Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЛЯ ПРОСУВАННЯ ПРОДАЖ АВТОМОБІЛІВ ЗАСОБАМИ C#

SOFTWARE AND HARDWARE COMPLEX FOR PROMOTING THE SALE OF CARS USING C#

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Серпухов Богдан Андрійович

(підпис)

Керівник:
к.т.н., доцент
Пех Петро Антонович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 06 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Серпухову Богдану Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Програмно-апаратний комплекс для просування продаж автомобілів засобами C#*

Керівник роботи *к.т.н., доцент Пех Петро Антонович*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.06.2025р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обгрунтування системного, наповнення, оцінка ергономічних та надійністних параметрів проектованої системи, функціонально-структурна схема роботи об'єкта проектування.

5. Перелік графічного (ілюстративного) матеріалу:
Код розробленого комплексу для просування продажу
Інтерфейс комплексу продажу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Огляд засобів для розробки програмно-апаратних комплексів	*к.т.н.доцент Пех П.А.*		
Інструменти для розробки програмно-апаратного комплексу засобами c#	*к.т.н.доцент Пех П.А.*		
Розробка програмно-апаратного комплексу для просування продажу автомобілів засобами c#	*к.т.н.доцент Пех П.А.*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2025 р.	Виконано
2.	*Вибір інструментів для розробки програмно-апаратного комплексу засобами c#*	до 02.03.2025 р.	Виконано
3.	*Розробка програмно-апаратного комплексу для просування продажу автомобілів засобами c#*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 15.05.2025 р.	Виконано
9.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
11.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ (підпис) Серпухов Б.А. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Пех П.А. (прізвище, ініціали)

АНОТАЦІЯ

Серпухов Б.А. Програмно-апаратний комплекс для просування продаж автомобілів засобами C#.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

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

У роботі виконано огляд середовищ та мов програмування для розробки програмно-апаратних комплексів засобами C# та обгрунтовано їх вибір. Дано детальну характеристику інструментів для розробки програмно-апаратного комплексу засобами C#WinForms.NET Framework. Розроблено програмно-апаратний комплекс у вигляді проекту засобами C#, який складається з наступних форм: «Напрями діяльності салону», «Перегляд моделей автомобілів», «Тест-драйв автомобіля», «Відгуки клієнтів салону», «База даних автомобілів» та «База даних клієнтів». У процесі розробки проекту використані такі компоненти платформи NET Framework, як Form, Button, Label, Textbox, GroupBox, RadioButton та багато інших. Налаштовані властивості об'єктів, які створені на базі цих компонент, запрограмовані обробники подій, пов'язані з цими об'єктами. Розроблені три локальні бази даних. Передбачено перегляд вмісту цих баз даних, додавання нових записів, редагування та вилучення існуючих записів.

Ключові слова: середовище Visual Studio, мова C#, платформа APP .NET Framework, фреймворк .NET Framework, програмно-апаратний комплекс.

ANNOTATION

Serpukhov B. Software and hardware complex for promoting car sales using C#.

Bachelor's qualification thesis of the EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technial University. Lutsk, 2025.

Bachelor's qualification thesis consists of an introduction, three sections, conclusions, a list of used sources and appendices.

The thesis reviews the environments and programming languages for developing software and hardware complexes using C# and justifies their choice. A detailed description of the tools for developing a software and hardware complex using C#WinForms.NET Framework is given. A software and hardware complex developed in the form of a project using C#, which consists of the following forms: «Directions of salon activity», «View of car models», «Car test drive», «Salon customer reviews», «Car database» and «Customer database». During the development of the project, the following components of the .NET Framework platform were used: Form, Button, Label, Textbox, GroupBox, RadioButton and many others. The properties of objects created on the basis of these components were configured, event handlers associated with these objects were programmed. Three local databases were developed. It is provided to view the contents of these databases, add new records, edit and delete existing records.

Keywords: Visual Studio environment, C# language, APP .NET Framework platform, .NET Framework framework, hardware and software complex.

ЗМІСТ

ВСТУП

Актуальність теми. Розробка сучасних програмно-апаратних комплексів є і залишатиметься у майбутньому актуальною проблемою, оскільки запит на такі комплеси великий і з часом лише зростатиме, а засоби їх розробки постійно удосконалюються. До таких розробок відноситься і програмно-апаратний комплекс, розроблений у кваліфікаційній роботі, який вирішує низку завдань салону з просування продаж автомобілів.

Мета дослідження полягає у розробці програмно-апаратного комлексу з просування продаж автомобілів засобами C#.

Об’єкт дослідження – програмні та апаратні засоби і процес розробки програмно-апаратного комлексу з просування продаж автомобілів засобами C#.

Предмет дослідження – програми та апаратні засоби для просування продаж автомобілів. Дослідження передбачає вибір та детальне вивчення інструментів для вирішення завдань розробки програм для просування продаж автомобілів та безпосереднє створення комплексу таких програм. Розроблений програмно-апаратний комплекс розглядається як результат виконання кваліфікаційної роботи.

Завдання, які необхідно виконати:

– виконати огляд середовищ та мов програмування і обгрунтувати їх вибір для розробки програмно-апаратного комплексу для просування продаж автомобілів;

– дати детальну характеристику інструментів для розробки програмно-апаратного комплексу у вигляді проекту C# WinForms;

– розробити форму «Напрями діяльності салону»;

– розробити форму «Перегляд марок та моделей автомобілів»;

– розробити форму «Тест-драйв автомобілів»;

– розробити форму «Відгуки клієнтів салону»;

– розробити форму «База даних автомобілів»;

– розробити форму «База даних клієнтів салону».

Методи досліджень. Теоретичні дослідження базуються на сучасних технологіях програмування, зокрема, на технології об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає у розробці програмно-апаратного комплексу засобами середовища C# WinForms.NET Framework.

Практичне значення одержаних результатів. Розроблений програмно-апаратний комплекс може бути використаний салонами з продажу автомобілів з метою збільшення обсягів цих продаж, а також у навчальному процесі університетів для ознайомлення студентів з новітніми технологіями розробки проектів.

Особистий внесок здобувача. Програмно-апаратний комплекс з просування продажу автомобілів розроблений здобувачем самостійно.

РОЗДІЛ 1
ОГЛЯД ЗАСОБІВ ДЛЯ РОЗРОБКИ ПРОГРАМНО-АПАРАТНИХ КОМПЛЕКСІВ

1.1 Середовище програмування Visual Studio

Будь-який програмно-апаратний комплекс створюєься у певному середовищі програмування засобами конкретної мови. Отже, вибір середовища та мови програмування – це перша задача, від вирішення якої значною мірою залежить подальша доля розробки [1, 2].

«Середовище програмування – це інтегроване середовище розробки програмних засобів (IDE – integrated development environment), яке містить редактор вихідного коду, компілятор чи/або інтерпретатор, засоби автоматизації збірки та засоби спрощення розробки графічного інтерфейсу користувача» [3].

Середовище програмування Visual Studio – це комплексне інтегроване середовище розробки, яке можна використовувати для введення, редагування, налагодження та виконання коду. Середовище Visual Studio включає компілятор, засоби виконання коду, керування версіями та багато інших функцій для покращення кожного етапу процесу розробки програмного забезпечення. Visual Studio гарно підходить, наприклад, для створення, налагодження та тестування .NET та C++ додатків, розробки ASP.NET сторінок у поданні веб-конструктора, розробки кросплатформенних мобільних та класичних додатків за допомогою фреймворку .NET або створення адаптивних веб-інтерфейсів [4-10].

Після запуску Visual Studio на екрані з’являється його перше вікно (рис. 1.1). У цьому вікні можна побачити версію середовища Visual Studio 2022, розділи Open recent та Get started. Розділ Open recent пропонує список проектів, з якими розробник працював останнім часом, а розділ Get started пропонує чотири вкладки: для роботи з репозиторієм проектів, для відкриття папок, для відкриття проектів та для створення нових проектів.

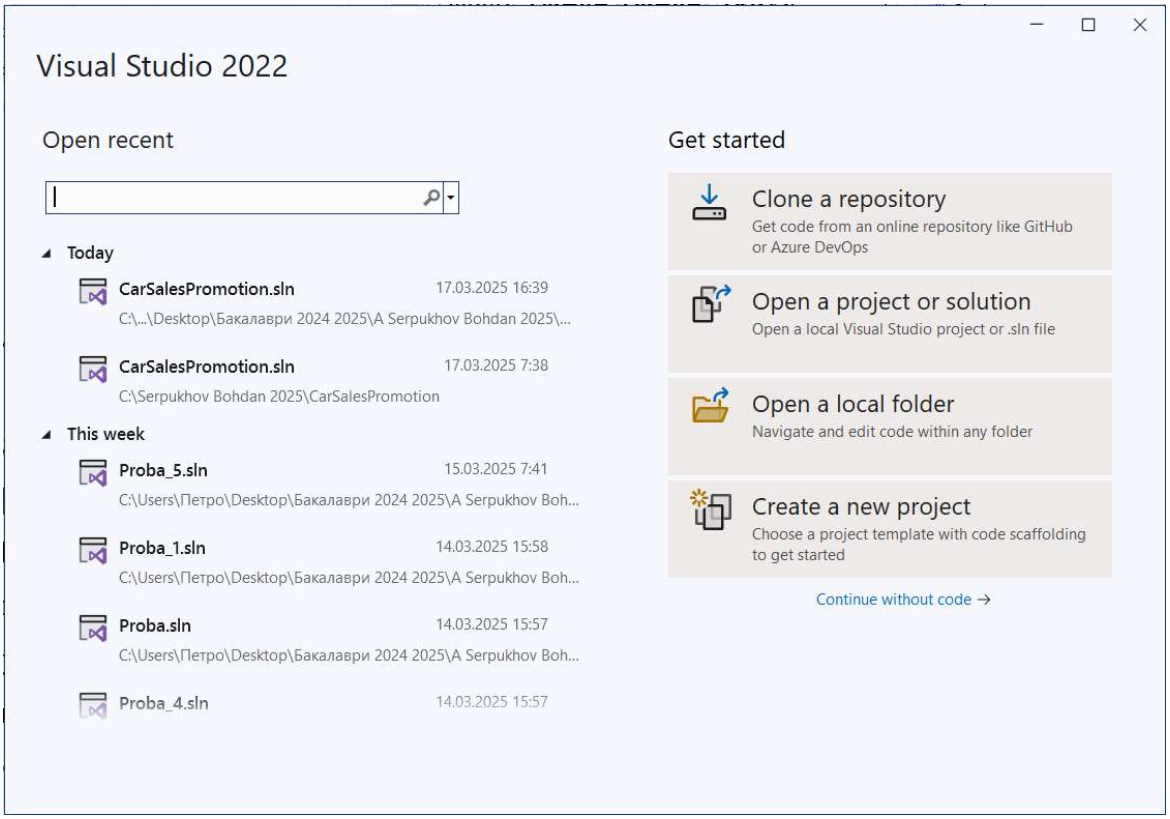

Рисунок 1.1 – Головне вікно середовища Visual Studio

Нехай потрібно створити новий проект CarSalesPromotion засобами C# Windows Forms. Для цього клацаємо по вкладці Create a new project. З'являється вікно Create a new project (рис. 1.2). У ньому вибираємо зі списку багатьох запропонованих мов вибрану нами мову C# (рис. 1.3). Той факт, що Visual Studio дозволяє створювати програми багатьма іншими мовами програмування, свідчить про його потужність і універсальність.

Знову ж таки, Visual Studio пропонує багато різних платформ для створення проектів мовою C#: Console App, Blazor Web App, Blazor Server App, ASP NET Core Web App (MVC), WPF Application та інші. Платформи можуть бути реалізовані у різних операційних системах (Windows, Linux і т.д.), тому вибране нами середовище програмування є кросплатформенним, і це одна із багатьох його сильних сторін.

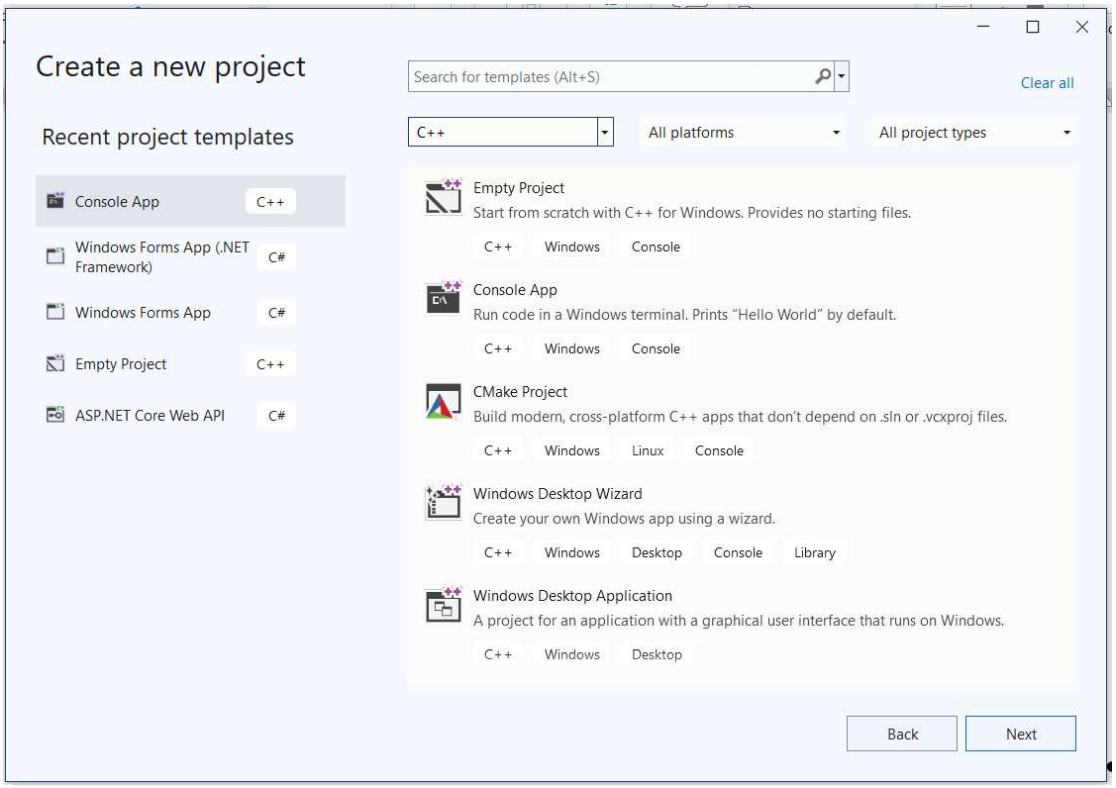

Рисунок 1.2 – Вікно Create a new project

Рисунок 1.3 – Вибір мови програмування C#

Вибравши мову та платформу для створення проекту, натискуємо кнопку Next, щоб перейти до наступного вікна Configure your new project.

У цьому ж вікні Create a new project вибираємо потрібну нам платформу Windows Forms APP (.NET Framework) (рис. 1.4).

У вікні Configure your new project у полі Project name вводимо ім'я проекту CarSalesPromotion, у полі Location задаємо шлях до папки проекту C:\Serpukhov, у полі Framework вибираємо фреймворк .NET Framework 4.7.2 і натискуємо кнопку Create, щоб створити порожній проект (рис. 1.5).

