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про інформацію» від 02.10.1992 № 2657-XII [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2657-12
2. Закон України «Про захист персональних даних» від 01.06.2010 № 2297-VI [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17
3. ДСТУ ISO/IEC 12207:2006. Інформаційні технології. Процеси життєвого циклу програмного забезпечення. – [Чинний від 2006-12-01].
4. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. – [Чинний від 01.07.2016].
5. Sommerville, I. Software Engineering. 10th Edition. – Boston: Pearson, 2016. – 792 p.
6. Pressman, R. S., Maxim, B. R. Software Engineering: A Practitioner's Approach. – 9th ed. – McGraw-Hill Education, 2019. – 936 p.
7. Грінченко С. М., Шестопалов Є. О. Системний аналіз в ІТ: Навчальний посібник. – Харків: ХНУРЕ, 2021. – 248 с.
8. Брауде Е. Дж., Бернштейн М. Основи розробки програмного забезпечення. – К.: Діалектика, 2020. – 464 с.
9. Бондаренко А. В., Костюченко Ю. І. Основи проектування інформаційних систем. – К.: КНЕУ, 2018. – 300 с.
10. Шаповал М. І., Ковальов А. М. Системи баз даних. – К.: НАУ, 2020. – 376 с.
11. Npgsql – .NET Data Provider for PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.npgsql.org/
12. MSDN documentation: Windows Forms [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/
13. PostgreSQL documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/
14. Мальцев В. В., Гресь О. І. Програмна інженерія: методи, моделі, засоби. – Харків: ХНУРЕ, 2022. – 344 с.
15. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. – International Standard, 2018.
16. Літвінов В. В., Остроух А. І. Архітектура програмного забезпечення. – Харків: ХНЕУ, 2021. – 288 с.
17. Пащенко Д. В. Засоби автоматизованого проєктування програмного забезпечення. – К.: Ліра-К, 2019. – 214 с.

18. Freeman, E., Robson, E. Head First Design Patterns. – 2nd ed. – O'Reilly Media, 2021. – 672 p.

19. Дж. Сонмез. Чистий код. Створення, аналіз і рефакторинг. – К.: Видавництво «Наш Формат», 2020. – 320 с.

20. Uber [Електронний ресурс]. – Режим доступу: https://www.uber.com – Назва з екрана.

21. Bolt: сервіс виклику таксі [Електронний ресурс]. – Режим доступу: https://bolt.eu – Назва з екрана.

22. Uklon – замовлення таксі онлайн [Електронний ресурс]. – Режим доступу: https://uklon.com.ua – Назва з екрана.

23. Opti – служба таксі в Україні [Електронний ресурс]. – Режим доступу: https://opti.global – Назва з екрана.

ДОДАТОК А

Лістинінг коду входу в акаунт

```
        private async void button1_Click(object sender, EventArgs e)
        {
            string username = textBox1.Text;
            string password = textBox2.Text;
            try
            {
                var connStringBuilder = new NpgsqlConnectionStringBuilder
                {
                string enteredLogin = textBoxLogin.Text;
                string enteredPassword = textBoxPassword.Text;
                bool loginSuccess = false;
        foreach (var line in File.ReadAllLines("users.txt"))
        {
            var parts = line.Split(';');
            if (parts.Length == 2)
            {
                string login = parts[0];
                string password = parts[1];

                if (login == enteredLogin && password == enteredPassword)
                {
                    loginSuccess = true;
                    break;
                }       };
                using          (NpgsqlConnection          conn          =          new
NpgsqlConnection(connStringBuilder.ConnectionString))
                {
                    await conn.OpenAsync();
                    MessageBox.Show("Вітаю! Ви увійшли в систему як адміністратор");
                    WindowAdmin desktopForm = new WindowAdmin(username, password);
                    desktopForm.Show();
                    this.Hide();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка: " + ex.Message);
```

