МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
на тему:

«Система обліку транспортних засобів на автостоянці»

Виконав здобувач освіти групи П-42
Сальніков Павло Дмитрійович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:
Устименко Людмила Миколаївна

Житомир – 2025

Зміст

РЕФЕРАТ

Записка: 51 с., 20 рис., 1 таблиця, 1 додаток, 20 джерел.

У дипломній роботі розроблено систему обліку транспортних засобів на автостоянці з використанням Windows Forms, .NET 8 та SQLite. Метою роботи є автоматизація процесів реєстрації заїзду та виїзду авто, управління паркувальними місцями та забезпечення зручного інтерфейсу для адміністраторів. Актуальність зумовлена зростанням потреби в ефективному управлінні паркувальними просторами малих і середніх автостоянок.

Розглянуто аналоги (комерційні системи, відкритий код, локальні додатки), обрано локальний настільний додаток за простоту та економічність. Проаналізовано алгоритми додавання, виїзду, видалення та відображення даних із часовою складністю O(1)–O(n). Спроєктовано класи (Vehicle, ParkingSpot, ParkingContext) та базу даних із таблицями Vehicles і ParkingSpots, пов’язаними через зовнішній ключ. Реалізовано функціонал: облік авто, вибір місця, додавання фото, управління місцями.

Система протестована на тестових даних, працює на Windows 10+ із мінімальними вимогами (1 ГГц процесор, 2 ГБ ОЗУ). Інтерфейс включає таблиці, кнопки та діалогові вікна для зручності. Інструкція описує всі операції. Результати підтверджують ефективність системи для малих автостоянок із можливістю розширення (звіти, інтеграція). Робота має практичне значення для автоматизації обліку.

Ключові слова: автостоянка, облік транспортних засобів, Windows Forms, .NET 8, SQLite, автоматизація, інтерфейс користувача.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – Система управління базами даних. Програмне забезпечення для створення, зберігання й обробки даних у структурованому вигляді.

UI – User Interface (інтерфейс користувача). Частина програми, через яку користувач взаємодіє із системою.

ORM – Object-Relational Mapping (об’єктно-реляційне відображення). Технологія, що дозволяє представляти дані бази як об’єкти програмної мови.

EF Core – Entity Framework Core. Фреймворк для роботи з базами даних у .NET, що реалізує ORM.

SQLite – Легка вбудована СУБД, яка зберігає базу даних у вигляді одного файлу.

.NET 8 – Платформа розробки від Microsoft, версія 8, що використовується для створення програми.

CLR – Common Language Runtime. Виконавче середовище .NET, що забезпечує запуск коду.

C# – Мова програмування, на якій реалізовано систему.

Windows Forms – Фреймворк для створення настільних додатків із графічним інтерфейсом у Windows.

ВСТУП

Сучасний розвиток транспортної інфраструктури та зростання кількості транспортних засобів у містах призводять до збільшення попиту на ефективне управління паркувальними просторами. Автостоянки, як комерційні, так і муніципальні, стикаються з необхідністю автоматизації процесів обліку транспортних засобів для підвищення продуктивності, зменшення людського фактору та забезпечення зручності для користувачів. У цьому контексті розробка програмного забезпечення для системи обліку транспортних засобів на автостоянці стає актуальним завданням, що дозволяє оптимізувати процеси заїзду, виїзду, моніторингу зайнятості місць та ведення документації.

Метою даної дипломної роботи є створення системи обліку транспортних засобів на автостоянці з використанням сучасних технологій розробки програмного забезпечення. Запропонована система спрямована на автоматизацію ключових операцій, таких як реєстрація транспортних засобів при заїзді, фіксація часу перебування, призначення вільних паркувальних місць та обробка виїзду, що забезпечить підвищення ефективності роботи автостоянки та зручність для її адміністраторів.

Актуальність теми зумовлена необхідністю впровадження інформаційних технологій у сферу управління транспортними потоками, що сприяє зменшенню витрат часу на рутинні операції, підвищенню точності обліку та покращенню якості обслуговування клієнтів. Розробка такої системи має практичне значення для малих і середніх автостоянок, де автоматизація може суттєво вплинути на операційну діяльність без значних фінансових вкладень у складні апаратно-програмні комплекси.

Для реалізації системи обрано технологію Windows Forms на платформі .NET 8, яка забезпечує зручний інтерфейс користувача та стабільну роботу в середовищі Windows. Як базу даних використано SQLite – легку та ефективну систему управління базами даних, що ідеально підходить для локальних додатків із невеликим обсягом даних. Вибір цих технологій обґрунтований

їхньою доступністю, простотою інтеграції та достатньою функціональністю для вирішення поставлених завдань.

У ході роботи передбачається розробити програмний продукт, який включатиме функціонал для перегляду, додавання, редагування та видалення записів про транспортні засоби та паркувальні місця, а також забезпечуватиме зв’язок між цими сутностями для відстеження зайнятості стоянки. Очікується, що реалізована система стане практичним інструментом для адміністраторів автостоянок, сприяючи оптимізації їхньої роботи та підвищенню рівня автоматизації процесів.

Розділи дипломної роботи охоплюють аналіз предметної області, обґрунтування вибору технологій, проектування структури системи, опис розробленого програмного забезпечення та оцінку його ефективності. У результаті буде представлено готовий продукт, протестований на тестових даних, із рекомендаціями щодо його впровадження та подальшого вдосконалення.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка системи обліку транспортних засобів на автостоянці є відповіддю на потребу в автоматизації процесів управління паркувальним простором, що набуває особливого значення в умовах зростання кількості автомобілів та обмеженості ресурсів для їх розміщення. Основна задача розробки полягає у створенні програмного забезпечення, яке забезпечить ефективний облік транспортних засобів, що заїжджають і виїжджають з автостоянки, а також моніторинг стану паркувальних місць у реальному часі. Система має бути зручною для користувачів, надійною в експлуатації та відповідати сучасним вимогам до програмних продуктів.

Конкретно, основна задача включає наступні ключові аспекти:

- Забезпечення можливості введення даних про транспортний засіб (номерний знак, тип авто), автоматична фіксація часу прибуття та призначення вільного паркувального місця.
- Фіксація часу виїзду транспортного засобу та звільнення зайнятого місця для подальшого використання.
- Відображення статусу зайнятості місць, можливість додавання, редагування та видалення записів про місця на стоянці.
- Створення структури для зберігання інформації про транспортні засоби та паркувальні місця з можливістю швидкого доступу до неї.
- Розробка інтуїтивно зрозумілого графічного інтерфейсу, який дозволить адміністратору автостоянки легко виконувати всі необхідні операції.

Для реалізації системи необхідно врахувати обмеження, пов’язані з її масштабом: вона орієнтована на локальне використання на невеликих або середніх автостоянках, де немає потреби в складних мережевих рішеннях чи інтеграції з зовнішніми системами (наприклад, відеоспостереженням чи платіжними терміналами). Водночас система повинна бути гнучкою для

подальшого розширення функціоналу, наприклад, додавання звітів чи інтеграції з апаратними засобами.

Технічні вимоги до розробки включають вибір технологій, які забезпечують стабільність, швидкодію та простоту впровадження. Програмне забезпечення має працювати на платформі Windows, оскільки це найпоширеніша операційна система для настільних комп'ютерів, які використовуються в адміністративних цілях. База даних повинна бути легкою, портативною та не вимагати складного адміністрування, що зумовлює вибір SQLite. Основним завданням є також забезпечення коректної взаємодії між графічним інтерфейсом, логікою обробки даних і базою даних для створення цілісного продукту.

Таким чином, основна задача розробки полягає в проектуванні та реалізації програмної системи, яка автоматизує облік транспортних засобів на автостоянці, оптимізує управління паркувальними місцями та забезпечує зручність для адміністратора. Успішне вирішення цієї задачі дозволить підвищити ефективність роботи автостоянки, мінімізувати ручні операції та створити основу для подальшого вдосконалення системи. Наступні підрозділи цього розділу будуть присвячені аналізу технологій, які найкраще відповідають поставленим вимогам.

1.2. Огляд та порівняння аналогічних систем.

Автоматизація обліку транспортних засобів на автостоянках є популярним напрямком у сфері інформаційних технологій, тому на ринку існує низка аналогічних систем, які вирішують подібні завдання. У цьому підрозділі розглядаються основні типи таких систем, їхні особливості, переваги та недоліки, а також проводиться порівняння з метою визначення оптимального підходу до розробки запропонованої системи.

1.2.1. Комерційні системи

Комерційні рішення, такі як **ParkMan**, **ParkOffice** та **SmartParking**, є широко відомими прикладами програмного забезпечення для управління паркуванням. Ці системи зазвичай пропонують комплексний функціонал, що включає:

- Інтеграцію з апаратними засобами (камери для розпізнавання номерів, шлагбауми, датчики зайнятості).
- Мобільні додатки для користувачів (бронювання місць, оплата онлайн).
- Генерацію звітів і аналітику (наприклад, статистика завантаженості стоянки).

Переваги:

- Високий рівень автоматизації та інтеграції.
- Підтримка великих паркувальних комплексів.
- Наявність технічної підтримки від розробників.

Недоліки:

- Висока вартість впровадження та обслуговування, що робить їх менш доступними для малих автостоянок.
- Залежність від мережевих технологій (хмарні сервіси), що може ускладнити роботу в умовах обмеженого доступу до Інтернету.
- Надлишковий функціонал для локальних потреб невеликих стоянок.