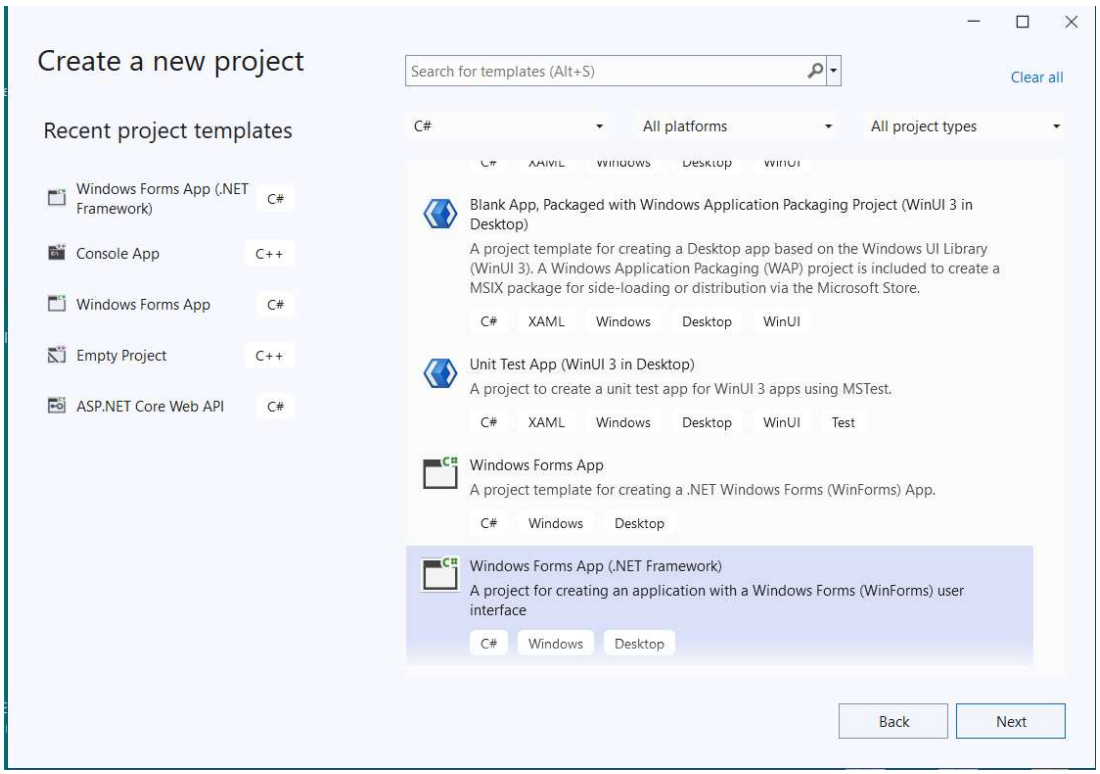

Рисунок 1.4 – Вибір мови C# та платформи Windows Forms APP (.NET Framework)

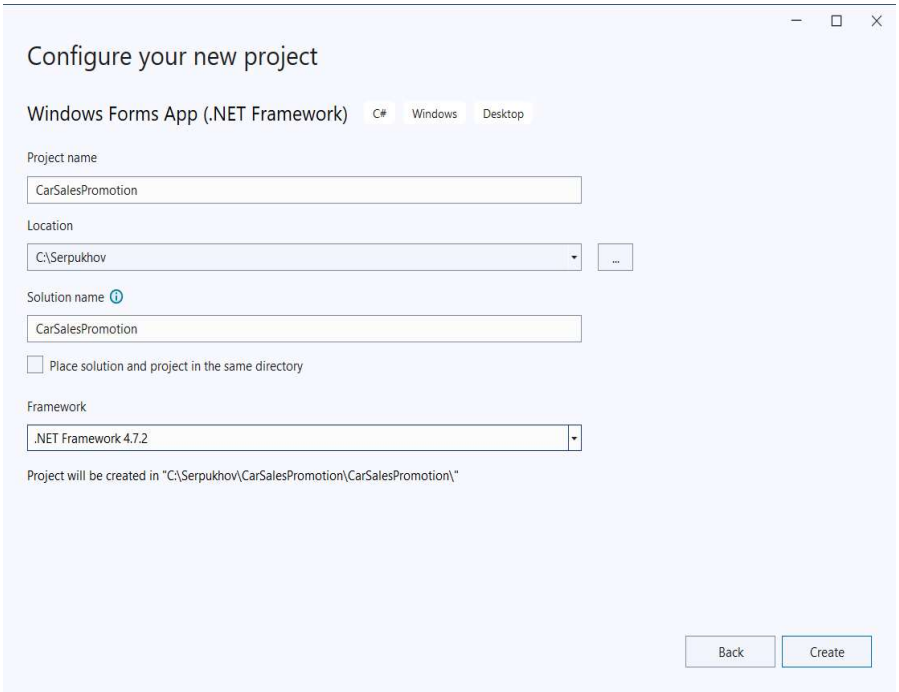

Рисунок 1.5 – Ввід імені проекту, вибір папки для розміщення проекту та версії фреймворку

Середовище Visual Studio надає розробникам широкі можливості для

ефективної та спільної розробки високоякісного коду, серед яких:

– потужні засоби написання коду та функції – все, що необхідно для створення програм в одному місці;

– підтримка багатьох мов – C++, C#, JavaScript, TypeScript, Python та інших;

– кросплатформенна розробка – створення програм на різних платформах;

– інтеграція управління версіями;

– спільна робота над кодом команди програмістів;

– розробка проектів за допомогою штучного інтелекту.

«Інтегроване середовище розробки Visual Studio надає безліч функцій, які спрощують написання коду та керування ними. Наприклад, можна швидко написати код за допомогою засобів штучного інтелекту. До цих засобів відносяться GitHub Copilot та IntelliCode. Є можливість вносити покращення в код за допомогою засобів, що пропонують дії, або розгортання та згортання блоків коду за допомогою функціоналу згортання. Є можливість впорядкування коду за допомогою браузера рішень, який показує код, впорядкований за файлами, або код, впорядкований за класами. Тобто, GitHub Copilot, GitHub Copilot Chat та IntelliCode допомагають розробникам швидше і з більшою точністю написати код, а також легше справлятися з іншими завданнями розробки, такими як написання модульних тестів, налагодження профілів» [4].

Середовище Visual Studio дозволяє компілювати та створювати програми, створювати збірки та тестувати їх у налагоджувачі. Можна запускати збірки з кількома процесорами для проектів C++ та C#. Можна створити конфігурацію збірки, приховати певні попереджувальні повідомлення або збільшити вихідні дані збірки.

«Вбудована система налагодження в Visual Studio дозволяє легко виконувати налагодження, профілювання та діагностику проектів. Є можливість проходити крок за кроком по коду і переглядати значення, що зберігаються у змінних, спостерігати за змінними, щоб встановити, коли змінюються їх значення, вивчати процес виконання коду» [4].

Середовище Visual Studio дає можливість отримати високоякісний код за допомогою комплексних засобів тестування. Модульні тести дозволяють розробникам та тестувальникам швидко знаходити помилки логіки у коді. Можна проаналізувати обсяг коду, який тестується, та побачити миттєві результати тестування.

«Середовище Visual Studio за допомогою інтегрованих функцій Git Visual Studio дозволяє клонувати, створювати або відкривати власні репозиторії. У вікні інструменту Git є все необхідне для фіксації змін, керування гілками та вирішення конфліктів злиття. Якщо у розробника є обліковий запис GitHub, він може керувати цими репозиторіями безпосередньо у Visual Studio.

Visual Studio Live Share забезпечує спільну розробку у режимі реального часу. З Live Share можна поділитися своїм проектом із колегами, незалежно від мови чи платформи. Можна швидко дійти до суті проблеми, дозволивши команді розробників підключатися, переміщатися, задавати точки зупинки та вводити текст у редакторі» [4].

«Розгорнувши програму, службу або компонент, можна розподіляти їх для встановлення на інших комп’ютерах, пристроях, серверах або в хмарі. Можна вибрати відповідний метод для типу розгортання. Є можливість надання спільного доступу до програм та коду шляхом публікації їх в Інтернеті або Azure або шляхом розгортання у мережевій або локальній папці.

Середовище Visual Studio випускається у трьох варіантах:

– Community – безкоштовне, повністю багатофункціональне інтегроване середовище розробки для учнів та окремих розробників з відкритим кодом;

– професійний варіант на основі підписки для окремих розробників або невеликих команд;

– Enterprise – варіант на основі підписки для малих та великих команд»[4].

1.2 Середовище програмування PyCharm

Середовище PyCharm було розроблене компанією JetBrains – відомою

міжнародною компанією, що спеціалізується на розробці інструментів для програмування різними мовами, як-от Python, Java, Kotlin, C#, F#, C++, Ruby, PHP, JavaScript і багатьох інших. Компанія, яка була заснована раніше під ім'ям IntelliJ, сьогодні надає різноманітні засоби для командної роботи та підтримки розробки провідних мов програмування.

«Для того, щоб почати використовувати PyCharm, слід:

– встановити PyCharm з офіційного сайту JetBrains, вибравши Community або Professional Edition;

– створити новий проєкт, вибравши мову (зазвичай Python) та інтерпретатор Python;

– відкрити файли з кодом Python, перетягнувши їх у вікно або через меню «File» – «Open»;

– ознайомитися з інтерфейсом, включно з редактором коду, вікном проєкту та «Tool Windows»;

– ввести код, використовуючи функцію автодоповнення (налагоджувач використовується за необхідності);

– вивчити основні функції PyCharm, включно з рефакторингом і роботою з Git;

– розширити можливості PyCharm за допомогою плагінів із "Plugin Repository» [11].

«PyCharm надає розробникам безліч базових функцій, які значно спрощують процес програмування. Ось кілька з них:

– налагодження коду: потужний налагоджувач дає змогу знаходити та виправляти помилки, встановлюючи точки зупинки та аналізуючи значення змінних;

– рефакторинг: інструменти рефакторингу допомагають змінювати структуру коду без втрати функціональності;

– підтримка систем контролю версій: інтеграція з Git, Mercurial та іншими системами дає змогу зручно працювати з історією змін;

– автодоповнення коду: інтелектуальне автодоповнення прискорює процес написання коду і знижує ймовірність помилок;

– інспектування коду: статичний аналіз коду допомагає виявляти потенційні помилки та підтримувати високу якість коду.

– інтеграція з віртуальними оточеннями: PyCharm надає зручний інтерфейс для роботи з віртуальними оточеннями Python; можна створювати та керувати різними оточеннями, що дозволяє ізолювати залежності та бібліотеки для кожного проекту» [11].

Це лише невеликий огляд базових функцій PyCharm. Маючи подібний набір інструментів, розробники можуть ефективніше працювати над проектами, покращувати якість коду та зосереджуватися на творчому боці програмування.

1.3 Середовище програмування Eclipse

Середовище програмування Eclipse Java IDE – відкрите інтегроване середовище розробки мовою Java, яке розповсюджується та підтримується компанією Eclipse Foundation. Спочатку Eclipse створювалася компанією як наступник середовища розробки IBM VisualAge. Незважаючи на те, що розробка коштувала компанії 40 мільйонів доларів, вихідний код став відкритим, і IBM передала інструмент на подальший розвиток незалежному співтовариству.

«Eclipse IDE безкоштовне, що дає йому велику перевагу. Будь-який розробник може встановити собі нову версію цього середовища. Eclipse також має платну версію – MyEclipse, але це скоріше окремий проект, побудований на базі основного Eclipse. MyEclipse пропонує повнофункціональну платформу для розробки програмного забезпечення, а також додаткові пакети. Наприклад, MyEclipse Blue підтримує інтеграцію зі сімейством продуктів WebSphere, а Professional-версія реалізує розширений функціонал для Enterprise-розробки. Eclipse, на відміну від платної версії, це ядро, до якого підключаються додаткові плагіни для створення просунутого IDE» [12].

«Зупинимося на основних характеристиках та інструментах середовища.

«Хоча про середовище кажуть «найпопулярніший інструмент» та «встановлюється безкоштовно», дехто думає, що функціонал IDE урізаний і підтримує лише стандартний набір функцій. Це не так. Eclipse Java IDE – повноцінний інструмент, прийнятий як корпоративний стандарт у багатьох великих компаніях. Під час встановлення середовища потрібно вибрати один з його варіантів. Варіантів багато, однак, на наш погляд, найбільш цікаві два – for Java Developers та for Enterprise Java Developers. Як випливає з їх назв, варіант for Java Developers підходить для розробки Java-проектів та Web-додатків. Поточна версія Eclipse на момент написання огляду – 2019-09 R (4.13.0). Для розробки серйозних проектів буде потрібна варіант for Enterprise Java Developers» [12].

Редактор коду в Eclipse виглядає просто і зрозуміло – стандартне вікно для роботи з кодом, де відображається структура проекту та інші параметри. За першого встановлення за замовчуванням активується темна тема. Світла тема не так тішить око, хоча це суб'єктивно. Також приємна особливість – класи стандартної бібліотеки імпортуються автоматично, ця функція включена за замовчуванням. В арсеналі Eclipse є корисна функція QuickFix, яка допомагає швидко виправити рядок коду. За допомогою цієї функції можна виконувати будь-які дії – від простого вилучення локальної змінної до складніших операцій.

«Рефакторинг коду Java в Eclipse відрізняється від цього процесу в IDEA. Справа в тому, що Eclipse не вистачає розуміння контексту, як це робить IDEA. Це помітно при рефакторингу. Найпростіший приклад – зміна імені змінної. IDEA враховує назву, тип, значення, імена попередніх змінних схожого типу та пропонує відповідне ім'я. Eclipse цього не вміє. Якщо користувач звик до «інтелектуального» рефакторингу в IDEA, то знадобиться час, щоб відвикнути від цьго. В іншому процедура рефакторингу підтримує всі стандартні функції. Наприклад, вилучення інтерфейсу, safe delete та інші (всього 23)» [12].

Процес налагодження програми в Eclipse нескладний, стандартні функції інтуїтивні, в debug-меню відображається стан змінних та поточного контексту. Eclipse має візуальний редактор для елементів графічного інтерфейсу – Visual

Editor, а також компілятор GUI. Visual Editor підтримує AWT/Swing і доступний Eclipse починаючи з версії 2.1, цей редактор потрібно підключати окремо.

Серед інструментів Eclipse також варто виділити роботу з системами контролю версій, автобудівник, систему збирання, інтеграцію з найпопулярнішими фреймворками, зручну роботу з додатковими типами файлів (sql, html, js тощо). Далі зупинимося на плюсах та мінусах використання Eclipse.

«Отже, до переваг Eclipse можна віднести наступне.

Eclipse можна гнучко налаштувати під себе завдяки простій розробці плагінів. Як уже сказано, Eclipse – це ядро: за першої установки це IDE ще не може називатися повноцінним середовищем. Для цього потрібно встановити додаткові плагіни, і тоді розробка на Eclipse стане набагато простішою та приємнішою. Кожен розробник може створити своє IDE.

Eclipse Java IDE розповсюджується безкоштовно.

Зупинимося на недоліках Eclipse. Хоч як це дивно, плагіни – це і плюс, і мінус. Справа в тому, що за наявності великої кількості несумісних плагінів IDE може впасти і буде потрібна повторна інсталяція. Таке зустрічається у великих проектах, але ця особливість не вказана у офіційній документації» [12].

1.4 Огляд мов програмування

«Мова програмування – це система позначень для опису алгоритмів і структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми та дії, які виконує виконавець (комп’ютер) під її управлінням» [13].

Наведемо популярні на сьогоднішній день мови програмування.

C# – це мова, яка дозволяє реалізувати всі переваги технології об’єктно-орієнтованого програмування. Вона постійно розвивається і удосконалюється, що гараантує їй хороше майбутнє [14-17].

C/C++ – це мова високого рівня, яка дає базові знання з програмування.

Вивчається у багатьох університетах [18-24].