ДОДАТОК Б

Лістинінг коду замовлення авто

```
        public partial class Order : Form
        {
            private readonly string _username;
            private readonly string _password;
            NpgsqlConnection conn;

            public Order(string username, string password)
            {
                InitializeComponent();
                webBrowser1.ObjectForScripting = new ScriptManager(this); // Додано
для інтеграції з WebBrowser
                _username = username;
                _password = password;
                conn                                =                                new
NpgsqlConnection($"Server=localhost;Port=5432;Database=taxiservicedb;User
ID={_username};Password={_password}");
            }

            private void Order_Load(object sender, EventArgs e)
            {

            }

            private void btnOrderTaxi_Click(object sender, EventArgs e)
            {
                if (ValidateInput()) // Перевірка правильності введених даних
                {
                    // Створення нового замовлення
                    var newOrder = new OrderDetails();

                    // Заповнення даних з форми
                    newOrder.CustomerName = txtCustomerName.Text;
                    newOrder.PickupAddress = txtPickupAddress.Text;
                    newOrder.DestinationAddress = txtDestinationAddress.Text;
                    newOrder.PickupDateTime = dateTimePickerPickup.Value;
                    newOrder.CarClass = comboCarClass.SelectedItem?.ToString();
      newOrder.PaymentMethod                                                =
comboPaymentMethod.SelectedItem?.ToString();

                    // Додаткові послуги
                    newOrder.CourierService = chkCourierService.Checked;
                    newOrder.PetTransport = chkPetTransport.Checked;
                    newOrder.ChildSeat = chkChildSeat.Checked;
                    newOrder.DisabilityTransport = chkDisabilityTransport.Checked;
                    newOrder.Minivan = chkMinivan.Checked;
                    newOrder.ExtraLuggage = chkExtraLuggage.Checked;

                    // Відображення підтвердження замовлення
                    MessageBox.Show($"Ваше                          замовлення:\nІм'я:
{newOrder.CustomerName}\n" +
                                    $"Адреса              подачі              авто:
{newOrder.PickupAddress}\n" +
                                    $"Адреса                            призначення:
{newOrder.DestinationAddress}\n" +
                                    $"Дата          та          час          подачі:
{newOrder.PickupDateTime}\n" +
                                    $"Клас авто: {newOrder.CarClass}\n" +
                                    $"Спосіб оплати: {newOrder.PaymentMethod}\n" +
                                    $"Кур'єрська доставка: {(newOrder.CourierService
? "Так" : "Ні")}\n" +
```

```
                                        $"Перевезення тварин: {(newOrder.PetTransport ? "Так" : "Ні")}\n" +
                                        $"Дитяче крісло: {(newOrder.ChildSeat ? "Так" : "Ні")}\n" +
                                        $"Перевезення людей з обмеженими можливостями: {(newOrder.DisabilityTransport ? "Так" : "Ні")}\n" +
                                        $"Мінівен/автобус: {(newOrder.Minivan ? "Так" : "Ні")}\n" +
                                        $"Додатковий багаж: {(newOrder.ExtraLuggage ? "Так" : "Ні")}",
                                        "Замовлення таксі",
                                        MessageBoxButtons.OK,
                                        MessageBoxIcon.Information);

                    // Збереження замовлення в файл
                    SaveOrderToFile(newOrder);
                }
            }

            private void btnClear_Click(object sender, EventArgs e)
            {
                // Очищення полів введення
                txtCustomerName.Clear();
                txtPickupAddress.Clear();
                txtDestinationAddress.Clear();
                dateTimePickerPickup.Value = DateTime.Now;
                comboCarClass.SelectedIndex = -1;
                comboPaymentMethod.SelectedIndex = -1;

                // Скидання додаткових послуг
                chkCourierService.Checked = false;
                chkPetTransport.Checked = false;
                chkChildSeat.Checked = false;
                chkDisabilityTransport.Checked = false;
                chkMinivan.Checked = false;
                chkExtraLuggage.Checked = false;

                // Сховати повідомлення про помилки
                errorProvider.Clear();
            }

            private bool ValidateInput()
            {
                // Перевірка, чи заповнені всі обов'язкові поля
                if (string.IsNullOrWhiteSpace(txtCustomerName.Text))
                {
                    errorProvider.SetError(txtCustomerName, "Введіть ваше ім'я");
                    return false;
                }

                if (string.IsNullOrWhiteSpace(txtPickupAddress.Text))
                {
                    errorProvider.SetError(txtPickupAddress, "Введіть адресу подачі авто");
                    return false;
                }

                if (string.IsNullOrWhiteSpace(txtDestinationAddress.Text))