1.2.2. Системи з відкритим кодом

Існують також проєкти з відкритим кодом, наприклад, **OpenPark** або модулі для платформ на кшталт Odoo, які дозволяють користувачам

адаптувати систему під свої потреби. Такі системи зазвичай включають базові функції:

- Реєстрацію транспортних засобів.
- Управління паркувальними місцями.
- Локальне зберігання даних.

Переваги:

- Безкоштовність і можливість модифікації коду.
- Гнучкість у налаштуванні під конкретні вимоги.

Недоліки:

- Відсутність офіційної підтримки, що вимагає від розробника значних зусиль для адаптації та усунення помилок.
- Обмежений інтерфейс користувача, який часто потребує доопрацювання.
- Необхідність технічних знань для встановлення та налаштування.

1.2.3. Локальні настільні додатки

Прикладами локальних настільних систем є невеликі розробки, створені для конкретних автостоянок, наприклад, програми на базі Microsoft Access або прості додатки на C#. Такі системи зазвичай мають:

- Реєстрацію заїзду та виїзду авто.
- Базу даних для зберігання інформації.
- Простий графічний інтерфейс.

Переваги:

- Низька вартість розробки та впровадження.
- Незалежність від Інтернету, що підходить для локального використання.
- Простота в освоєнні для адміністраторів без спеціальних навичок.

Недоліки:

- Обмежений функціонал порівняно з комерційними системами.
- Відсутність інтеграції з апаратними засобами.
- Складність масштабування для великих стоянок.

1.2.4. Порівняння систем

Для оцінки аналогів використаємо такі критерії: вартість, простота впровадження, функціональність, адаптивність і залежність від мережі. Таблиця 1.1 наводить порівняльний аналіз.

Таблиця 1.2.4.1.
Порівняння аналогічних систем

Тип системи	Вартість	Простота впровадження	Функціональність	Адаптивність	Залежність від мережі
Комерційні системи	Висока	Низька	Висока	Середня	Висока
Відкритий код	Низька	Середня	Середня	Висока	Низька
Локальні додатки	Низька	Висока	Середня	Низька	Низька

З таблиці видно, що комерційні системи підходять для великих паркувальних комплексів із значним бюджетом, тоді як системи з відкритим кодом вимагають додаткових зусиль для адаптації. Локальні настільні додатки є оптимальним вибором для невеликих автостоянок завдяки простоті впровадження та низькій вартості, що відповідає цілям цієї роботи.

Аналіз аналогічних систем показав, що для невеликої автостоянки доцільно розробити локальний настільний додаток, який не потребує складної інфраструктури та зовнішніх сервісів. Запропонована система має поєднувати простоту використання, базовий функціонал для обліку транспортних засобів і паркувальних місць, а також можливість роботи в автономному режимі. На відміну від комерційних рішень, акцент робиться на мінімалізації витрат і легкості освоєння, а порівняно з відкритим кодом – на створенні готового продукту з інтуїтивним інтерфейсом. Ці висновки слугуватимуть основою для вибору технологій у наступному підрозділі.**Вибір та обґрунтування технологій розробки**

Для реалізації системи обліку транспортних засобів на автостоянці необхідно обрати технології, які відповідають поставленим завданням, забезпечують ефективність розробки, стабільність роботи та зручність використання. У цьому підрозділі розглядаються обрані технології – Windows

Forms, .NET 8 та SQLite – з обґрунтуванням їхнього вибору на основі аналізу вимог і порівняння з альтернативами.

1.3.1. Технологія створення інтерфейсу - Windows Forms

Для розробки графічного інтерфейсу користувача обрано **Windows Forms** – фреймворк від Microsoft, призначений для створення настільних додатків із зручним візуальним дизайном.

Основні причини вибору:

- Windows Forms пропонує інтуїтивний дизайнер у Visual Studio, що дозволяє швидко створювати форми, кнопки, таблиці та інші елементи інтерфейсу без необхідності глибоких знань у програмуванні UI.
- Оскільки система орієнтована на локальне використання на комп'ютерах із операційною системою Windows, яка є стандартом для адміністративних робочих станцій, Windows Forms є природним вибором.
- Для базових операцій (перегляд, додавання, редагування, видалення записів) можливостей Windows Forms цілком достатньо, що відповідає потребам невеликої автостоянки.

Розглядалися WPF (Windows Presentation Foundation) і WinUI. WPF має переваги у створенні сучасних інтерфейсів із підтримкою анімацій і складних стилів, але його складність і більші вимоги до ресурсів роблять його надлишковим для цього проєкту. WinUI, хоча й є новішою технологією, потребує додаткового часу на освоєння та орієнтована на Windows 10/11, що може обмежити сумісність із старішими системами. Таким чином, Windows Forms обрано як оптимальний баланс між простотою, швидкістю розробки та відповідністю вимогам.

1.3.2. Платформа розробки - NET 8

Для реалізації логіки програми обрано **.NET 8** – останню версію платформи .NET від Microsoft на момент розробки. .NET 8 пропонує покращену швидкодію порівняно з попередніми версіями завдяки оптимізації роботи CLR (Common Language Runtime) і підтримці сучасних апаратних

можливостей. .NET 8 повністю підтримує Windows Forms, що забезпечує безперебійну інтеграцію інтерфейсу з логікою програми. Платформа надає доступ до широкого набору бібліотек, зокрема Entity Framework Core для роботи з базами даних, що спрощує розробку.

Використання нової версії гарантує підтримку та оновлення від Microsoft у найближчі роки, що робить систему перспективною для подальшого розвитку.

Розглядалися .NET Framework 4.x і старіші версії .NET Core. .NET Framework є застарілим і менш продуктивним, тоді як .NET Core (до 5-ї версії) поступається .NET 8 за швидкодією та новими функціями. Тому .NET 8 обрано як найсучасніше та найефективніше рішення.

1.3.3. Система управління базами даних - SQLite

Для зберігання даних обрано SQLite – легку вбудовану базу даних. Простота інтеграції: SQLite не потребує окремого серверного процесу, працює як файл на диску, що ідеально для локальних додатків. Розмір бази даних мінімальний, що підходить для невеликих обсягів інформації (список авто та паркувальних місць). SQLite забезпечує високу швидкість роботи з простими запитами, що відповідає потребам системи. Відсутність ліцензійних витрат робить її економічно вигідною для малих автостоянок.

Розглядалися Microsoft SQL Server і MySQL. SQL Server є потужним рішенням для великих систем, але його складність і потреба в сервері роблять його надлишковим. MySQL вимагає налаштування сервера та мережевої інфраструктури, що суперечить вимозі автономності. SQLite обрано як оптимальний вибір для локального зберігання даних із мінімальними вимогами до адміністрування.

1.3.4. Додаткові інструменти

Для роботи з базою даних у .NET 8 використано Entity Framework Core (EF Core) – ORM-систему, яка спрощує взаємодію з SQLite через об’єктно-реляційне відображення. EF Core дозволяє працювати з даними як із об’єктами C#, що зменшує обсяг коду та підвищує його читабельність.

Обрані технології – Windows Forms, .NET 8 та SQLite – відповідають вимогам до простоти, автономності та ефективності системи обліку транспортних засобів на автостоянці. Windows Forms забезпечує зручний інтерфейс, .NET 8 – сучасну платформу з високою продуктивністю, а SQLite – легке та надійне зберігання даних. Цей набір технологій дозволяє створити компактний, функціональний і доступний продукт, який легко впровадити на невеликих автостоянках із мінімальними затратами. Наступні етапи розробки базуватимуться на цих рішеннях для реалізації поставлених завдань.

Висновки до розділу 1

Проведений аналіз у першому розділі дозволив сформувати чітке уявлення про задачі розробки системи обліку транспортних засобів на автостоянці та визначити оптимальний набір технологій для їх реалізації. Основна задача полягає в автоматизації процесів реєстрації заїзду та виїзду транспортних засобів, управління паркувальними місцями та забезпечення зручного інтерфейсу для адміністратора. Система орієнтована на локальне використання на невеликих автостоянках, що зумовлює вимоги до простоти впровадження, автономності та економічності.

Огляд аналогічних систем показав, що комерційні рішення, хоча й пропонують широкий функціонал, є надто дорогими та складними для малих об’єктів. Системи з відкритим кодом потребують значних зусиль для адаптації, тоді як локальні настільні додатки найкраще відповідають поставленим вимогам завдяки низькій вартості та простоті використання. Цей аналіз підтвердив доцільність розробки власного локального додатка, адаптованого до потреб невеликої автостоянки.

Вибір технологій – Windows Forms, .NET 8 та SQLite – обґрунтовано їхньою відповідністю технічним і функціональним вимогам. Windows Forms забезпечує швидке створення зрозумілого інтерфейсу, .NET 8 гарантує високу продуктивність і перспективність завдяки сучасним можливостям платформи, а SQLite ідеально підходить для локального зберігання даних завдяки компактності та простоті інтеграції. Використання Entity Framework Core додатково спрощує роботу з базою даних, підвищуючи ефективність розробки.

Перший розділ заклав теоретичну основу для подальшого проектування та реалізації системи. Обрані технології та сформульовані задачі створюють міцну базу для розробки програмного продукту, який буде ефективним, зручним і доступним для використання на автостоянках малого та середнього розміру. Наступні етапи роботи будуть спрямовані на проектування структури системи та її програмну реалізацію.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Проектування системи обліку транспортних засобів на автостоянці потребує ретельного вибору алгоритмів для реалізації основних функцій, таких як додавання, редагування, видалення записів, а також управління зв'язками між транспортними засобами та паркувальними місцями. У цьому підрозділі розглядаються обрані алгоритми, їхня логіка, обґрунтування вибору та оцінка ефективності з точки зору швидкості виконання та використання ресурсів.