Python – це мова високого рівня загального призначення. Python почав користуватися популярністю протягом останніх років. Це сталося тому, що університети США та деяких інших західних країн почали викладати цю мову, яка відрізняється від інших мов програмування своєю простотою [25, 26].

Java – це також мова високого рівня загального призначення. Вона використовується для розробки і backend, і front-end сайтів, як і в багатьох інших різнопланових сферах, але є достатньо складною.

JavaScript (JS) – це мова для розробки веб-сайтів, мобільних програм, серверів та багато іншого. Знати JS потрібно і фронтенд- розробникам, і бекенд-розробникам, і фулстек-спеціалістам. Мова JavaScript була створена для браузера завдяки технології, яка називається WebAssembly, що дозволяє трансформувати код і виконувати його в браузері. Відповідно, вона працює на всіх комп'ютерах та смартфонах.

PHP – це мова програмування, за допомогою якої створюються веб – сайти. Цією мовою написана також і програма WordPress, яка сама є системою управління контентом сайту (CMS), тобто самостійною комплексною програмою, за допомогою якої розробляються сайти. Зокрема, і сайт Facebook. То ж не дивно, що ця мова була і є досить затребуваною.

SQL/MySQL – це мова для роботи з базами даних. Глибоке знання лише однієї цієї мови достатнє для працевлаштування у сфері IT [25-28]

Таким чином, за результатами проведеного огляду для розробки програмно-апаратного комплексу ми обираємо середовище програмування Visual Studio та мову програмування C#.

РОЗДІЛ 2
ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ЗАСОБАМИ C#

2.1 Створення C# проекту засобами середовища Visual Studio

Головне вікно проекту CarSalesPromotion (рис. 2.1), що з'являється зразу після його створення, складається зі звичних нам Меню команд та Панелі інструментів, розташованих у верхній частині вікна, та вікон Form1, Toolbox, Solution Explorer, Properties, Output, Error List. Якщо якесь із цих вікон не відображається на екрані, його легко встановити за допомогою команди View головного меню. Кількість вікон і які саме вікна потрібні розробнику проекту в даний момент, він вирішує самостійно. На нашу думку, під час розробки всі шість вікон, перерахованих вище, бажано мати постійно присутніми на екрані, тому що доступність до них значно скорочує час розробки проекту.

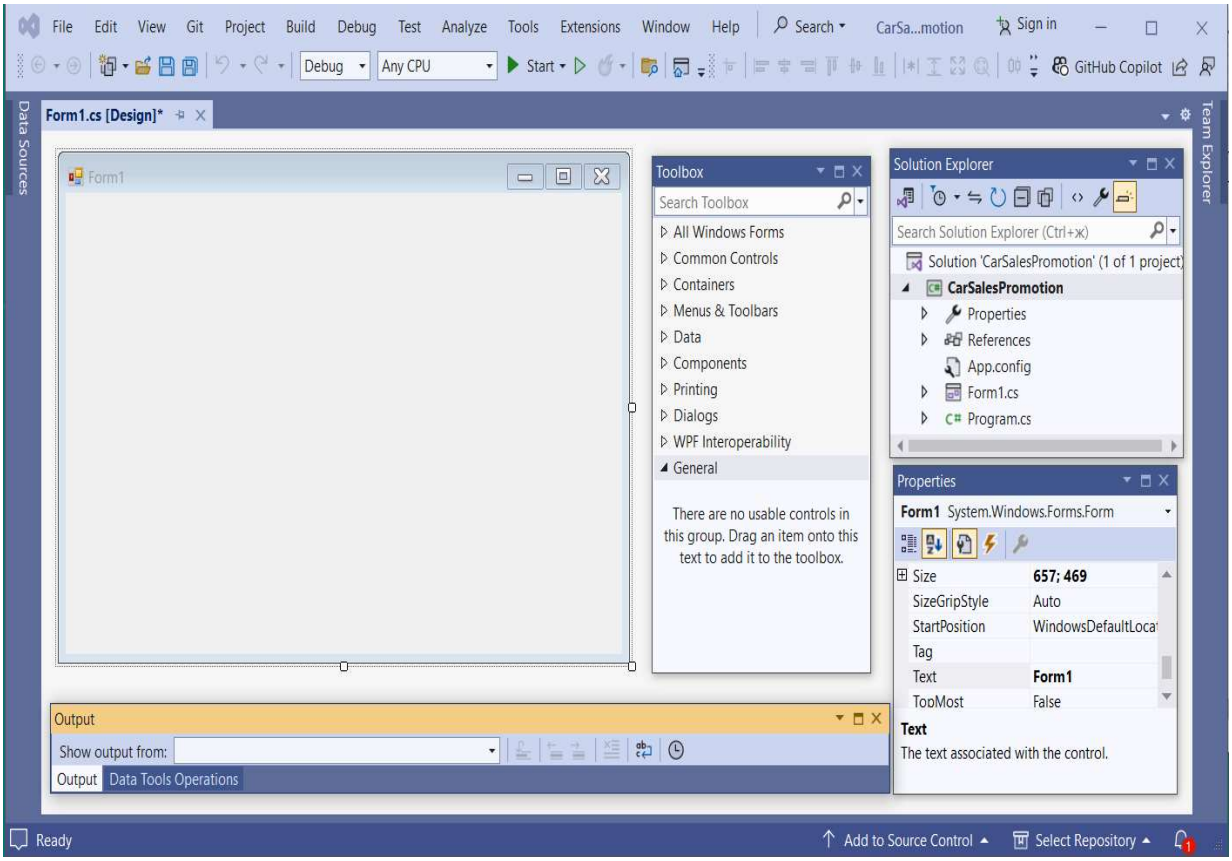

Рисунок 2.1– Головне вікно проекту CarSalesPromotion

2.2 Створення елементів проекту та налаштування їх властивостей

Проект засобами середовища Windows Forms складається з багатьох форм. Структуру проекту можна переглянути у вікні браузера проектів Solution Explorer (рис. 2.2). Як бачимо, наш проект поки що складається лише з однієї форми Form1. Варто зазначити, що вікно Solution Explorer часто використовуються під час роботи над проектом. Це вікно дозволяє швидко виконувати різні операції з додавання нових або вилучення існуючих компонент проекту, або виконати багато інших операцій. Наприклад, щоб додати до нашого проекту нову форму Form2, потрібно клацнути правою кнопкою миші по імені проекту CarSalesPromotion, у спадному меню, що відкривається вибрати вкладку Add, далі у новому спадному меню вибрати вкладку Forms (Windows Forms), у вікні Add New Item у полі Name ввести ім’я форми або залишити його таким, що пропонується (Form2) , і натиснути кнопку Add. З’явиться нова форма з ім’ям Form2. Процес створення форм можна продовжити. Кожна форма вирішує одну задачу зі загального алгоритму, тому форм має бути стільки, скільки задач ми плануємо розв’язати у проекті.

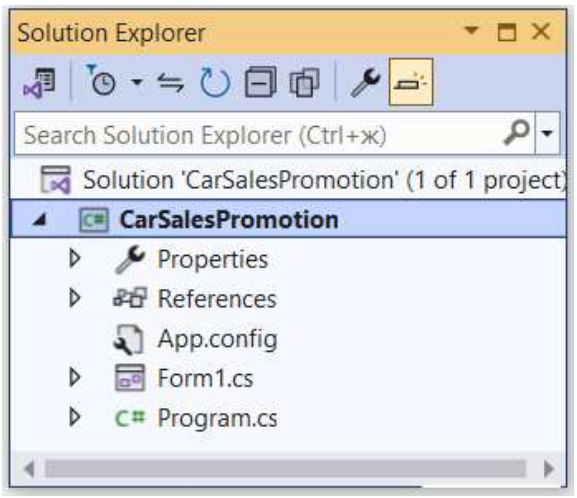

Рисунок 2.2 – Вікно Solution Explorer

Компонент Form є одним із найважливіших у кожному проекті. Головна (перша) форма Form1 (рис. 2.3) у проекті є обов’язковою. Чому ж форми є такими

важливими? Справа в тому, що всі інші елементи проекту створюються шляхом перенесення готових компонент з наявних бібліотек середовища на форми за відомою технологією «перетягнути і кинути». Звичайно, далі потрібно встановити потрібні властивості цих елементів та відповідним чином запрограмувати події, які пов’язані з ними.

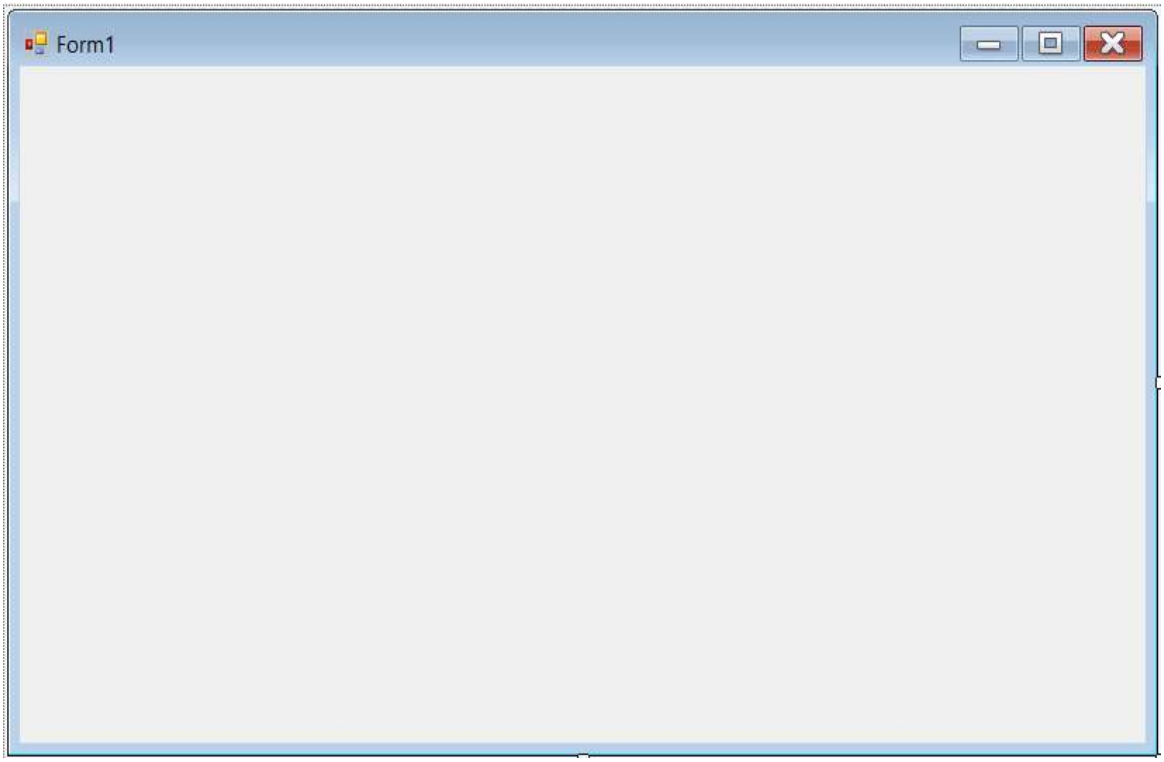

Рисунок 2.3 – Вигляд головної форми Form1 проекту

Ті компоненти, які переносяться на форму, вибираються за допомогою вікна Панель інструментів (Toolbox) (рис. 2.4). Всі компоненти цього вікна поділені на ряд категорій: All Windows Forms, Common Controls, Menus & Toolbars, Containers, Menus & Toolbars, Menus & Toolbars, Data, Components, Printing, Dialogs, WPF Interoperability, General. Звичайно, компоненти проекту можуть розроблятися і самим програмістом, якщо таких компонент немає на вкладці All Windows Forms. Крім того, є можливість підключити компоненти операційної системи Windows за допомогою діалогового вікна Choose Toolbox Items, яке викликається до роботи клацанням правої кнопки миші по вмісту контенту панелі інструментів Toolbox.

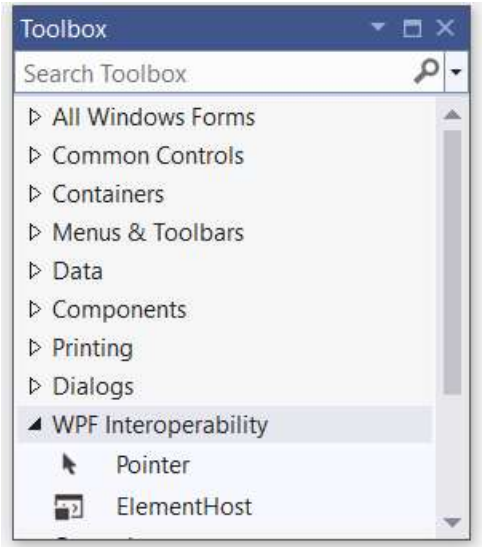

Рисунок 2.4 – Панель інструментів Toolbox

Вибравши категорію Common Controls, отримаємо доступ до компонентів цієї вкладки (рис. 2.5).

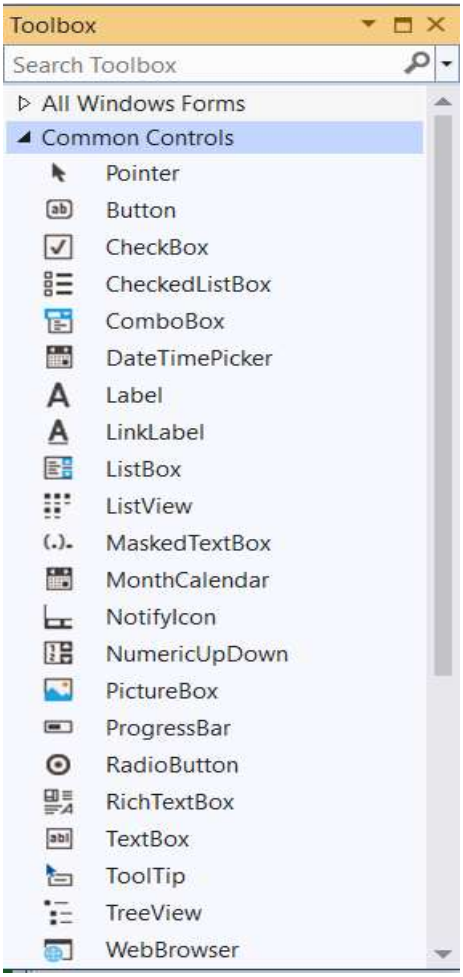

Рисунок 2.5 – Компоненти категорії Common Controls

Зокрема, для того, щоб перенести компонент Button на форму, досить двічі клацнути по імені Button у списку категорії Common Controls. Інший спосіб – натиснути лівою кнопкою миші на імені Button і, утримуючи кнопку натиснутою, перенести цей елемент у потрібне місце форм і відпустити кнопку. У результаті буде створений об’єкт класу button1, який успадковує всі властивості класу Button. Створені у такий спосіб елементи проекту потребують налаштування їх властивостей і програмування подій.

2.3 Налаштування властивостей елементів проекту

Об’єкти проекту, розташовані на певній формі проекту, мають багато різних властивостей, конкретні значення яких встановлюються за принципом замовчування. Розглянемо, яким чином можна змінювати значення цих властивостей за допомогою вікна Properties (рис. 2.6).

У верхній частині вікна Properties знаходяться піктограми Categorized, Alphabetical, Properties, Events, які використовуються для зміни властивостей об’єктів та програмування пов’язаних з ними подій. Для зручності пошуку властивості Properties та події Events об’єкта можуть бути відсортовані за категоріями за допомогою режиму Categorized, або за алфавітом за допомогою режиму Alphabetical.

Для зміни значення властивості об’єкта знаходимо в режимі Categorized потрібну властивість у лівому стовпчику вікна Properties, і клацаємо по ній, а у правому стовпчику Properties, задаємо значення цієї властивості об’єкта. У такий спосіб можна, наприклад, змінити розміри об’єкта.