                {
                    errorProvider.SetError(txtDestinationAddress, "Введіть адресу призначення");
                    return false;
                }

                if (comboCarClass.SelectedItem == null)
                {
```

```
                errorProvider.SetError(comboCarClass, "Виберіть клас авто");
                return false;
            }

            if (comboPaymentMethod.SelectedItem == null)
            {
                errorProvider.SetError(comboPaymentMethod,    "Виберіть    спосіб
оплати");
                return false;
            }

            return true;
        }

        private void SaveOrderToFile(OrderDetails order)
        {
            // Діалогове вікно для збереження файлу
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "Текстові файли (*.txt)|*.txt";
            saveFileDialog.FileName                                                =
$"Order_{DateTime.Now:yyyyMMddHHmmss}.txt";

            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                // Збереження замовлення в файл
                using         (StreamWriter         writer         =         new
StreamWriter(saveFileDialog.FileName))
                {
                    writer.WriteLine($"Ім'я: {order.CustomerName}");
                    writer.WriteLine($"Адреса                          підвезення:
{order.PickupAddress}");
                    writer.WriteLine($"Адреса            подачі            авто:
{order.DestinationAddress}");
                    writer.WriteLine($"Дата        та        час        подачі:
{order.PickupDateTime}");
                    writer.WriteLine($"Клас авто: {order.CarClass}");
                    writer.WriteLine($"Спосіб оплати: {order.PaymentMethod}");
                    writer.WriteLine($"Кур'єрська                      доставка:
{(order.CourierService ? "Так" : "Ні")}");
                    writer.WriteLine($"Перевезення  тварин:  {(order.PetTransport
? "Так" : "Ні")}");
                    writer.WriteLine($"Дитяче крісло: {(order.ChildSeat ? "Так"
: "Ні")}");
                    writer.WriteLine($"Перевезення    людей    з    обмеженими
можливостями: {(order.DisabilityTransport ? "Так" : "Ні")}");
                    writer.WriteLine($"Мінівен/автобус: {(order.Minivan ? "Так"
: "Ні")}");
                    writer.WriteLine($"Додатковий багаж: {(order.ExtraLuggage ?
"Так" : "Ні")}");
                    writer.WriteLine($"Чайові: {order.TipAmount} грн");
                }
            }
        }

        private void CryptoPay_Click(object sender, EventArgs e)
        {
            Process.Start(new ProcessStartInfo
            {
                FileName = "https://www.binance.com/uk-UA",
                UseShellExecute = true
            });
        }

        private void PayPal_Click(object sender, EventArgs e)
        {
            Process.Start(new ProcessStartInfo
```

```
                {
                    FileName = "https://www.paypal.com/ru/home",
                    UseShellExecute = true
                });
            }

            // Додаємо кнопку для ініціації голосового вводу
            private void btnVoiceInput_Click(object sender, EventArgs e)
            {
                // Відкриваємо сторонній сайт для голосового введення
                webBrowser1.Navigate("https://www.textfromtospeech.com/uk/voice-to-
text/");
            }
            // Метод для отримання розпізнаного тексту з браузера
            public void SetText(string text)
            {
                txtCustomerName.Text = text; // Збереження розпізнаного тексту у
відповідне текстове поле
            }
            //Обробник події для кнопки "Лишити чайові"
            private void btnLeaveTip_Click(object sender, EventArgs e)
            {
                if (decimal.TryParse(txtTipAmount.Text, out decimal tipAmount))
                {
                    MessageBox.Show($"Ви залишили {tipAmount} грн чайових.
Дякуємо!", "Чайові", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    MessageBox.Show("Будь ласка, введіть правильну суму чайових.",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        [System.Runtime.InteropServices.ComVisible(true)]
        public class ScriptManager
        {
            private Order form;

            public ScriptManager(Order form)
            {
                this.form = form;
            }

            public void Notify(string text)
            {
                form.SetText(text);
            }
        }
        // Клас, який представляє деталі замовлення таксі
        class OrderDetails
        {
            public string CustomerName { get; set; }
            public string PickupAddress { get; set; }
            public string DestinationAddress { get; set; }
            public DateTime PickupDateTime { get; set; }
            public string CarClass { get; set; }
            public string PaymentMethod { get; set; }
            public bool CourierService { get; set; }
            public bool PetTransport { get; set; }
            public bool ChildSeat { get; set; }
            public bool DisabilityTransport { get; set; }
            public bool Minivan { get; set; }
            public bool ExtraLuggage { get; set; }
            public decimal TipAmount { get; set; } // Додано поле для чайових
        }
```

```
}
```