Алгоритм додавання нового транспортного засобу включає введення даних (номерний знак, тип авто), вибір вільного паркувального місця та збереження запису в базі даних.

Алгоритм:

- Отримати список вільних паркувальних місць (IsOccupied = false) з таблиці ParkingSpots.
- Перевірити наявність вільних місць; якщо їх немає, видати повідомлення про помилку.
- Зібрати дані від користувача через графічний інтерфейс (номерний знак, тип авто, вибір місця).
- Створити новий об'єкт Vehicle із поточним часом прибуття та прив'язкою до вибраного місця.
- Оновити статус вибраного місця на IsOccupied = true.
- Зберегти зміни в базі даних.

Ефективність:

- Часова складність: O(n) для пошуку вільних місць, де nnn – кількість паркувальних місць. У реальних умовах nnn невелике (десятки чи сотні), тому операція виконується швидко.
- Використання пам'яті: мінімальне, оскільки завантажується лише список вільних місць.

- Обґрунтування: Використання LINQ-запиту через Entity Framework Core (Where) оптимізує пошук, а транзакційне збереження змін (SaveChanges) гарантує цілісність даних.

Алгоритм виїзду транспортного засобу фіксує виїзд авто, оновлюючи час виїзду та звільняючи місце.

Алгоритм:

1. Отримати вибраний запис Vehicle із DataGridView.
2. Перевірити, чи авто ще не виїхало (DepartureTime == null).
3. Оновити DepartureTime поточним часом.
4. Знайти пов’язане місце за ParkingSpotId і встановити IsOccupied = false.
5. Зберегти зміни в базі.

Ефективність:

- Часова складність: O(1)O(1)O(1), оскільки пошук за ідентифікатором (FirstOrDefault) у SQLite має константну швидкість завдяки індексації первинного ключа.
- Використання пам’яті: мінімальне, бо оперуємо одним записом.
- Обґрунтування: Прямий доступ до записів за ідентифікатором є найшвидшим способом, а оновлення двох сутностей (авто і місце) в одній транзакції забезпечує атомарність операції.

Алгоритм видалення запису видаляє запис про транспортний засіб і звільняє місце, якщо воно прив’язане.

Алгоритм:

1. Отримати вибраний запис Vehicle.
2. Знайти його в базі за Id.
3. Якщо є ParkingSpotId, оновити статус місця на IsOccupied = false.
4. Видалити запис із таблиці Vehicles.
5. Зберегти зміни.

Ефективність:

- Часова складність: O(1) для пошуку та видалення за ідентифікатором.
- Використання пам’яті: мінімальне, оскільки обробляється один об’єкт.
- Обґрунтування: Видалення за первинним ключем є швидким у SQLite, а перевірка наявності місця додає лише константний час.

Алгоритм відображення даних у таблиці завантажує дані з бази та відображає їх у DataGridView.

Алгоритм:

1. Завантажити всі записи Vehicles із пов’язаними ParkingSpot через Include.
2. Налаштувати колонки DataGridView із прив’язкою до властивостей об’єктів.
3. Використати CellFormatting для відображення SpotNumber із ParkingSpot.

Ефективність:

- Часова складність: O(n) для завантаження n записів і O(n·m) для форматування, де m – кількість колонок (у даному випадку константа).
- Використання пам’яті: пропорційне кількості записів (O(n)), що прийнятно для невеликих стоянок.
- Обґрунтування: Використання Include оптимізує завантаження пов’язаних даних одним запитом, а CellFormatting дозволяє гнучко відображати навігаційні властивості.

2.1.5. Порівняння з альтернативами

- **Прямі SQL-запити**: Можуть бути швидшими для великих обсягів даних (O(logn) з індексами), але ускладнюють код і знижують переносимість. EF Core обрано за простоту та читабельність.
- **Кешування**: Для малих обсягів даних (до 100-200 записів) кешування не дає значного приросту швидкості, тому не використовується.
- **Асинхронність**: Асинхронні методи (async/await) могли б покращити чуйність інтерфейсу, але для локальної бази затримки мінімальні, тому синхронний підхід достатній.

Обрані алгоритми є простими, ефективними та відповідають масштабам задачі. Їхня часова складність (O(1) для точкових операцій і O(n) для перегляду) забезпечує швидкодію при невеликій кількості записів, а використання Entity Framework Core оптимізує доступ до даних і гарантує цілісність. Ці рішення створюють надійну основу для реалізації системи, враховуючи її локальний характер і обмежені ресурси цільового середовища.

2.2. Спроєктовані класи програми

У процесі проектування системи обліку транспортних засобів на автостоянці було розроблено набір класів, які відображають основні сутності, їхні взаємозв'язки та логіку роботи з даними. У цьому підрозділі описано спроєктовані класи, їхні властивості, методи та роль у системі. Всі класи реалізовано з використанням C# на платформі .NET 8, з урахуванням принципів об'єктно-орієнтованого програмування.

2.2.1. Клас Vehicle (Транспортний засіб).

Представляє сутність транспортного засобу, що перебуває на автостоянці. Клас моделює транспортний засіб із прив'язкою до паркувального місця через ParkingSpotId. Навігаційна властивість ParkingSpot дозволяє отримувати дані про місце без додаткових запитів.

```
// Models/Vehicle.cs
12 references
public class Vehicle
{
    8 references
    public int Id { get; set; }                    // Унікальний ідентифікатор транспортного засобу
    8 references
    public string? LicensePlate { get; set; }      // Номерний знак
    4 references
    public string? VehicleType { get; set; }       // Тип авто
    4 references
    public DateTime ArrivalTime { get; set; }      // Час прибуття
    6 references
    public DateTime? DepartureTime { get; set; }   // Час виїзду
    8 references
    public string? PhotoPath { get; set; }         // Шлях до фото
    7 references
    public int ParkingSpotId { get; set; }         // Зовнішній ключ до паркувального місця
    3 references
    public ParkingSpot? ParkingSpot { get; set; }  // Навігаційна властивість до ParkingSpot
}
```

Рисунок 2.2.1 – Клас Vehicle

Властивості:

- int Id – унікальний ідентифікатор транспортного засобу (первинний ключ).
- string? LicensePlate – номерний знак авто (nullable, для гнучкості).
- string? VehicleType – тип транспортного засобу (наприклад, "Легковий").
- DateTime ArrivalTime – час прибуття на стоянку.
- DateTime? DepartureTime – час виїзду зі стоянки (nullable, якщо авто ще не виїхало).
- string? PhotoPath – шлях до фотографії авто (nullable, якщо фото відсутнє).
- int ParkingSpotId – зовнішній ключ до паркувального місця.
- ParkingSpot? ParkingSpot – навігаційна властивість до об’єкта ParkingSpot.

2.2.2. Клас ParkingSpot (Паркувальне місце)

Описує паркувальне місце на стоянці. Клас представляє фізичне місце на автостоянці. Відсутність навігаційної властивості до Vehicle у цьому класі обумовлена одностороннім зв’язком (одне місце може бути пов’язане з одним авто, але авто завжди має місце).

```
9 references
public class ParkingSpot
{
    // Унікальний ідентифікатор паркувального місця в базі даних
    9 references
    public int Id { get; set; }

    // Номер паркувального місця на стоянці (наприклад, 1, 2, 3 тощо)
    10 references
    public int SpotNumber { get; set; }

    // Статус зайнятості паркувального місця (true - зайняте, false - вільне)
    9 references
    public bool IsOccupied { get; set; }
}
```

Рисунок 2.2.2 – Клас ParkingSpot

Властивості:

- int Id – унікальний ідентифікатор місця (первинний ключ).
- int SpotNumber – номер місця на стоянці.
- bool IsOccupied – статус зайнятості (true – зайняте, false – вільне).

2.2.3. Клас ParkingContext (Контекст бази даних)

Забезпечує доступ до бази даних через Entity Framework Core. Клас є центральним елементом для роботи з базою даних. Він налаштовує зв’язок один-до-багатьох між ParkingSpot і Vehicle та використовує SQLite як сховище даних.

```
12 references
public class ParkingContext : DbContext
{
    private readonly string _dbPath;
    10 references
    public ParkingContext(string dbPath)...
    10 references
    public DbSet<Vehicle> Vehicles { get; set; }
    16 references
    public DbSet<ParkingSpot> ParkingSpots { get; set; }
    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)...
    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)...
}
```

Рисунок 2.2.3 – Клас ParkingContext

Властивості:

- DbSet<Vehicle> Vehicles – набір записів транспортних засобів.
- DbSet<ParkingSpot> ParkingSpots – набір записів паркувальних місць.

Методи:

- ParkingContext(string dbPath) – конструктор, що приймає шлях до бази SQLite.
- OnConfiguring(DbContextOptionsBuilder optionsBuilder) – налаштовує підключення до SQLite.
- OnModelCreating(ModelBuilder modelBuilder) – визначає зв'язок між Vehicle і ParkingSpot.