Тут варто підкреслити той факт, що нам не доводиться кожен раз розробляти програмний код для створення об’єкта з новими розмірами, тобто, «придумувати велосипед», достатньо лише налаштувати властивості готового об’єкту. У цьому і полягає велика перевага об’єктно-орієнтованого програмування у порівнянні з раніше існуючими технологіями функціонального та структурованого програмування.

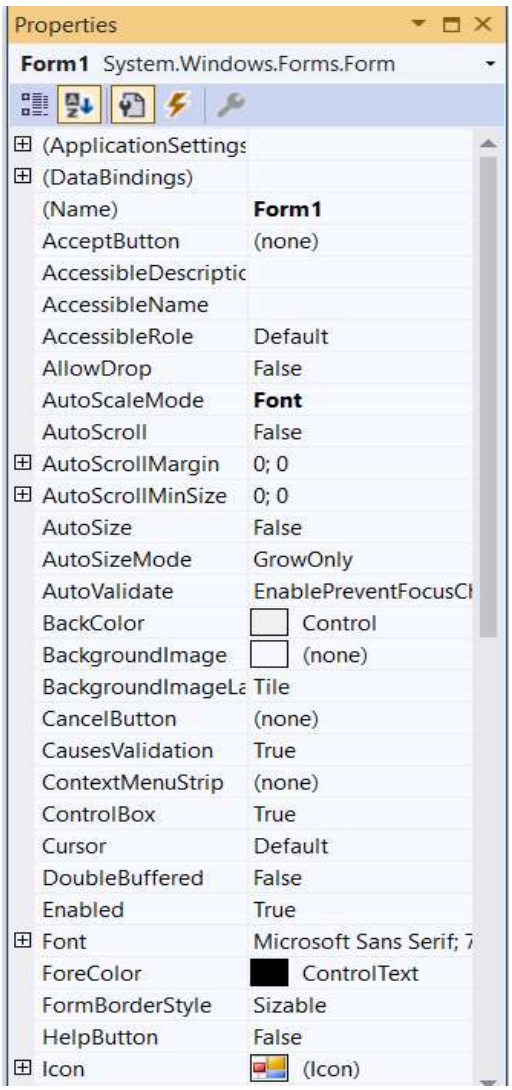

Рисунок 2.6 – Вікно Properties з відображенням властивостей форми Form1

2.4 Технологія налаштування деяких параметрів об’єктів проекту

Вікно Properties надає розробнику можливість змінювати значення властивостей об’єктів, розташованих на формі. Як правило, більшість значень властивостей об’єктів розробник залишає такими, якими вони встановлюються за принципом замовчування. Однак, деякі інші значення властивостей розробник змінює на свій розгляд. Тому далі розглянемо послідовність налаштування деяких властивостей об’єктів.

Налаштування шрифту тексту. Заголовок об’єкта відображається на формі у вигляді текстового надпису. Отже, потрібно вміти задавати значення властивості Text. Для цього у лівому стовпчику вікна Properties (рис. 2.7)

виділяємо властивість Font, а напроти неї у правому стовпчику клацаємо по піктограмі «…». З'явиться вікно Шрифт (рис. 2.8), у якому вибираємо тип шрифту Arial Rounded MT, стиль шрифту жирний, розмір шрифту 12 і натискуємо кнопку Ok. Інший спосіб задання параметрів вікна полягає у тому, що потрібно клацнути по значку «+» зліва від слова Font, і тоді нижче з'являться всі параметри шрифту, значення яких можна встановити безпосередньо у цьому вікні. Як бачимо, налаштування параметрів об'єкта можна виконати різними способами.

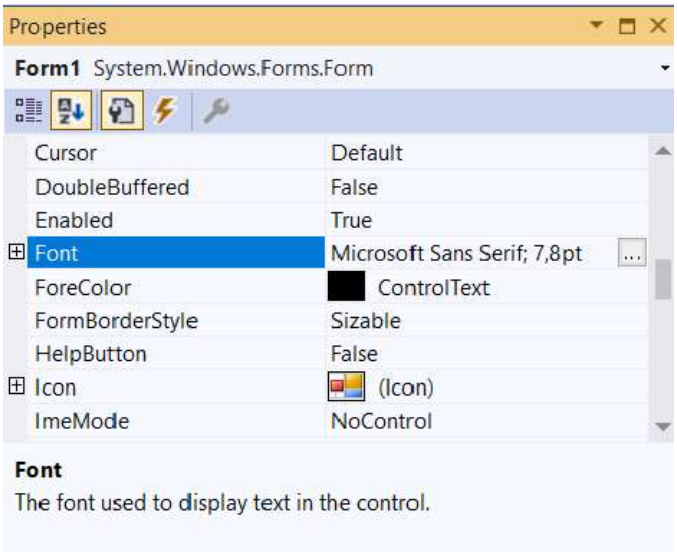

Рисунок 2.7 – Вибір властивості Font форми Form1

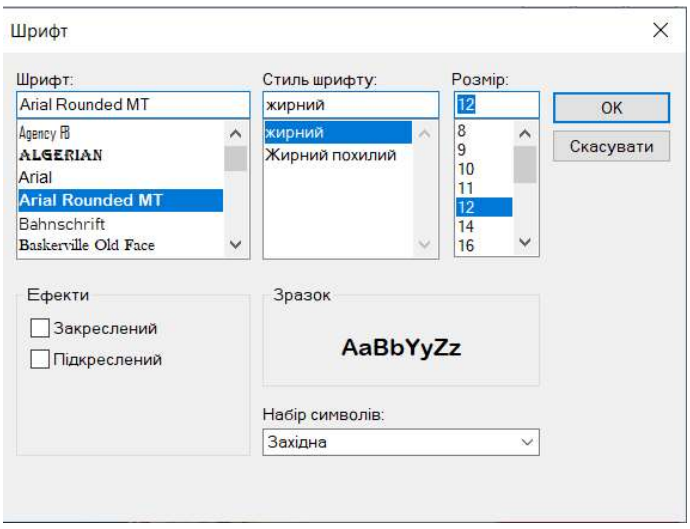

Рисунок 2.8 – Вибір властивості Font форми Form1

Вибір кольору. Потрібний колір можна знайти на сайті «colorpicker.me». Зайти на сайт просто. Для цього у поле пошуку вікна Google вводимо текст «colorpicker.me», і зразу відкриється вікно сайту (рис. 2.9), у якому за допомогою повзуна візуально підбираємо підходящий колір.

У даному випадку ми вибрали для заднього фону форми Form1 колір, який за системою RGB має значення 163; 25Ж 129, шістнадцятковий код кольору #a31981 (рис. 2.10).

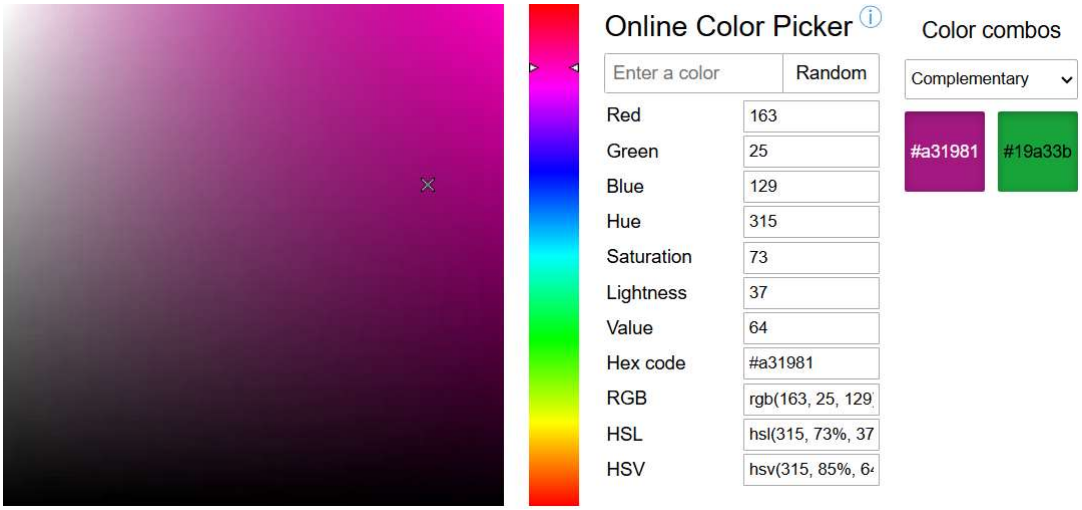

Рисунок 2.9 – Вибір кольору за допомогою сайту «colorpicker.me»

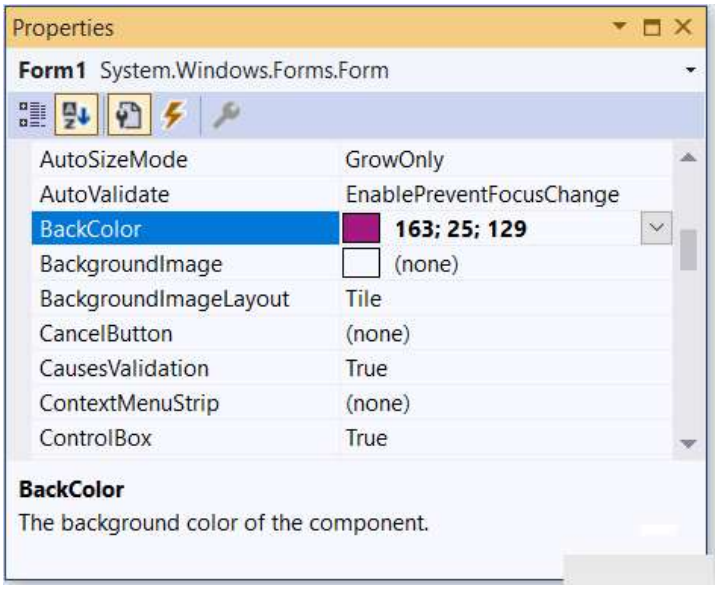

Рисунок 2.10 – Значення параметра BackColor форми Form1

2.5 Програмування обробників подій елементів проекту

У процесі розробки проекту часто виникає необхідність у програмуванні певних подій, пов'язаних з об'єктами. Для цього потрібно виділити об'єкт на формі, знайти у вікні Properties потрібну подію і двічі клацнути по ній, щоб створити заготовку коду функції, яка називається обробником події. Далі необхідно заповнити цю заготовку функції-обробника командами, які забезпечують власне обробку цієї події, тобто її функціонал.

Нехай, наприклад, нам потрібно, щоб клацання по кнопці button5 закривало форму Form1 і відкривало форму Form6, то обробник події Click кнопки button5, розташованої на формі Form1, виглядатиме як на рисунку 2.11. Перша команда закриває форму Form1, друга створює динамічний об'єкт f6 класу Form6, який за допомогою методу Show() відкриває Form6.

```
private void button5_Click(object sender, EventArgs e)
{
    this.Hide();
    Form6 f6 = new Form6();
    f6.Show();
}
```

Рисунок 2.11 – Код обробника події Click кнопки button5 форми Form1

Звичайно, після встановлення тих чи інших значень властивостей об'єктів, проект потрібно запустити на виконання, щоб переконатися, чи правильно він працює. Тоді у пригоді можуть стати вікна Error List та Output, які постійно використовуються під час налагодження та тестування проекту (рис. 2.12).

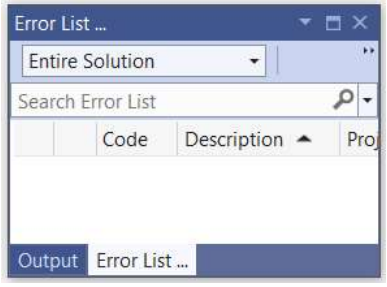

Рисунок 2.12 – Вікна Error List та Output

2.6 Створення локальної бази даних засобами C# Windows Forms

Часто у проектах не обійтися без використання баз даних. Платформа C# Windows Forms. NET Framework має інструменти для створення баз даних. Розглянемо, як у нашому проекті CarSalesPromotion можна створити локальну базу даних Database1.mdf, яка складатиметься лише з однієї таблиці Table з такими полями: Id – ідентифікатор запису, Name – прізвище, ім'я та по батькові студента. Тобто, база даних міститиме інформацію про студентів групи.

Додаємо до проекту CarSalesPromotion файл нової бази даних, яка базується на службах. Клацаємо правою кнопкою миші по імені проекту CarSalesPromotion у вікні Solution Explorer і виконуємо команду Add/Components… У вікні Add New Item вибираємо вкладку Service-based Database, залишаємо стандартне ім'я файла бази даних Database1.mdf і клацаємо по кнопці Add. У Solution Explorer з'явиться ім'я файлу створеної бази даних Database1.mdf.

У вікні Solution Explorer двічі клацаємо по імені файлу Database1.mdf. З'явиться вікно Server Explorer (рис. 2.13), у якому відобразиться структура нашої бази даних, але вона буде поки що порожньою. Клацаємо правою кнопкою миші по вкладці Table і виконуємо команду Add New Table. Новій таблиці можна задати оригінальне ім'я, але ми залишимо його без змін, тобто Table. Виконуємо команду Save All, щоб завершити створення нової таблиці Table бази даних Database1.mdf. Оновлюємо вміст бази даних Database1.mdf переходимо до створення структури таблиці Table та заповнення її даними (рис. 2.14). Ключовим полем встановлюємо поле Id таблиці Table, а для типу поля Name таблиці вибираємо тип nvarchar(50).

Структура єдиної таблиці Table нашої бази даних розроблена. Оновлюємо вміст таблиці. Взагалі кажучи, операцію оновлення потрібно виконувати постійно після внесення у неї нових даних або у разі зміни її структури.

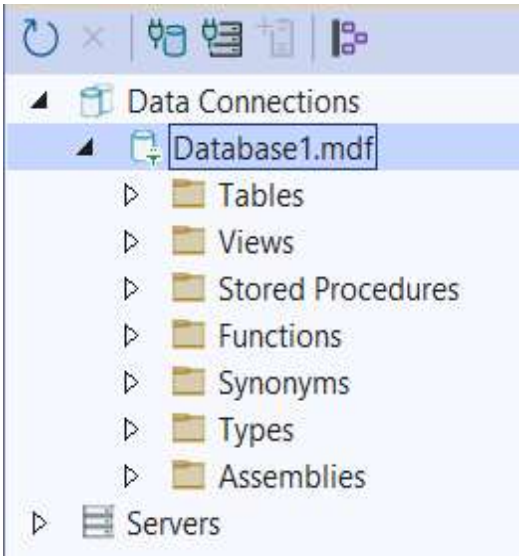

Рисунок 2.13 – Структура бази даних Database1.mdf

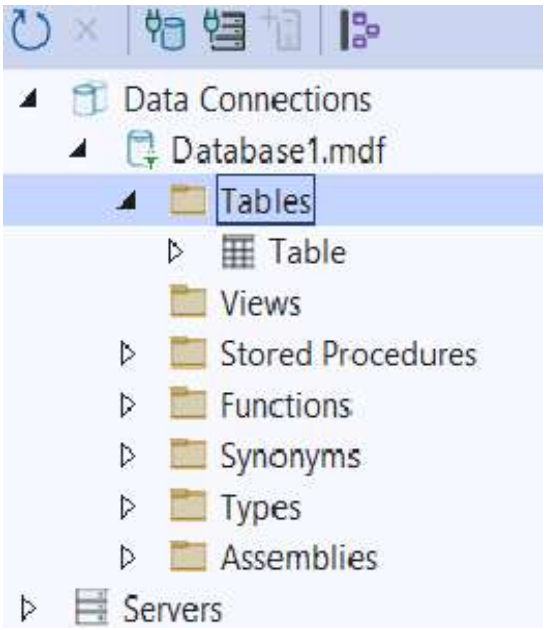

Рисунок 2.14 – Виклик таблиці Table бази даних Database1

Далі відкриваємо її в режимі перегляду і вводимо в неї інформацію про студентів групи (рис. 2.15). Як бачимо, таблиця бази даних створена, і вона може бути використана у проекті.

dbo.Table [Data] | dbo.Table [Design] | Form1.cs

Max Rows: 1000

Id	Name
1	Абраменко Іван Іванович
2	Бондарчук Петро Сергійович
3	Власюк Олександр Миколайович
4	Гончаренко Остп Володимирович
5	Дорошенко Пилип Захарович
6	Жовнірук Сергій Сергійович
7	Заяць Василь Вікторович
8	Єрмак Микола Аркадійович
9	Іщенко Тарас Степанович
NULL	Йосенко Степан Фомович
NULL	NULL

Рисунок 2.15 – Вигляд заповненої таблиці Table

.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ДЛЯ ПРОСУВАННЯ ПРОДАЖУ АВТОМОБІЛІВ ЗАСОБАМИ C#

3.1 Структура та інтерфейс програмно-апаратного комплексу

Програмно-апаратний комплекс, розроблений у кваліфікаційній роботі, реалізований у вигляді проекту CarSalesPromotion засобами C#WinForms з використанням фреймворку .NET Framework 4.7.2. Цей проект складається зі семи форм. Перша форма проекту зображена на рисунку 3.1. Вона забезпечує взаємозв'язок з рештою форм проекту, на яких реалізовані окремі програми комплексу. Призначення кожної форми проекту наведені у таблиці 3.1.