2.2.4. Клас MainForm (Головна форма)

Реалізує основний інтерфейс користувача для роботи з транспортними засобами. Клас є основним елементом UI, що об'єднує логіку роботи з транспортними засобами. Використовує DataGridView для відображення даних і PictureBox для показу фото.

```
3 references
partial class MainForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private DataGridView dataGridView1;
    private PictureBox pictureBox1;
    private Button btnSelectPhoto;
    private Button btnAddVehicle;
    private Button btnDepartVehicle;
    private Button button1;
    private TableLayoutPanel tableLayoutPanel1;
    private Button btnDeleteCar;
}
```

Рисунок 2.2.4 – Клас MainForm

Властивості:

- string _dbPath – шлях до файлу бази даних.
- string _photosPath – шлях до папки з фотографіями.

Методи:

- LoadVehicles() – завантажує список транспортних засобів у DataGridView.
- AddVehicleOnArrival() – додає новий транспортний засіб із вибором вільного місця.
- DepartVehicle() – фіксує виїзд авто та звільняє місце.
- DeleteVehicle() – видаляє запис про авто.
- SelectVehiclePhoto() – дозволяє вибрати фото для авто.
- DataGridView1_SelectionChanged(object sender, EventArgs e) – обробляє вибір рядка для відображення фото.
- CreateNoPhotoImage() – створює зображення "No photo".

2.2.5. Клас ParkingSpotForm (Форма управління місцями)

Забезпечує інтерфейс для роботи з паркувальними місцями. Клас відповідає за управління паркувальними місцями через окрему форму. Діалог ShowEditDialog забезпечує зручне введення даних.

```
3 references
partial class ParkingSpotForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private DataGridView dataGridViewSpots;
    private Button btnAdd;
    private Button btnEdit;
    private Button btnDelete;
    private TableLayoutPanel tableLayoutPanel1;
}
```

Рисунок 2.2.5 – Клас ParkingSpotForm

Властивості:

- string _dbPath – шлях до бази даних.

Методи:

- LoadParkingSpots() – завантажує список місць у DataGridView.
- btnAdd_Click(object sender, EventArgs e) – додає нове місце.
- btnEdit_Click(object sender, EventArgs e) – редагує вибране місце.
- btnDelete_Click(object sender, EventArgs e) – видаляє місце.
- ShowEditDialog(ParkingSpot spot, string title) – відображає діалог для введення/редагування даних.

2.2.6. Структура взаємозв'язків

- Vehicle пов'язаний із ParkingSpot через ParkingSpotId (один-до-багатьох).
- ParkingContext виступає посередником між моделями та базою SQLite.
- MainForm і ParkingSpotForm взаємодіють із ParkingContext для доступу до даних і відображення їх у UI.

Спроєктовані класи чітко відображають предметну область системи: транспортні засоби та паркувальні місця. Клас ParkingContext забезпечує ефективну роботу з базою даних, а форми MainForm і ParkingSpotForm реалізують користувацький інтерфейс із необхідним функціоналом. Використання навігаційних властивостей і ORM спрощує доступ до пов'язаних даних, а модульна структура дозволяє легко розширювати систему в майбутньому.

2.3. База даних

Розробка системи обліку транспортних засобів на автостоянці передбачає створення бази даних для зберігання інформації про транспортні засоби та паркувальні місця. У цьому підрозділі описано структуру бази даних, її таблиці, зв'язки між ними та обґрунтування вибору підходу до організації даних.

2.3.1. Вибір СУБД

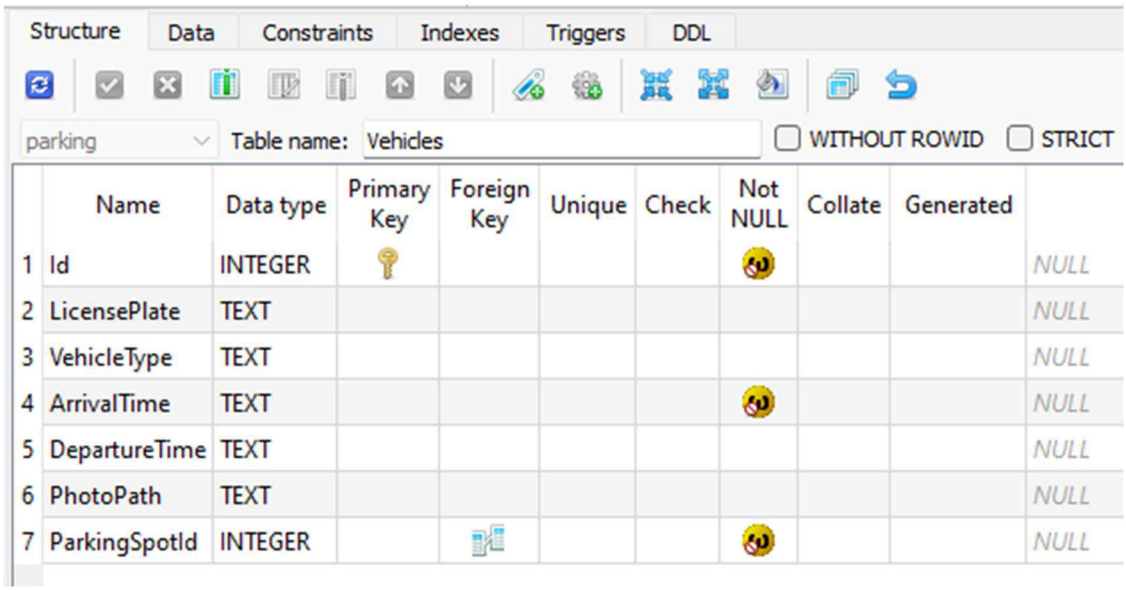
Для зберігання даних обрано SQLite – легку вбудовану систему управління базами даних (СУБД). Вибір обґрунтовано наступними факторами:

- SQLite працює як єдиний файл на диску, що ідеально для автономного настільного додатка без потреби в сервері.
- Не вимагає складного адміністрування чи конфігурації, що відповідає масштабам невеликої автостоянки.
- Забезпечує швидкий доступ до даних при малому обсязі записів (до кількох сотень), що є типовим для цільового сценарію.

2.3.2. Структура бази даних

База даних складається з двох основних таблиць: Vehicles (Транспортні засоби) та ParkingSpots (Паркувальні місця). Вони пов’язані через зовнішній ключ для відображення зв’язку між авто та місцем, яке воно займає.

Таблиця Vehicles зберігає інформацію про транспортні засоби, що перебувають або перебували на стоянці. Кожний запис у таблиці відповідає одному транспортному засобу. Поле ParkingSpotId вказує на місце, яке займає авто, забезпечуючи зв’язок із таблицею ParkingSpots.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	Id	INTEGER								NULL
2	LicensePlate	TEXT								NULL
3	VehicleType	TEXT								NULL
4	ArrivalTime	TEXT								NULL
5	DepartureTime	TEXT								NULL
6	PhotoPath	TEXT								NULL
7	ParkingSpotId	INTEGER								NULL

Рисунок 2.3.1 – Таблиця Vehicles

Поля:

- Id (INTEGER, PRIMARY KEY, AUTOINCREMENT) – унікальний ідентифікатор транспортного засобу.
- LicensePlate (TEXT, NULLABLE) – номерний знак авто.
- VehicleType (TEXT, NULLABLE) – тип транспортного засобу (наприклад, "Легковий").
- ArrivalTime (DATETIME, NOT NULL) – дата та час прибуття на стоянку.
- DepartureTime (DATETIME, NULLABLE) – дата та час виїзду (NULL, якщо авто ще на стоянці).
- PhotoPath (TEXT, NULLABLE) – шлях до фотографії авто.

- ParkingSpotId (INTEGER, NOT NULL, FOREIGN KEY) – зовнішній ключ, що посилається на таблицю ParkingSpots.

Таблиця ParkingSpots зберігає інформацію про паркувальні місця на стоянці. Таблиця відображає стан паркувальних місць. Поле SpotNumber є унікальним номером для фізичного позначення місця, а IsOccupied показує, чи зайняте місце авто.

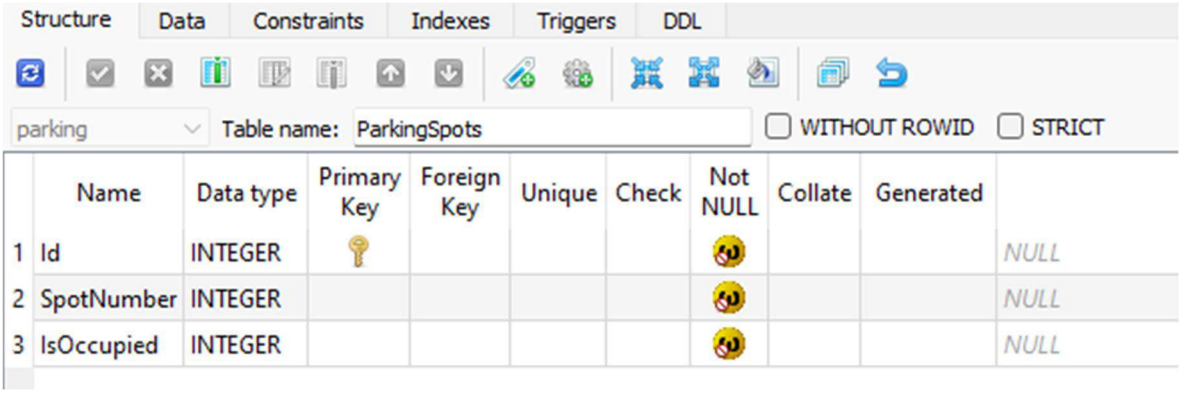


Рисунок 2.3.2 – Таблиця ParkingSpots

Поля:

- Id (INTEGER, PRIMARY KEY, AUTOINCREMENT) – унікальний ідентифікатор місця.
- SpotNumber (INTEGER, NOT NULL) – номер місця на стоянці.
- IsOccupied (BOOLEAN, NOT NULL) – статус зайнятості (1 – зайняте, 0 – вільне).

2.3.3. Схема бази даних

Схематично структуру представлено на рисунку 2.3.3.

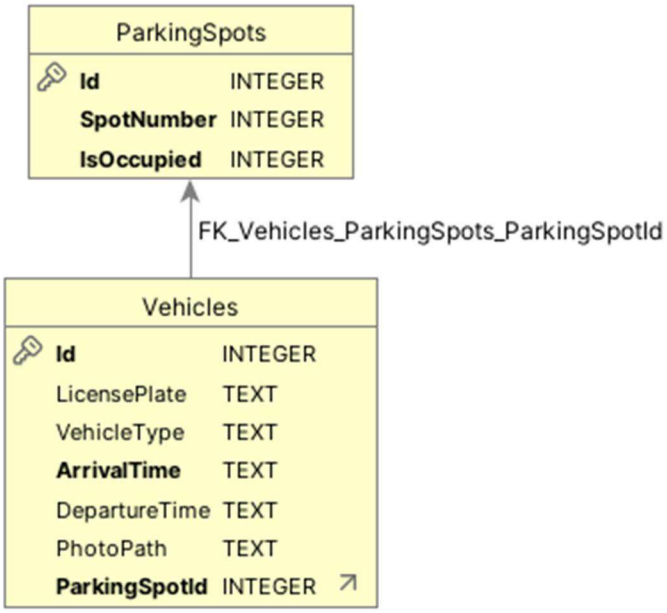


Рисунок 2.3.3 – Схема бази даних

Таблиця ParkingSpotId та таблиця Vehicles пов’язані зв’язком один-до-багатьох (одне паркувальне місце може бути пов’язане з одним транспортним засобом у певний момент часу, але транспортний засіб завжди прив’язаний до одного місця). Реалізовано через зовнішній ключ ParkingSpotId у таблиці Vehicles, який посилається на поле Id у таблиці ParkingSpots. Налаштовано поведінку ON DELETE RESTRICT, щоб запобігти видаленню місця, якщо воно прив’язане до авто, забезпечуючи цілісність даних.

Спроєктована база даних на основі SQLite є компактною, ефективною та повністю відповідає функціональним вимогам системи. Таблиці Vehicles і ParkingSpots з їхніми зв’язками забезпечують облік транспортних засобів і стану паркувальних місць. Використання Entity Framework Core для роботи з базою спрощує доступ до даних і гарантує їхню цілісність. Ця структура є основою для реалізації всіх операцій системи, описаних у попередніх підрозділах.

2.4. Програмна реалізація

У цьому підрозділі описано процес програмної реалізації системи обліку транспортних засобів на автостоянці, включаючи структуру проєкту, ключові фрагменти коду та особливості впровадження функціоналу. Реалізація виконана на основі обраних технологій – Windows Forms, .NET 8 та SQLite – із використанням Entity Framework Core для роботи з базою даних.

2.4.1. Структура проєкту

Проєкт організовано в кількох папках для логічного поділу коду:

- **Models**: Містить класи Vehicle і ParkingSpot, які відображають сутності бази даних.
- **Data**: Включає клас ParkingContext для роботи з SQLite.
- **Forms**: Містить форми MainForm (головна форма для роботи з авто) та ParkingSpotForm (форма управління місцями).

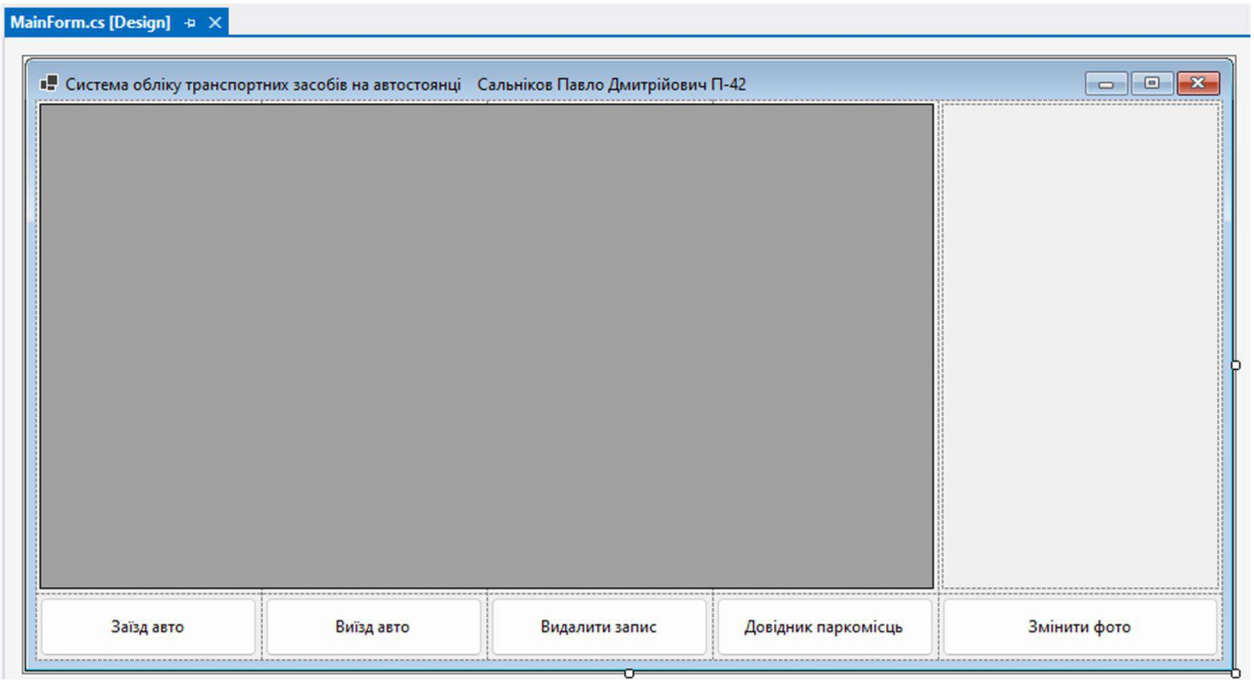


Рисунок 2.4.1 – Головна форма для роботи з авто

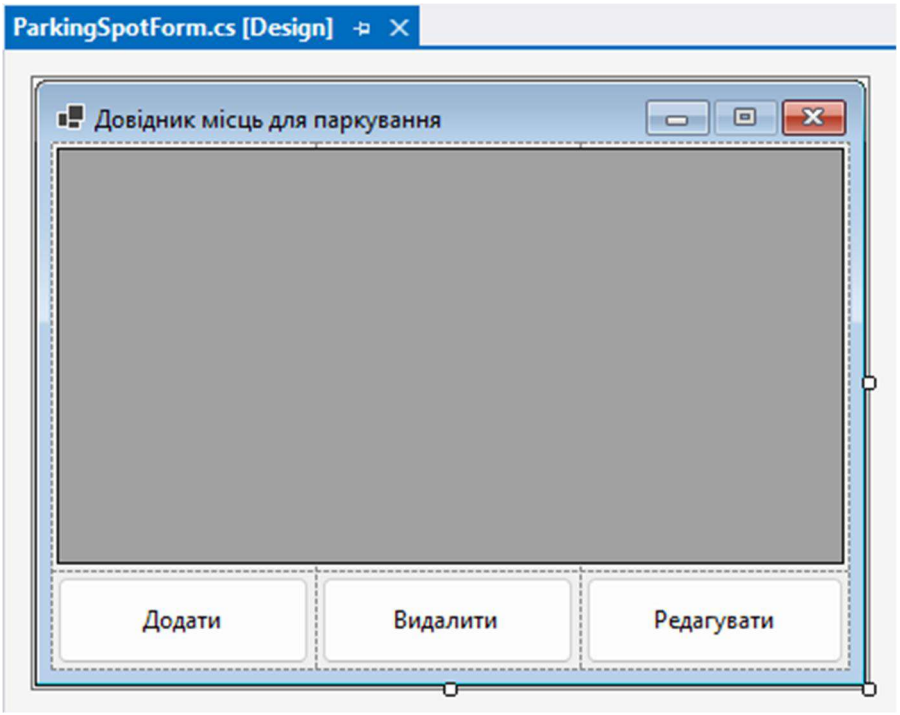


Рисунок 2.4.2 – Форма управління місцями

- **Program.cs**: Точка входу в програму, де ініціалізується база даних і запускається головна форма.

Файл бази даних (parking.db) зберігається в корені виконуваного каталогу, а фотографії – у папці Photos.