Завдання, які має вирішити програмно-апаратний комплекс, полягають у реалізації низки заходів, що забезпечують більш ефективну продажу салоном автомобілів. Відповідно, кожна форма проекту реалізує один із таких заходів. Комплекс може бути використаний як клієнтами салону під час вибору потрібного їм автомобіля, так і його співробітниками для взаємодії з клієнтами.

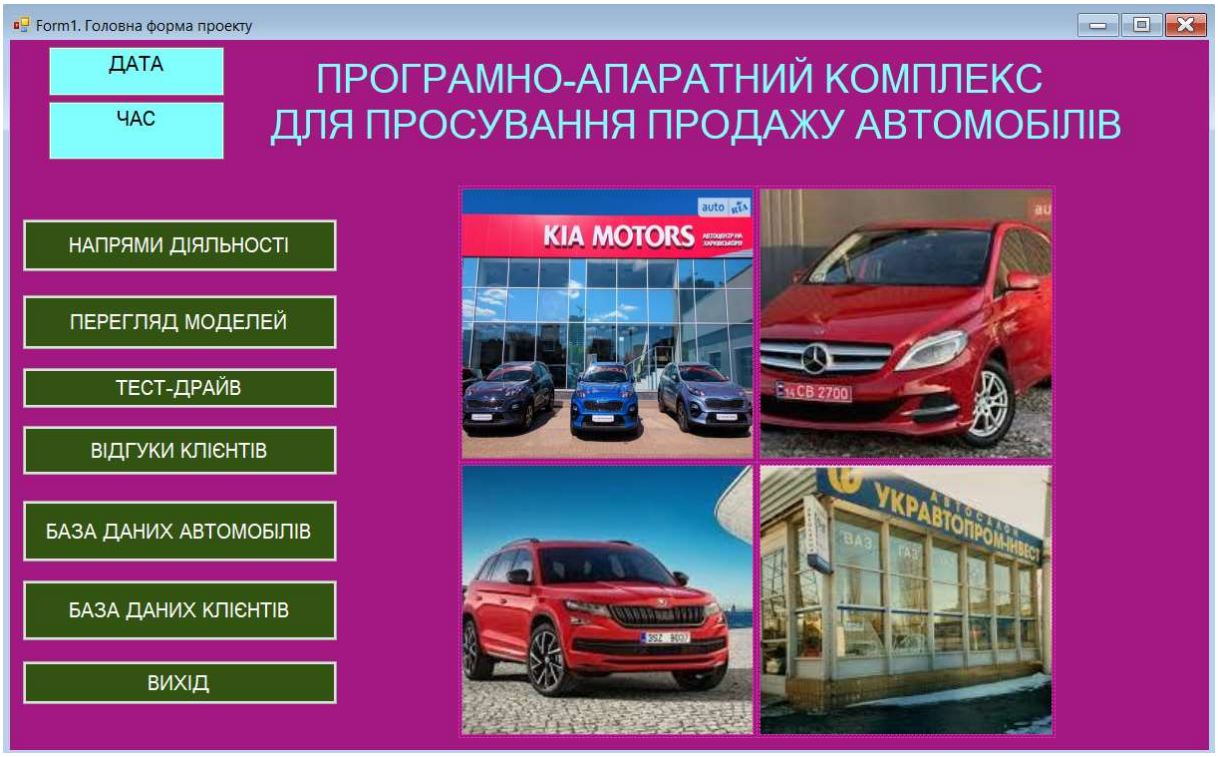

Рисунок 3.1 – Головна форма Form1 проекту

Таблиця 3.1 – Призначення форм програмно-апаратного комплексу

№ п/п	Назва форми	Призначення форми
1	Form1	Інтерфейс програмного комплексу
2	Form2	Напрями діяльності салону з продажу автомобілів
3	Form3	Перегляд марок та моделей автомобілів
4	Form4	Тест-драйв автомобілів
5	Form5	Відгуки клієнтів салону з продажу автомобілів
6	Form6	База даних автомобілів
7	Form7	База даних клієнтів салону

На формі Form1 розміщено ряд компонент, необхідних для забезпечення відповідного її функціоналу. Склад компонент форми Form1 та їх призначення наведено у таблиці 3.2.

Таблиця 3.2 – Склад та призначення компонент форми Form1

№ п/п	Назва компонента	Призначення компонента
1	button1	Перехід на форму «Form2. Напрями діяльності салону з продажу автомобілів»
2	button2	Перехід на форму «Form3. Перегляд марок та моделей автомобілів»
3	button3	Перехід на форму «Form4. Тест-драйв автомобілів»
4	button4	Перехід на форму «Form5. Відгуки клієнтів салону з продажу автомобілів»
5	button5	Перехід на форму «Form6. База даних автомобілів»
6	button6	Перехід на форму «Form7. База даних клієнтів салону»
7	button7	Вихід з програми
8	label1	Надпис «ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС»
9	label2	Надпис «ДЛЯ ПРОСУВАННЯ ПРОДАЖУ АВТОМОБІЛІВ»

Продовження таблиці 3.2

10	textBox1	Відображення дати у форматі «dd:MM:yyyy»
11	textBox2	Відображення часу у форматі «hh:mm:ss»
12	tableLayoutPanel1	Розміщення чотирьох компонент pictureBox
13	pictureBox1	Відображення фото офісу салону «Motor KIA»
14	pictureBox2	Відображення фото автомобіля марки Mercedes
15	pictureBox3	Відображення фото автомобіля марки Skoda
16	pictureBox4	Відображення фото офісу салону «УкрАвтоПромінвест»
17	timer1	Забезпечення відображення часу у режимі реального часу

Властивості форми Form1, які задаються за допомогою вікна Properties, наведені у таблиці 3.3.

Таблиця 3.3 – Властивості форми Form1

№ з/п	Назва властивості	Значення властивості
1	Name	Form1
2	Enabled	True
3	BackColor	Задаємо за системою RGB у вигляді 163:25:129
4	Size	1250; 800. Розміри можна регулювати
5	Text	Form1. Головна форма проекту

Оскільки програмно-апаратний комплекс складається з багатьох форм, то перш за все потрібно вирішити питання навігації по них. Це завдання вирішується за допомогою командних кнопок button, розташованих ліворуч чи праворуч на кожній формі. Надпис на командній кнопці (властивість Text) підказує на яку саме форму ми хочемо передати управління проектом. Якщо, наприклад, нам потрібно перейти з форми Form1 на форму Form5, яка має назву «Відгуки клієнтів», необхідно на формі Form1 знайти кнопку з надписом

«Відгуки клієнтів» і клацнути по ній. Відповідно на формі Form5 потрібно клацнути по кнопці з текстом «До головної форми», для того щоб повернутися назад на форму Form1. У такий спосіб можна перейти з будь-якої форми проекту на будь-яку іншу форму. Для завершення роботи проекту слід клацнути по кнопці «Вихід», яка передбачена на всіх формах.

Описаний вище функціонал командних кнопок забезпечується відповідним програмуванням обробника події Click кожної з них. Наприклад, щоб перейти з форми Form1 на форму Form5, код обробника події Click кнопки button4 форми Form1 має містити код, наведений на рисунку 3.2. Як бачимо з цього коду, він забезпечує закриття форми Form1 та відкриття форми Form5. Для того, щоб повернутися із форми Form5 назад на форму Form1, код обробника події Click кнопки button1 форми Form5 має містити код, наведений на рисунку 3.3. Код обробника події Click кнопки button2 з надписом «Вихід» наведений на рисунку 3.4. Він забезпечує вихід з проекту з будь-якої форми.

```
private void button4_Click_1(object sender, EventArgs e)
{
    this.Hide();
    Form5 f5 = new Form5();
    f5.Show();
}
```

Рисунок 3.2 – Код обробника події Click кнопки button4 форми Form1

```
private void button1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 f1 = new Form1();
    f1.Show();
}
```

Рисунок 3.3 – Код обробника події Click кнопки button1 форми Form2

Властивості компонент, розміщених на формі Form1, наведені далі у таблицях 3.4-3.5.

```
private void button2_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Рисунок 3.4 – Код обробника події Click кнопки button2 з надписом «Вихід»

Таблиця 3.4 – Властивості мітки label1

№ з/п	Назва властивості	Значення властивості
1	Name	label1
2	AutoSize	False
3	Dock	None
4	BackColor	163;25;129
5	Font	Arial Rounder MT Bold 24
6	ForeColor	128; 255; 255
7	Text	ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС
8	TextAllign	MiddleCenter

Таблиця 3.5 – Властивості текстового вікна textBox1

№ з/п	Назва властивості	Значення властивості
1	Name	textBox1
2	Dock	None
3	BackColor	128;255;255
4	Font	Arial Rounder MT Bold 24
5	ForeColor	Widows Text
6	Size	180;50
7	Text	ДАТА
8	TextAllign	MiddleCenter
9	WordWrap	True

Значення цих властивостей ми встановлюємо під час створення відповідних компонент за допомогою вікна Properties (табл. 3.6-3.8).

Таблиця 3.6 – Властивості панелі tableLayoutPanel1

№ з/п	Назва властивості	Значення властивості
1	Name	tableLayoutPanel1
2	AutoSize	False
3	Dock	None
4	BackColor	163;25;129
5	Size	600;550

Таблиця 3.7 – Властивості вікна pictureBox1

№ з/п	Назва властивості	Значення властивості
1	Name	pictureBox1
2	Anchor	Top; Left
3	Dock	Fill
4	Image	SystemDrawing (виклик вікна)

Таблиця 3.8 – Властивості таймера timer1

№ з/п	Назва властивості	Значення властивості
1	Name	timer1
2	Enabled	True
3	Interval	1000

Звичайно, клієнт салону з продажу автомобілів цінує свій час. Тому на першій формі передбачаємо два текстових вікна textBox1 та textBox2 для динамічного відображення у них поточної дати та поточного часу. Це потребує створення відповідних динамічних об’єктів date1 та date2 класу DateTime

(рис. 3.5). Далі ці об’єкти викликаються під час завантаження форми Form1 (рис. 3.6). Спочатку за допомогою методу DateTimeToday() визначається поточна дата, яка відобразиться у тестовому вікні textBox1.Text у форматі dd:MM:yyyy, тобто день:місяць:рік. Далі за допомогою методу DateTimeNow() визначається поточний час, який відобразиться у тестовому вікні textBox2.Text у форматі hh:mm:ss, тобто години:хвилини:секунди.

```
public partial class Form1 : Form
{
    DateTime date1 = new DateTime();
    DateTime date2 = new DateTime();
```

Рисунок 3.5 – Оголошення динамічних об’єктів date1 та date2 класу DateTime

```
public Form1()
{
    InitializeComponent();

    date1 = DateTime.Today;
    textBox1.Text = date1.ToString("dd:MM:yyyy");

    date2 = DateTime.Now;
    textBox2.Text = date2.ToString("hh:mm:ss");
```

Рисунок 3.6 – Ініціалізація динамічних об’єктів date1 та date2 класу DateTime

Однак ці значення дати і часу будуть вірні лише для моменту завантаження форми. Для того, щоб значення дати і часу оновлювалися щосекунди, передбачаємо використання компонента timer1. Відповідно, під час ініціалізації форми Form1 робимо доступним компонент таймер timer1 (рис. 3.7), надаючи його властивості timer1.Enabled значення true. Властивості timer1.Interval присвоюємо значення 1000, що відповідає значенню часу величиною 1 секунда. Далі двічі клацаємо по піктограмі timer1, і програмуємо обробник події tick таймера timer1 (рис. 3.8).

Крім описаних вище компонент форми Form1, передбачаємо на ній і деякі рекламні компоненти. З цією метою перетягуємо на форму панель tableLayoutPanel1, у кожній із чотирьох клітин якої розташовуємо компоненти pictureBox, за допомогою яких відображаємо фотографії офісів двох салонів та

двох моделей автомобілів, якими торгують ці салони. Просте споглядання цих фотографій має за мету мотивувати клієнта обов'язково здійснити покупку автомобіля. У результаті всіх дій після запуску проекту з'явиться перша форма, яка матиме вигляд, зображений на рисунку 3.9.

```
public Form1()
{
    InitializeComponent();

    date1 = DateTime.Today;
    textBox1.Text = date1.ToString("dd:MM:yyyy");

    date2 = DateTime.Now;
    textBox2.Text = date2.ToString("hh:mm:ss");

    timer1.Enabled = true;
    timer1.Interval = 1000; // 1 sec

}
```

Рисунок 3.7 – Код ініціалізації об'єкта timer1 під час завантаження Form1

```
private void timer1_Tick(object sender, EventArgs e)
{
    date2 = DateTime.Now;
    textBox2.Text = date2.ToString("hh:mm:ss");
}
```

Рисунок 3.8 – Код щосекундного оновлення поточного часу

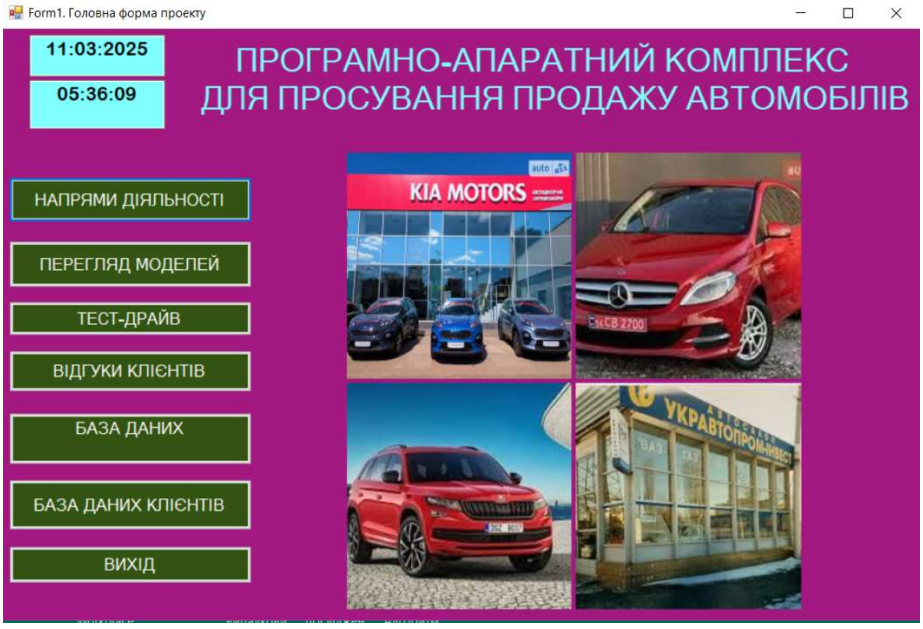

Рисунок 3.9 – Вигляд форми Form1 після запуску проекту

3.2 Напрями діяльності салону з продажу автомобілів

На відміну від першої форми, яка відповідає за інтерфейс програмно–апаратного комплексу, кожна наступна форма відповідає за один із способів просування продажу автомобілів. Форма Form2 (рис. 3.10) призначена для відображення списку напрямів діяльності салону.

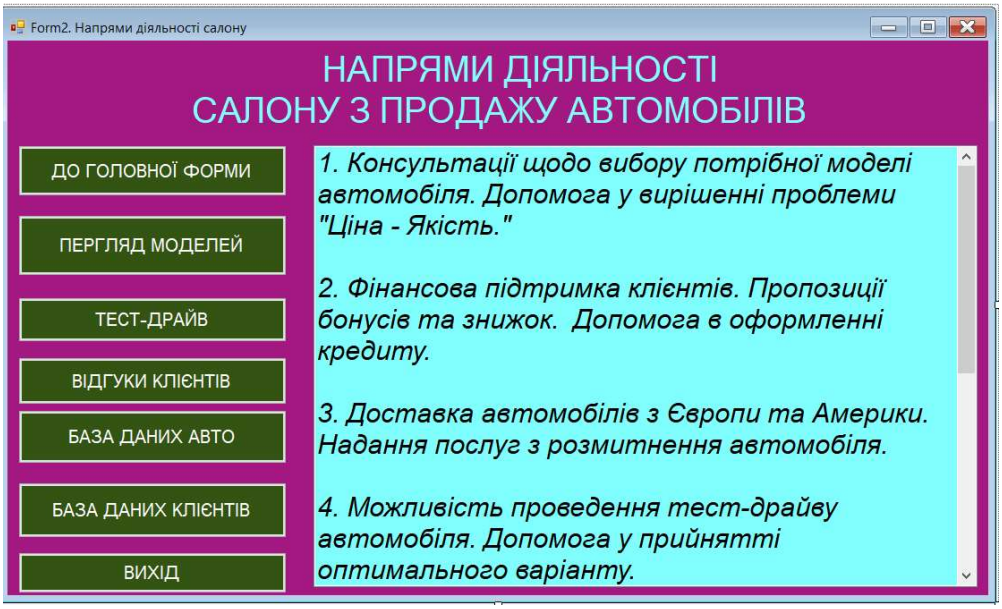

Рисунок 3.10 – Початковий вигляд форми Form2. Напрями діяльності салону

Призначення компонент форми Form2 показані в таблиці 3.9. Для відображення всього списку напрямів діяльності салону використаний компонент listBox1. Його властивості наведені у таблиці 3.10.

Таблиця 3.9 – Призначення компонент форми Form2

№ з/п	Ім’я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button1 – button7	Button	Перехід на інші форми проекту
2	label1 – label2	Label	Відображення надписів
3	listBox1	ListBox	Відображення всього списку напрямів діяльності салону

Таблиця 3.10 – Властивості компонента listBox1

№ п/п	Назва властивості	Значення властивості
1	Name	listBox1
2	BackColor	128; 255; 255
3	Font	Arial; 18pt; style = Italic
4	ForeColor	Windows Text
5	Items	Collection (Виклик допоміжного вікна)

Насамперед створюємо текстовий файл, у якому у вигляді таблиці 3.11 показаний повний список напрямів діяльності салону.

Таблиця 3.11 – Повний список напрямів діяльності салону

№ п/п	Опис напрямів діяльності салону
1	Консультації щодо вибору потрібної моделі автомобіля. Допомога у вирішенні проблеми «Ціна – Якість».
2	Фінансова підтримка клієнтів. Пропозиції бонусів та знижок. Допомога в оформленні кредиту.
3	Доставка автомобілів з Європи та Америки. Надання послуг з розмитнення автомобіля.
4	Можливість проведення тест-драйву автомобіля. Допомога у прийнятті оптимального варіанту.
5	Продаж автомобіля під ключ. Швидке та якісне оформлення потрібних документів.
6	Доступ до бази даних автомобілів.
7	Доступ до бази даних клієнтів
8	Після продажне сервісне обслуговування автомобілів - від техоглядів до капітального ремонту.
9	Гарантії.

Для перенесення інформації з таблиці 3.11 у компонент listBox1 потрібно скопіювати один рядок тексту з таблиці 3.11, клацнути по компоненту listBox1, щоб виділити його, активувати властивість Items, викликати вікно String Collection Editor (рис. 3.11), вставити у це вікно скопійований текст з таблиці 3.11, відредагувати його і натиснути кнопку Ok. Повторивши ці дії стільки разів, скільки є стрічок тексту у таблиці 3.11, і запустивши проект на виконання, отримаємо форму Form2, як показано на рисунку 3.12.

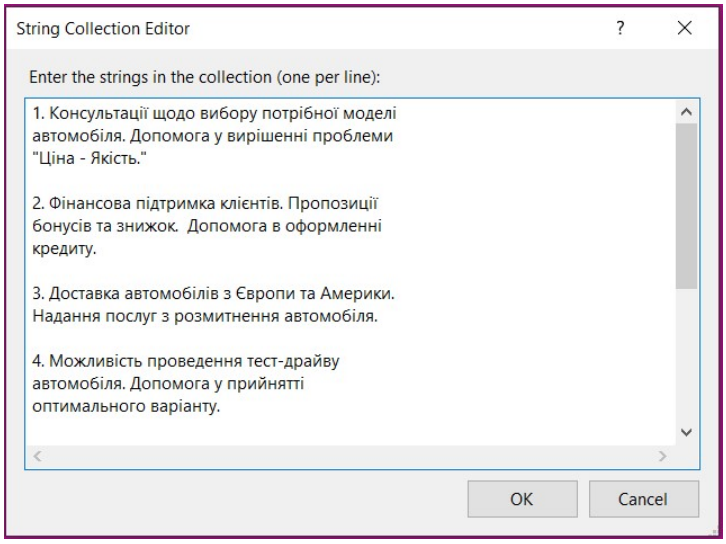

Рисунок 3.11 – Вікно редактора тексту

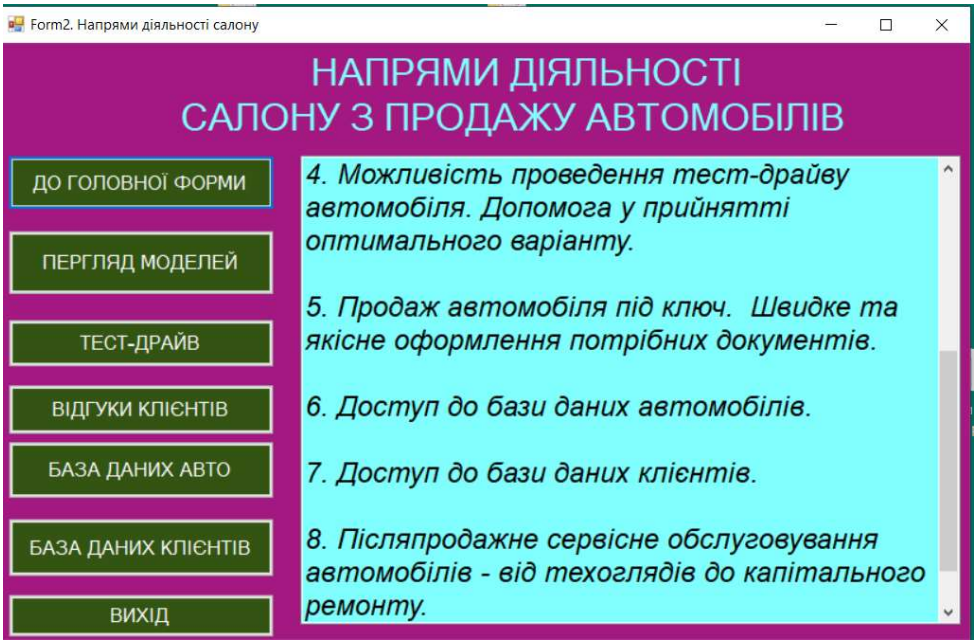

Рисунок 3.12 – Вигляд форми Form2 після її активізації

3.3 Перегляд марок та моделей автомобілів

Форма Form3 (рис. 3.13) дозволяє клієнтам переглядати фотографії автомобілів різних марок та моделей, щоб полегшити свідомий вибір автомобіля потрібної моделі. Достатньо клацнути по відповідній радіо кнопці, як на формі з'явиться фотографія вибраного автомобіля.

Рисунок 3.13 – Вигляд форми Form3. Перегляд моделей автомобілів

Призначення компонент форми Form3 показані в таблиці 3.12. Для забезпечення вибору одного автомобіля з багатьох використані компоненти groupBox та radiobutton. Їх властивості наведені у таблицях 3.13 та 3.14. Властивості компонента pictureBox1 наведені у таблиці 3.15.

Таблиця 3.12 – Призначення компонент форми Form3

№ з/п	Ім'я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button1 – button7	Button	Перехід на інші форми проекту
2	label1 – label6	Label	Відображення надписів
3	groupBox1 – groupBox2	groupBox	Формування групи компонентів
4	radiobutton1 – radiobutton30	radioButton	Вибір одного компонента з групи
5	pictureBox1	pictureBox	Відображення фото автомобіля

Таблиця 3.13 – Властивості компонента groupBox1

№ п/п	Назва властивості	Значення властивості
1	Name	groupBox1
2	BackColor	128; 255; 129
3	Font	Arial Rounded MT; 14pt; style = Bold
4	ForeColor	Control Text
5	Size	170; 410
6	Text	Виберіть модель:

Таблиця 3.14 – Властивості компонента radioButton1

№ п/п	Назва властивості	Значення властивості
1	Name	radioButton1
2	BackColor	128; 255; 129
3	Font	Arial; 14pt
4	ForeColor	Control Text
5	Size	100; 30

Таблиця 3.15 – Властивості компонента pictureBox1

№ п/п	Назва властивості	Значення властивості
1	Name	pictureBox1
2	BackColor	163; 25; 129
3	Image	System Drawing Bitmap
4	Dock	None
5	Size	460; 410

Для того, щоб мати змогу переглядати фотографії автомобілів, їх потрібно перш за все мати. З цією метою у папці нашого проекту CarSalesPromotion\CarSalesPromotion створюємо три підлеглі папки BMW, Ford і Skoda, і в кожній з них зберігаємо десять файлів з назвами p1.jpg, p2.jpg і т.д. з фотографіями автомобілів відповідних марок (рис. 3.14). Самі ж фотографії знаходимо в Інтернеті, використовуючи його пошукові ресурси за ключовими словами, які в даному випадку можуть бути, як от: «фото автомобіля Skoda Octavia».

Рисунок 3.14 – Фотографії автомобілів марки Skoda

Отже, нехай нам потрібно відобразити фотографію автомобіля Skoda Octavia у вікні pictureBox1. Для цього під час виконання проекту на формі Form3 потрібно натиснути радіо кнопку radioButton1. З’явиться фото автомобіля, а властивість Text компонента label6 прийме значення «ЦЕ МОДЕЛЬ SKODA OCTAVIA». Такий функціонал ми забезпечимо, якщо запрограмуємо відповідним чином обробник події radioButton1_CheckedChanged (рис. 3.15).

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    pictureBox1.Image = new Bitmap(@"C:\Users\Desktop\Бакалаври 2024 2025\
    A Serpukhov Bohdan 2025\CarSalesPromotion\CarSalesPromotion\images\Skoda\p1.jpg");
    label6.Text = "ЦЕ МОДЕЛЬ SKODA OCTAVIA";
}
```

Рисунок 3.15 – Код обробника події radioButton1_CheckedChanged

У результаті усіх попередніх дій, після запуску на виконання форми Form3 вона виглядатиме так, як показано на рисунку 3.16.

Рисунок 3.16 – Вигляд форми Form3 після запуску проекту

3.4 Тест-драйв автомобілів

Форма Form4 (рис. 3.17) надає клієнтам можливість переглядати тест-

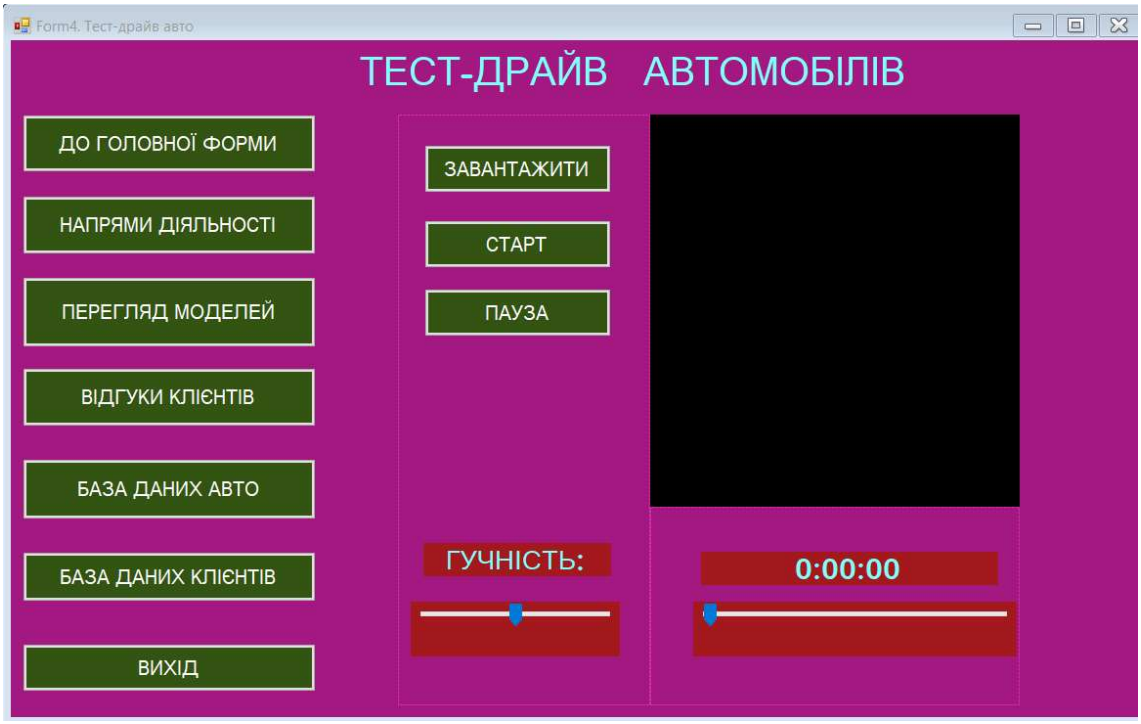

Рисунок 3.17 – Початковий вигляд форми Form4

драйв автомобілів різних марок та моделей, що є потужним рекламним фактором під час купівлі автомобіля. Компоненти, які розміщені на формі Form4, їх категорія та призначення, наведені у таблиці 3.17. Властивості деяких компонент наведені у таблицях 3.18-3.21.