2.4.2. Реалізація основного функціоналу в MainForm

Метод LoadVehicles завантажує дані і відображає список авто з номерами місць.

```
5 references
private void LoadVehicles()
{
    using (var db = new ParkingContext(_dbPath))
    {
        // Завантажуємо Vehicles разом із пов'язаними ParkingSpot
        var vehicles = db.Vehicles
            .Include(v => v.ParkingSpot)
            .ToList();
        // Очищаємо попередні обробники події CellFormatting, щоб уникнути дублювання
        dataGridView1.CellFormatting -= DataGridView1_CellFormatting; // Видаляємо, якщо вже є
        dataGridView1.AutoGenerateColumns = false;
        dataGridView1.Columns.Clear();
        // Додаємо колонки з українськими підписами
        dataGridView1.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridView1.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridView1.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridView1.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridView1.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridView1.Columns.Add(new DataGridViewTextBoxColumn...);
        // Прив'язуємо дані
        dataGridView1.DataSource = vehicles;
        // Додаємо обробник CellFormatting після створення колонок
        dataGridView1.CellFormatting += DataGridView1_CellFormatting;
        SettingGrid(dataGridView1);

    }
}
```

Рисунок 2.4.3 – Метод LoadVehicles

Метод btnAddVehicle_Click дозволяє додати авто на парковку та вибрати вільне місце:

```
1 reference
private void btnAddVehicle_Click(object sender, EventArgs e)
{
    using (var db = new ParkingContext(_dbPath))
    {
        var freeSpots = db.ParkingSpots.Where(s => !s.IsOccupied).ToList();
        if (freeSpots.Count == 0)...

        // Змінна для зберігання шляху до фото
        string? photoPath = null;

        using (Form addForm = new Form
        {
            Text = "Додавання нового авто",
            Size = new Size(350, 280),  // Збільшено ширину і висоту для нових елементів
            StartPosition = FormStartPosition.CenterParent,
            FormBorderStyle = FormBorderStyle.FixedDialog,
            MaximizeBox = false,
            MinimizeBox = false
        }
        )...
    }
}
```

Рисунок 2.4.4 – Метод btnAddVehicle_Click

Метод btnDepartVehicle_Click фіксує виїзд авто з парковки.

```
1 reference
private void btnDepartVehicle_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count == 0)...

    var selectedRow = dataGridView1.SelectedRows[0];
    var vehicle = selectedRow.DataBoundItem as Vehicle;

    if (vehicle == null)...

    if (vehicle.DepartureTime.HasValue)...

    var result = MessageBox.Show($"Підтвердити виїзд авто {vehicle.LicensePlate}?",
        "Підтвердження виїзду", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (result == DialogResult.Yes)...
}
```

Рисунок 2.4.5 – Метод btnDepartVehicle_Click

2.4.3. Реалізація управління місцями в ParkingSpotForm

Метод LoadParkingSpots відображає список місць.

```
4 references
private void LoadParkingSpots()
{
    using (var db = new ParkingContext(_dbPath))
    {
        var spots = db.ParkingSpots.ToList();
        dataGridViewSpots.AutoGenerateColumns = false;
        dataGridViewSpots.Columns.Clear();

        // Додаємо колонки з українськими підписами
        dataGridViewSpots.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridViewSpots.Columns.Add(new DataGridViewTextBoxColumn...);
        dataGridViewSpots.Columns.Add(new DataGridViewCheckBoxColumn...);
        dataGridViewSpots.DataSource = spots;
    }
}
```

Рисунок 2.4.6 – Метод LoadParkingSpots

2.4.4. Особливості реалізації

Фото авто копіюються в папку Photos із унікальним ім’ям (LicensePlate.jpg), а шлях зберігається в PhotoPath.

Використання транзакцій у SaveChanges() забезпечує атомарність операцій (наприклад, додавання авто та оновлення місця).

DataGridView із подією CellFormatting дозволяє гнучко відображати пов’язані дані (SpotNumber).

Програмна реалізація охоплює весь необхідний функціонал: облік заїзду/виїзду, управління місцями, роботу з фото. Код структурований, модульний і готовий до тестування. Використання EF Core спрощує доступ до даних, а Windows Forms забезпечує зручний інтерфейс.

Висновки до розділу 2

Другий розділ дипломної роботи був присвячений проектуванню та розробці системи обліку транспортних засобів на автостоянці. У ході виконання цього етапу сформовано чітку структуру системи, обрано алгоритми, спроєктовано класи та базу даних, а також реалізовано програмний продукт із повним набором необхідного функціоналу.

Аналіз і вибір алгоритмів показали, що для основних операцій (додавання, виїзду, видалення та відображення даних) достатньо простих і ефективних рішень із часовою складністю O(1)O(1)O(1) для точкових операцій і O(n)O(n)O(n) для завантаження списків. Використання Entity Framework Core оптимізувало доступ до бази даних, забезпечивши швидкодію та цілісність даних при невеликому обсязі записів, характерному для малих автостоянок.

Спроєктовані класи (Vehicle, ParkingSpot, ParkingContext, MainForm, ParkingSpotForm) відображають предметну область і забезпечують модульність системи. Клас Vehicle пов’язаний із ParkingSpot через зовнішній ключ, що дозволяє ефективно відстежувати зайнятість місць. Класи форм реалізують зручний інтерфейс користувача з підтримкою всіх необхідних операцій, таких як додавання авто, фіксація виїзду, вибір фото та управління паркувальними місцями.

База даних на основі SQLite із двома таблицями (Vehicles і ParkingSpots) є компактною та функціональною. Зв’язок один-до-багатьох між таблицями, реалізований через ParkingSpotId, забезпечує логічну структуру для зберігання

даних про авто та їхні місця. Використання легкої СУБД SQLite відповідає вимогам автономності та простоти впровадження.

Програмна реалізація охопила всі аспекти системи: від ініціалізації бази з початковими даними до відображення інформації в DataGridView і обробки користувацьких дій. Код структуровано в модулі, що полегшує його підтримку та розширення. Особливу увагу приділено обробці фото та забезпеченню цілісності даних при операціях із базою.

РОЗДІЛ 2 завершив етап розробки, створивши готовий програмний продукт, який відповідає поставленим завданням. Система готова до тестування, оцінки її ефективності та подальшого вдосконалення, що буде розглянуто в наступних розділах роботи. Отримані результати підтверджують правильність вибору технологій і підходів до проектування, закладених у першому розділі.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для коректної роботи системи обліку транспортних засобів на автостоянці необхідно, щоб комп’ютер відповідав певним технічним характеристикам, які забезпечують стабільне функціонування програми. Операційна система має бути Windows 10 або новіша версія (наприклад, Windows 11), оскільки додаток розроблено з використанням Windows Forms і .NET 8, які повністю сумісні з цими платформами. Старіші версії, такі як Windows 7, можуть потребувати додаткового встановлення бібліотек, що не входить до стандартного пакета підтримки.

Щодо апаратного забезпечення, достатньо процесора з частотою від 1 ГГц, бажано двоядерного, для забезпечення базової продуктивності при обробці даних і відображенні інтерфейсу. Оперативна пам’ять обсягом 2 ГБ є мінімальною для комфортної роботи, хоча для оптимальної швидкості рекомендується 4 ГБ або більше, особливо якщо одночасно запущено інші програми. Вільний простір на диску має становити щонайменше 100 МБ для розміщення виконуваного файлу, бази даних SQLite (parking.db) і папки з фотографіями (Photos), хоча обсяг може зростати залежно від кількості збережених зображень.

Програма не потребує підключення до Інтернету, оскільки працює локально, а база даних зберігається на комп’ютері користувача. Для роботи з базою даних SQLite не потрібне додаткове програмне забезпечення, адже вона вбудована в додаток через .NET 8. Однак для запуску програми необхідно, щоб на комп’ютері було встановлено .NET 8 Runtime, який можна завантажити з офіційного сайту Microsoft, якщо його ще немає в системі. Роздільність екрана рекомендовано не нижче 1024x768 пікселів, щоб усі елементи інтерфейсу (таблиці, кнопки, діалогові вікна) відображалися коректно без обрізання.

3.2. Інтерфейс користувача

Інтерфейс користувача системи обліку транспортних засобів на автостоянці розроблено з використанням Windows Forms і є інтуїтивно зрозумілим для адміністраторів, які не мають глибоких технічних знань. Він складається з двох основних форм: головної форми для роботи з транспортними засобами та додаткової форми для управління паркувальними місцями, кожна з яких має чітко структуровані елементи для зручного виконання операцій.