Таблиця 3.17 – Призначення компонент форми Form4

№ з/п	Ім'я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button1 – button10	button	Перехід на інші форми проекту або виконання певних дій
2	label1 – label4	label	Відображення надписів
3	panel1 – panel3	panel	Розміщення інших компонентів
4	Windows Media Player1	Windows Media Player	Перегляд (програвання) відео
5	openFileDialogs1	openFileDialogs	Виклик вікна провідника файлів
6	trackBar1 – trackBar2	trackBar	Регулювання гучності або визначення положення кадра відео, який відображається в даний момент
7	timer1	timer	Щосекундне відображення положення повзуна, що визначає положення кадра відео, який відображається в даний момент

Таблиця 3.18 – Властивості компонента panel1

№ п/п	Назва властивості	Значення властивості
1	Name	panel1
2	AutoSize	None
3	BackColor	163; 25; 229
4	Dock	None
5	Size	710; 600
6	Text	Виберіть модель:

Таблиця 3.19 – Властивості компонента openFileDialogs1

№ п/п	Назва властивості	Значення властивості
1	Name	openFileDialogs1
2	AutoUpgradeEnable	True
3	CheckFileExists	True
4	CheckPathExists	True

Таблиця 3.20 – Властивості компонента Windows Media Player1

№ п/п	Назва властивості	Значення властивості
1	Name	WMP1
2	Dock	Fill
3	Size	450; 400

Таблиця 3.21 – Властивості компонента trackBar1

№ п/п	Назва властивості	Значення властивості
1	Name	trackBar1
2	BackColor	163; 25; 29
3	Dock	None
4	TickStyle	None
5	Size	210; 60

Як бачимо з описаного вище, для розміщення компонент відео програвача нами використані три панелі panel1, panel2 та panel3, які зафарбовані кольором 163; 25; 229 за схемою RGB. Першою розміщуємо панель panel1, а вже на ній розміщуємо панелі panel2, panel3 та відео плеєр Windows Media Player1. На панелі panel2 розміщуємо командні кнопки для управління програвачем та

регулятор гучності, а на панелі panel3 розміщуємо регулятор положення повзуна, що визначає положення кадра відео, який відображається в даний момент часу. Властивість TickStyle обох регуляторів задаємо рівною None, аби не відображати лінію з поділками внизу повзуна.

Компонент Windows Media Player1 встановлюється наступним чином. Клацаємо правою кнопкою миші по середині вікна інструментів Toolbox (не по заголовку, а саме по середині) і вибираємо вкладку Choose Items. У вікні Choose ToolBoxItems, що з'являється, вибираємо вкладку COM Components. У списку компонент, що відкривається, знаходимо компонент Windows Media Player, і напроти його ставимо «галочку». Далі перетягуємо цей компонент на форму і створюємо об'єкт Windows Media Player1, який перейменовуємо на WMP1. Цим самим ми створили інструмент для програвання відео файлів.

Для того, щоб мати змогу програвати відео тест-драйву автомобіля, це відео потрібно перш за все мати. З цією метою у пошуковій системі Google завантажуємо текст «тест-драйв авто українською мовою», і зі запропонованого списку вибираємо файл «Kia Sportage. Тест-драйв українською» (рис. 3.18). Запускаємо його на виконання і копіюємо URL-адресу (рис. 3.19).

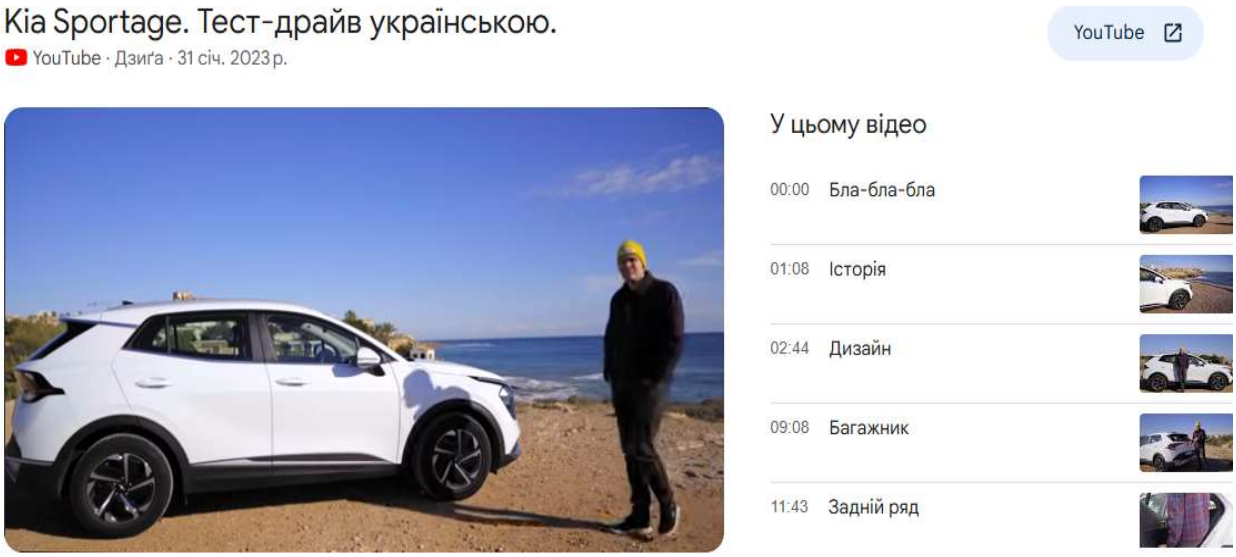

Рисунок 3.18 – Відео Kia Sportage. Тест-драйв українською

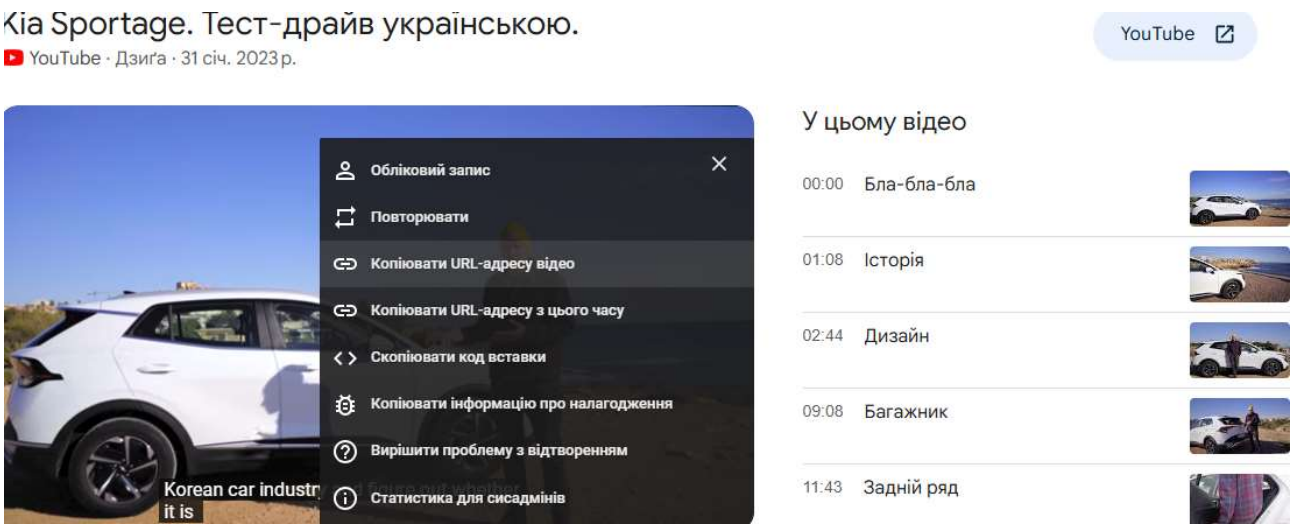

Рисунок 3.19 – Копіювання URL-адреси відео файлу Kia Sportage

Далі у пошуковій системі Google завантажуємо текст «скачати відео з youtube», і зі запропонованого списку сайтів вибираємо сайт ssyoutube.com (рис. 3.20).

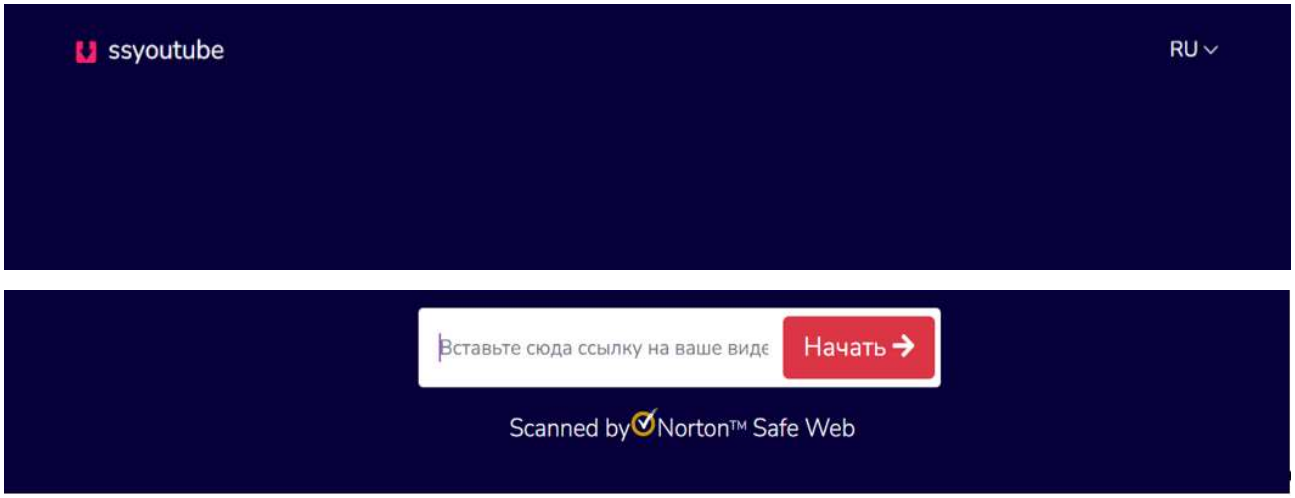

Рисунок 3.20 – Перша сторінка сайту ssyoutube.com

Завантажуємо у текстове поле сайту скопійоване раніше значення URL-адреси відео файлу Kia Sportag і, натискуємо на кнопку Download напроти поля 360.mp4 (рис. 3.21).

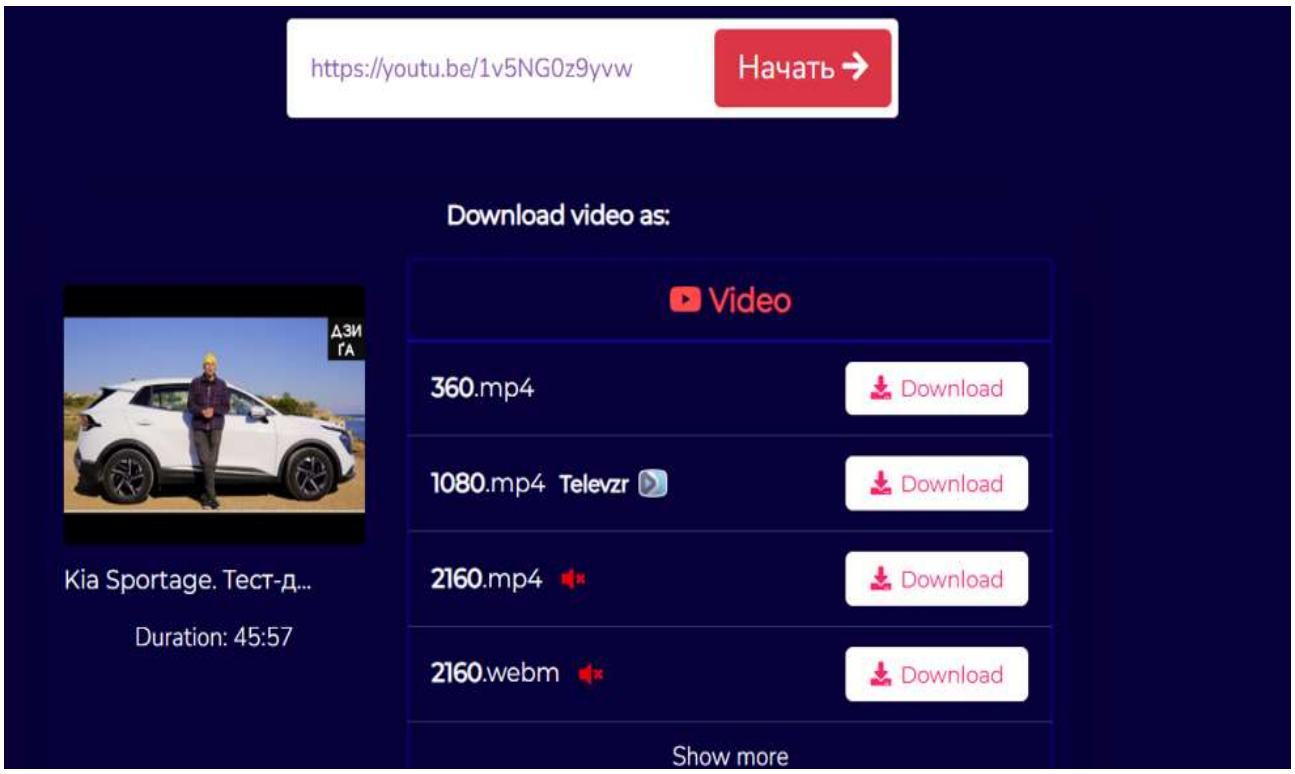

Рисунок 3.21 – Скачування відео файлу

Щоб отримати скопійований відео файл, клацаємо правою кнопкою миші по кнопці завантаження файлу у правому верхньому куті вікна сайту (рис. 3.22), копіюємо «відео файл Kia Sportage. Тест-драйв українською.mp4» і вставляємо його у папку CarSalesPromotion\CarSalesPromotion\Videos нашого проекту (рис. 3.23).