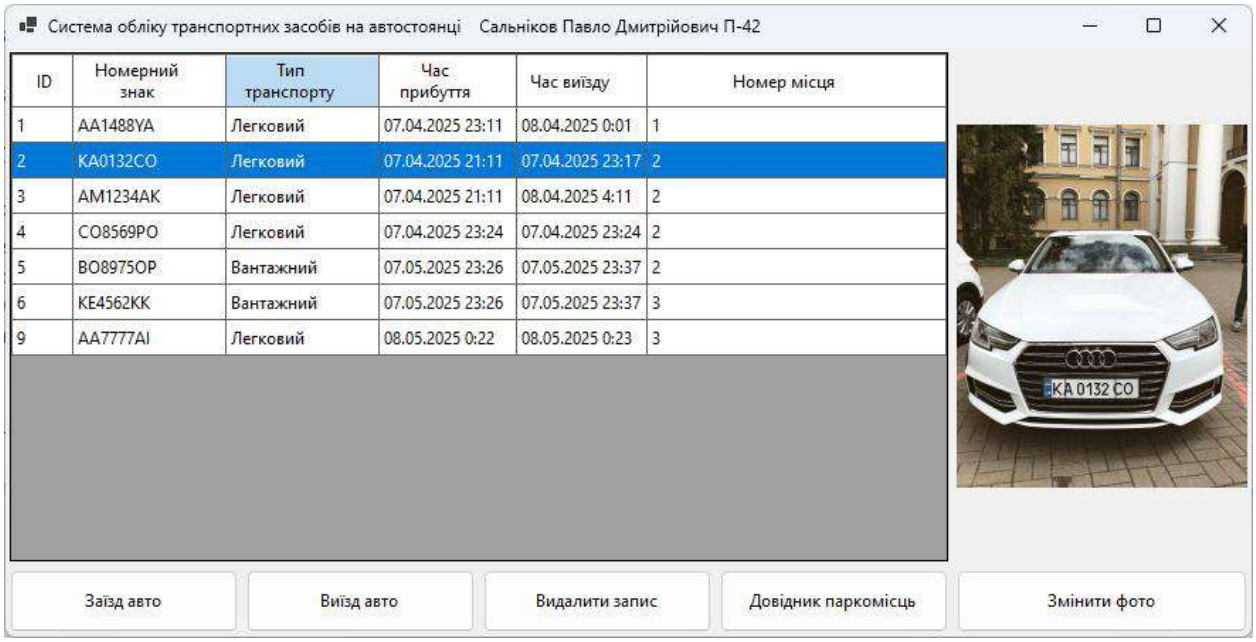


Рисунок 3.2.1 – Інтерфейс програми

Головна форма відкривається при запуску програми і є центральним елементом інтерфейсу. У її верхній частині розміщено таблицю (DataGridView), яка відображає список транспортних засобів. Таблиця містить колонки з інформацією про ідентифікатор, номерний знак, тип авто, час прибуття, час виїзду та номер паркувального місця. Користувач може виділити рядок, клікнувши на нього, після чого праворуч у спеціальному полі (PictureBox) з’являється фотографія вибраного авто або зображення "No photo", якщо фото відсутнє. Під таблицею розташовано кілька кнопок: "Заїзд авто" для додавання нового транспортного засобу, "Виїзд авто" для фіксації

виїзду, "Видалити" для видалення запису та "Вибрати фото" для додавання зображення до вибраного авто

При натисканні "Заїзд авто" відкривається діалогове вікно, де користувач вводить номерний знак у текстове поле, обирає тип транспортного засобу зі спадного списку (наприклад, "Легковий", "Вантажний", "Мотоцикл") і вибирає вільне паркувальне місце з іншого спадного списку, який містить номери доступних місць. Кнопки "OK" і "Скасувати" дозволяють підтвердити або скасувати введення. Аналогічно, при виборі фото відкривається стандартний діалог Windows для вибору файлу зображення, після чого фото копіюється до папки програми та відображається у PictureBox.

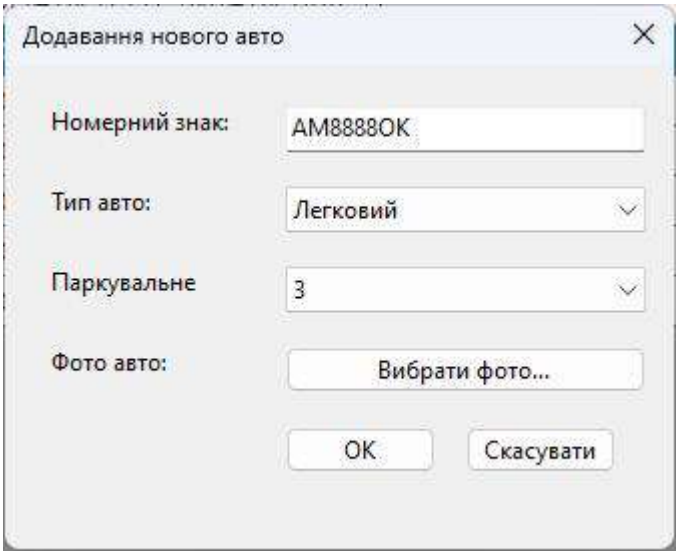


Рисунок 3.2.2 – Заїзд авто

Додаткова форма "Довідник паркомісць" відкривається через відповідну кнопку на головній формі та містить власну таблицю (DataGridView) зі списком паркувальних місць. Таблиця показує ідентифікатор, номер місця та статус зайнятості у вигляді прапорця (включений – зайняте, виключений – вільне). Під таблицею розміщені кнопки "Додати", "Редагувати" та "Видалити". Натиснення "Додати" або "Редагувати" викликає діалогове вікно з полем для введення номера місця та прапорцем для статусу зайнятості, а також кнопками "OK" і "Скасувати". Кнопка "Видалити" після вибору рядка запитує підтвердження перед видаленням місця.

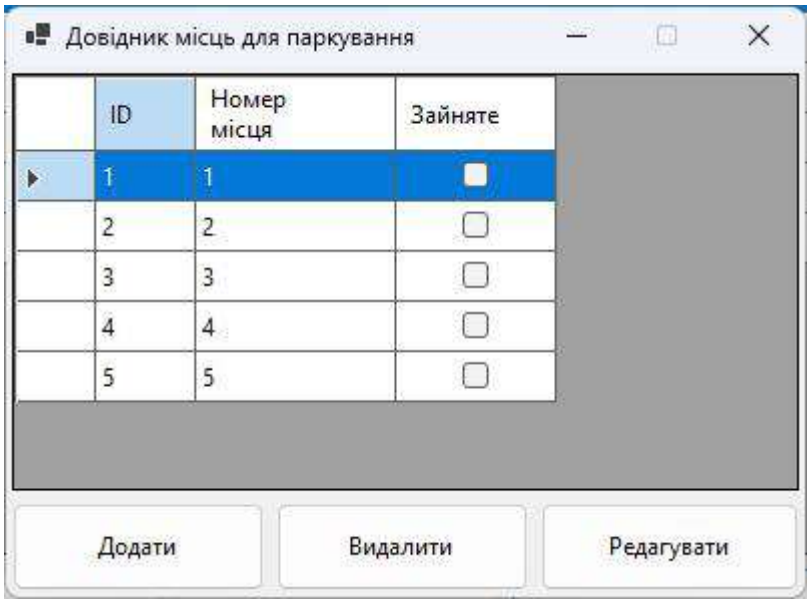


Рисунок 3.2.3 – Довідник місць для паркування

Інтерфейс спроєктовано так, щоб усі дії були максимально простими: користувач бачить усю необхідну інформацію в таблицях, а операції виконуються через кнопки з зрозумілими назвами. Використання діалогових вікон із валідацією (наприклад, перевірка на порожній номерний знак) запобігає помилкам введення. Колонки таблиць мають фіксовану ширину для зручного перегляду, а часові поля форматуються у зрозумілому вигляді (наприклад, "08.04.2025 14:30"). Загалом, інтерфейс забезпечує легкий доступ до всіх функцій системи, роблячи її зручною для щоденного використання адміністратором автостоянки.

3.3. Інструкція для роботи з програмою

Інструкція для роботи з програмою "Система обліку транспортних засобів на автостоянці" призначена для адміністраторів і описує основні дії для виконання повсякденних операцій. Перед початком роботи переконайтеся, що програма встановлена на комп’ютері з Windows 10 або новішою версією, а .NET 8 Runtime присутній у системі.

Після запуску програми відкривається головна форма з таблицею, що відображає список транспортних засобів. У таблиці видно ідентифікатор, номерний знак, тип авто, час прибуття, час виїзду та номер паркувального

місця. Щоб переглянути деталі, клікніть на рядок – праворуч з’явиться фото авто або напис "No photo", якщо зображення відсутнє.

Для додавання нового транспортного засобу при заїзді натисніть кнопку "Заїзд авто". У діалоговому вікні введіть номерний знак у текстове поле, виберіть тип авто зі спадного списку (наприклад, "Легковий") і оберіть вільне місце з іншого списку, де відображаються номери доступних місць. Натисніть "ОК" для збереження або "Скасувати" для відміни. Якщо вільних місць немає, з’явиться повідомлення про це.

Щоб зафіксувати виїзд авто, виділіть його рядок у таблиці та натисніть "Виїзд авто". Програма запропонує підтвердити дію, вказавши номерний знак. Натисніть "Так" для завершення – час виїзду оновиться, а місце стане вільним. Якщо авто вже виїхало, з’явиться відповідне попередження.

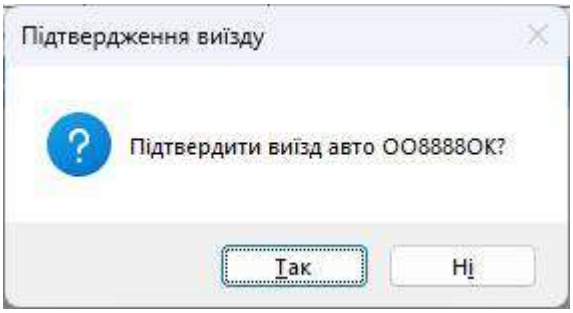


Рисунок 3.3.1 – Підтвердження виїзду

Для видалення запису про авто виділіть рядок і натисніть "Видалити". З’явиться запит на підтвердження з номером авто. Натисніть "Так" для видалення – запис зникне з таблиці, а місце, якщо було зайняте, звільниться.

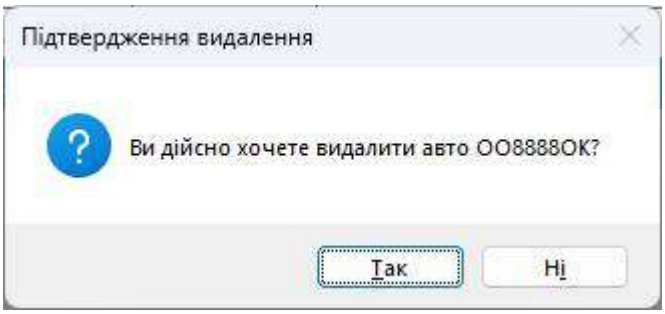


Рисунок 3.3.2 – Видалення запису про паркування

Щоб додати фото до авто, виділіть його рядок і натисніть "Вибрати фото". У вікні вибору файлу знайдіть зображення (формати .jpg, .png, .bmp), виберіть його та натисніть "Відкрити". Фото автоматично скопіюється до папки програми та відобразиться у полі праворуч.

Для управління паркувальними місцями натисніть "Довідник паркомісць", щоб відкрити додаткову форму. У таблиці видно ідентифікатор, номер місця та статус зайнятості. Щоб додати нове місце, натисніть "Додати", введіть номер місця у текстове поле діалогового вікна, позначте прапорець "Зайняте", якщо потрібно, і натисніть "ОК". Для редагування виділіть місце, натисніть "Редагувати", змініть дані у вікні та підтвердіть "ОК". Щоб видалити місце, виділіть його, натисніть "Видалити" і підтвердіть дію у запиті.

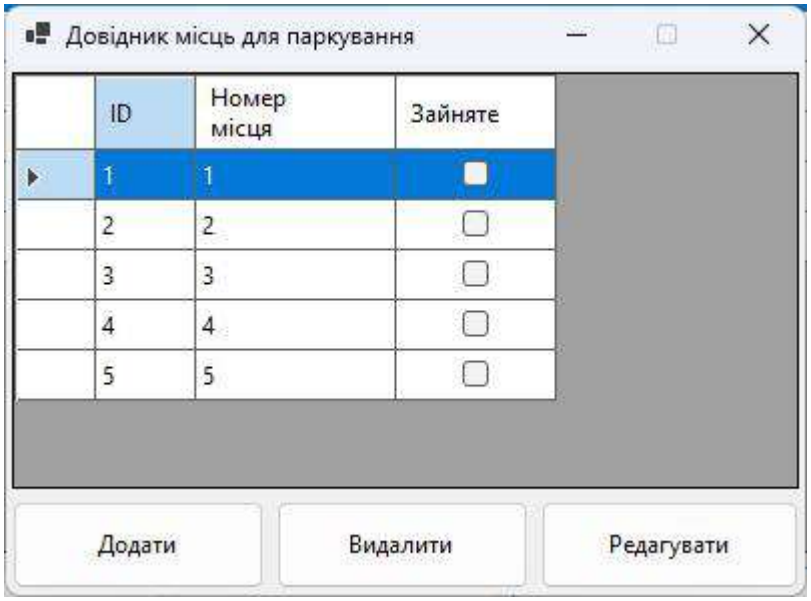


Рисунок 3.3.3 – Довідник місць для паркування

Програма автоматично оновлює таблиці після кожної операції, тому зміни відображаються одразу. У разі помилок, наприклад, спроби додати авто без номера чи видалити зайняте місце, з’являться повідомлення з поясненням. Для завершення роботи просто закрийте головну форму. Дані зберігаються у файлі parking.db у каталозі програми, а фото – у папці Photos.

Висновки до розділу 3

Третій розділ дипломної роботи присвячений керівництву користувача системою обліку транспортних засобів на автостоянці, де детально описано вимоги до системи, інтерфейс користувача та інструкцію для роботи з програмою. Цей етап завершує підготовку системи до практичного використання, надаючи адміністраторам необхідну інформацію для її ефективної експлуатації.

Мінімальні вимоги до системи сформульовано з урахуванням її локального характеру та орієнтації на невеликі автостоянки. Програма працює на стандартних офісних комп'ютерах із Windows 10 або новішою версією, потребує лише базових апаратних ресурсів і .NET 8 Runtime, що робить її доступною для впровадження без значних витрат. Відсутність потреби в Інтернеті чи складному обладнанні підкреслює простоту розгортання.

Інтерфейс користувача розроблено з акцентом на зручність і простоту. Головна форма забезпечує швидкий доступ до роботи з транспортними засобами через таблицю та кнопки для основних операцій, таких як заїзд, виїзд, видалення та додавання фото. Додаткова форма для управління паркувальними місцями дозволяє легко контролювати їхній стан. Використання діалогових вікон із валідацією даних мінімізує помилки, а чітке розташування елементів сприяє інтуїтивному освоєнню.

Інструкція для роботи з програмою охоплює всі ключові дії: від додавання авто при заїзді до управління паркувальними місцями. Вона описує кроки покроково, із зазначенням можливих помилок і способів їх уникнення, що робить систему зрозумілою навіть для користувачів із мінімальним досвідом роботи з комп'ютером. Автоматичне оновлення таблиць після операцій і збереження даних у локальній базі забезпечують надійність і зручність використання.

ВИСНОВКИ

Ось і дійшла до свого логічного завершення наша подорож у світ автоматизації автостоянок – дипломна робота, що стала не просто набором коду чи сторінок тексту, а справжнім мостом між ідеєю та реальністю. Система обліку транспортних засобів на автостоянці, народжена в уяві й втілена через Windows Forms, .NET 8 та SQLite, постала як надійний помічник, що здатен розплутати хаос паркувальних буднів і подарувати адміністраторам ясність і порядок.

Розпочавши з аналізу технологій, ми проклали шлях через густі ліси комерційних гігантів і хиткі стежки відкритих кодів, зупинившись на простій, але міцній основі локального додатка. Кожен алгоритм, кожен клас і кожен рядок бази даних були ретельно вирізьблені, немов деталі пазла, що разом склали цілісну картину – систему, яка вміє слухати потреби користувача й відповідати на них із точністю годинника. Головна форма стала вітриною, де транспортні засоби оживають у таблицях і фотографіях, а паркувальні місця – мов вірні стражі – стоять на варті порядку в окремій формі управління.

Ця робота – не просто технічний проєкт, а й творчий акт, де прагнення до зручності й ефективності переплітається з прагматичною простотою. Ми створили інструмент, який не потребує хмарних висот чи дорогих машинерій, а живе на скромному офісному комп’ютері, тихо й невпинно виконуючи свою справу. Інструкція для користувача стала її голосом, що лагідно підказує, як зробити кожен заїзд і виїзд частиною гармонійного процесу.

Та це не кінець, а лише початок. Система, що дихає в ритмі автостоянки, відкриває двері до нових можливостей – від звітів про завантаженість до інтеграції з камерами чи платіжними модулями. Вона готова до випробувань реальним життям, до шумних ранків і тихих вечорів, коли адміністратор, спираючись на її підтримку, зможе зітхнути з полегшенням, знаючи, що все під контролем. Ця дипломна робота – не просто звіт, а запрошення до світу, де технології стають невидимими нитками, що тримають порядок у наших буднях.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Глинський Я. М. Основи програмування на C#: навч. посібник. Львів: СПОЛОМ, 2020. 320 с.
2. Троелсен Е., Джепікс Е. Мова програмування C# 10 та платформа .NET 8. Київ: Діалектика, 2023. 1152 с.
3. Ріхтер Дж. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C#. Харків: Фабула, 2019. 896 с.
4. Фаулер М. Архітектура корпоративних програмних систем. Київ: Вільямс, 2018. 544 с.
5. Кнут Д. Е. Мистецтво програмування. Том 1. Основні алгоритми. Львів: Видавництво Старого Лева, 2021. 672 с.
6. Сілбершац А., Гелвін П., Гейн Г. Операційні системи: концепції та принципи. Київ: Книжкове видавництво "Наш Формат", 2020. 944 с.
7. Дейтел П., Дейтел Х. Як програмувати на C#. Дніпро: Моноліт, 2017. 1088 с.
8. Буч Г., Максимчук Р., Енгл М. Об’єктно-орієнтований аналіз і проектування з прикладами додатків. Харків: Ранок, 2019. 720 с.
9. Седжвік Р., Уейн К. Алгоритми. Київ: Видавнича група "КМ-Букс", 2022. 960 с.
10. Макконнелл С. Досконалий код: практичний посібник із розробки програмного забезпечення. Львів: Літопис, 2020. 896 с.
11. Гама Е., Хелм Р., Джонсон Р., Влісідес Дж. Прийоми об’єктно-орієнтованого проєктування. Патерни проєктування. Київ: Вільямс, 2016. 432 с.
12. Шилдт Г. C# 8.0: повний довідник. Харків: Фабула, 2021. 1296 с.
13. Фріман Е., Фріман Е., Сьєрра К., Бейтс Б. Патерни проєктування. Львів: СПОЛОМ, 2018. 672 с.
14. Кормен Т., Лейзерсон Ч., Рівест Р., Стайн К. Алгоритми: побудова й аналіз. Київ: Видавництво "Наш Формат", 2019. 1296 с.

15. Скіннер Дж. Програмування баз даних із використанням SQLite. Дніпро: Моноліт, 2020. 384 с.
16. Ліберті Дж., Мак-Дональд Б. Програмування на C# для початківців. Харків: Ранок, 2017. 608 с.
17. Альбахарі Дж., Альбахарі Б. C# 8 у стислому викладі. Київ: Діалектика, 2021. 1056 с.
18. Майерс Г. Мистецтво тестування програм. Львів: Видавництво Старого Лева, 2018. 224 с.
19. Прессман Р. Інженерія програмного забезпечення: практичний підхід. Київ: Книжкове видавництво "Наш Формат", 2020. 928 с.
20. Соммервіль І. Інженерія програмного забезпечення. Харків: Фабула, 2019. 816 с.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace ParkingManagementSystem
{
    partial class MainForm
    {
        /// <summary>
        ///  Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        ///  Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        ///  Required method for Designer support - do not modify
        ///  the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            dataGridView1 = new DataGridView();
```