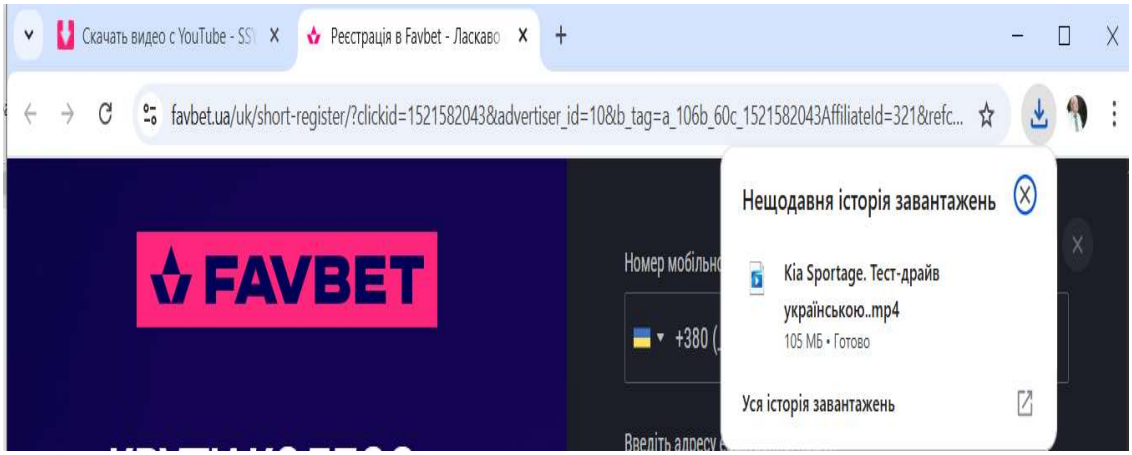

Рисунок 3.22 – Копіювання відео файлу Kia Sportage. Тест-драйв українською.mp4

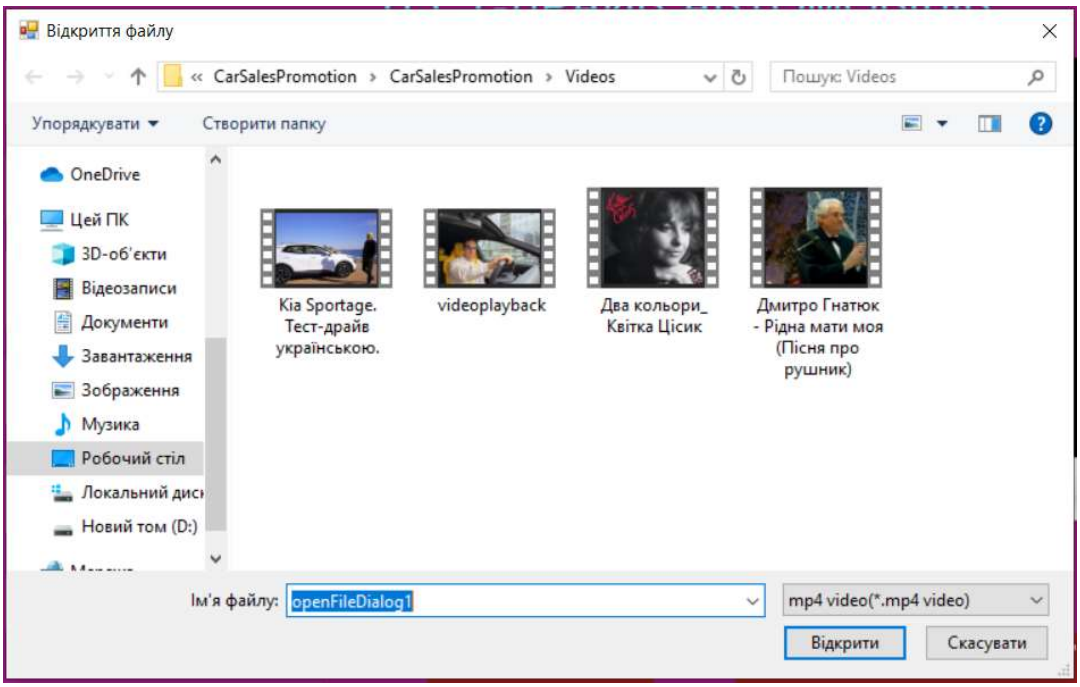

Рисунок 3.23 – Вміст папки CarSalesPromotion\CarSalesPromotion\Videos

Тепер перейходимо до програмування компонент форми Form4. Код обробника події Click кнопки button8 (ЗАВАНТАЖИТИ) наведено на рисунку 3.24. Він формує значення змінної filename, яка містить шлях до відео файлу. Це забезпечує його зчитування і завантаження у медіа плеєр. Щоб полегшити вибір потрібного відео файлу за допомогою компоненти openFileDialog1, змінюємо код ініціалізації форми Form4 (рис. 3.25), додавши у нього метод Filter, який просіює всі файли, залишаючи для перегляду лише ті, що мають перелічені формати. Код обробника події Click кнопки button9 (СТАРТ) наведено на рисунку 3.26, а код обробника події Click кнопки button10 (ПАУЗА) наведено на рисунку 3.27.

```
private void button8_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
    {
        return;
    }
    String filename = openFileDialog1.FileName;
    WMP1.URL = filename;
}
```

Рисунок 3.24 – Код обробника події Click кнопки button8 (ЗАВАНТАЖИТИ)

```
public Form4()
{
    InitializeComponent();
    openFileDialog1.Filter = "mp4 video(*.mp4 video)|*.mp4|MP3 file(*.mp3|"
    "*.mp3|All files(*.*) | *.*";
}
```

Рисунок 3.25 – Код ініціалізації форми Form4

```
private void button9_Click(object sender, EventArgs e)
{
    WMP1.Ctlcontrols.play();
}
```

Рисунок 3.26 – Код обробника події Click кнопки button9 (СТАРТ)

```
private void button10_Click(object sender, EventArgs e)
{
    WMP1.Ctlcontrols.pause();
}
```

Рисунок 3.27 – Код обробника події Click кнопки button10 (ПАУЗА)

Код обробника події ValueChanged компоненти trackBar1 показано на рисунку 3.28.

```
private void trackBar1_ValueChanged(object sender, EventArgs e)
{
    WMP1.settings.volume = trackBar1.Value;
}
```

Рисунок 3.28 – Код обробника події ValueChanged компоненти trackBar1

Код обробника події Scroll компоненти trackBar2 показано на рисунку 3.29.

```
private void trackBar2_Scroll(object sender, EventArgs e)
{
    WMP1.Ctlcontrols.currentPosition = trackBar2.Value;
}
```

Рисунок 3.29 – Код обробника події Scroll компоненти trackBar2

Код обробника події PlayStateChange об’єкта WMP1 показано на рисунку 3.30.

```
private void WMP1_PlayStateChange(object sender,
    AxWMPLib._WMPOCXEvents_PlayStateChangeEvent e)
{
    timer1.Enabled = true;
    timer1.Interval = 1000; // 1 sec
}
```

Рисунок 3.30 – Код обробника події PlayStateChange об’єкта WMP1

Код обробника події Tick таймера timer1 показано на рисунку 3.31.

```
private void timer1_Tick(object sender, EventArgs e)
{
    trackBar2.Maximum = Convert.ToInt32(WMP1.currentMedia.duration);
    trackBar2.Value = Convert.ToInt32(WMP1.Ctlcontrols.currentPosition);

    if (WMP1 != null)
    {
        int s = (int)WMP1.Ctlcontrols.currentPosition;
        int h = s / 3600;
        int m = (s - (h * 3600)) / 60;
        s = s - (h * 3600 + m * 60);
        label4.Text = String.Format("{0:D}:{1:D2}:{2:D2}", h, m, s);

    }

}
```

Рисунок 3.31 – Код обробника події Tick таймера timer1

Вигляд форми Form4 після запуску проекту на виконання подано на рисунку 3.32.

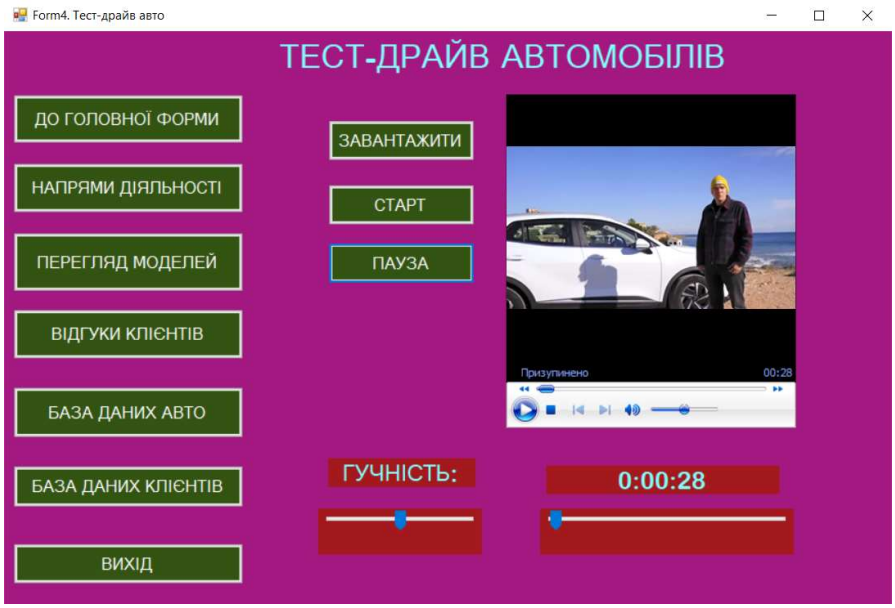

Рисунок 3.32 – Вигляд форми Form4 після запуску проекту на виконання

3.5 Відгуки клієнтів салону

Форма Form5 (рис. 3.33) надає клієнтам можливість переглядати відгуки клієнтів салону стосовно якості їх обслуговування або додати свій відгук, що, зазвичай, позитивно впливає на підвищення культури обслуговування і сприяє росту об'ємів продажу автомобілів. Компоненти, які розміщені на формі Form5, їх категорія та призначення, наведені у таблиці 3.21. Відразу зазначимо, що згодом довелося суттєво доповнити склад цих компонентів.

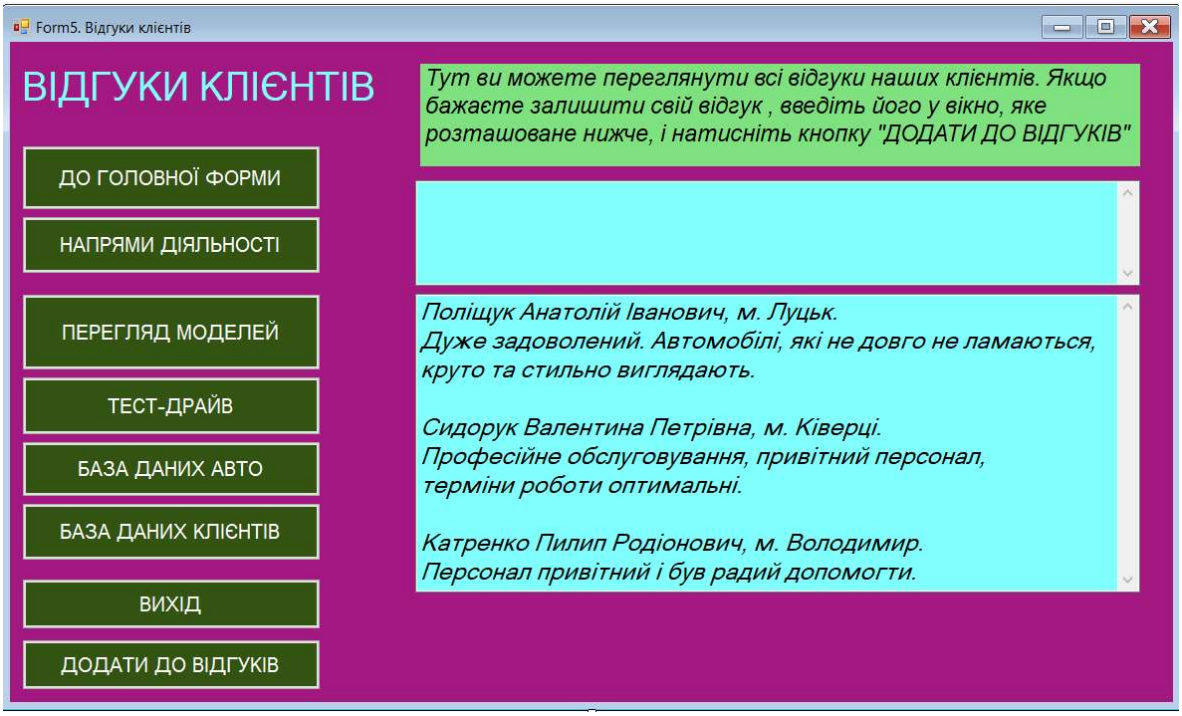

Рисунок 3.33 – Вигляд форми Form6 тесту на базі компонента RadioButton

Призначення компонент форми Form4 показано у таблиці 3.22.

Таблиця 3.22 – Призначення компонент форми Form4

№ з/п	Ім'я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button1 – button8	button	Перехід на інші форми проекту або виконання певних дій
2	label1 – label3	label	Відображення надписів
3	textBox1, textBox2	textBox	Робота з текстовими даними

Властивості компонента textBox1 показані у таблиці 3.23.

Таблиця 3.23 – Властивості компонента textBox1

№ п/п	Назва властивості	Значення властивості
1	Name	textBox1
2	BackColor	128; 255; 255
3	Font	Arial; 14pt; style = Italic
4	ForeColor	Windows Text

Код обробника події Click кнопки button9 (ДОДАТИ ДО ВІДГУКІВ) містить таку команду `textBox2.Text += «\r\n» + textBox1.Text;`. Запустивши форму Form5 на виконання, ми вводимо відгук у поле об’єкта textBox1 і клацаємо по кнопці button9 (ДОДАТИ ДО ВІДГУКІВ). У полі об’єкта textBox2 відбувається додавання поточного відгуку до тих, що вже на момент створення форми вже були введені у цей об’єкт (рис. 3.34).

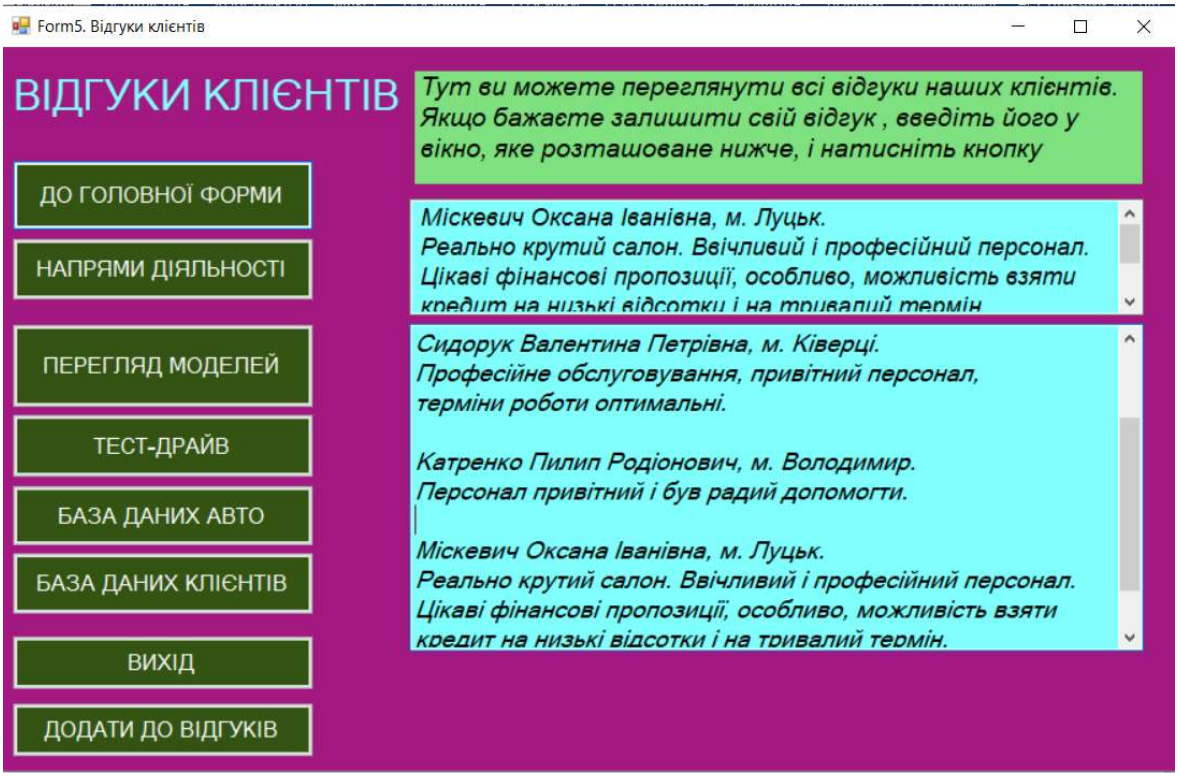

Рисунок 3.34 – Вигляд форми Form4 тесту після запуску проекту

Але, якщо вийти з цієї форми, а потім знову зайти в неї, побачимо, що поточний відгук не додається до цієї форми. Досягти цього ефекту можна, підключивши до цієї форми базу даних, яка зберігатиме всі відгуки. Про роботу з базами даних докладно зупинимося у наступному параграфі. Остаточний вигляд форми Form5 з під'єднаною базою даних, яка зберігає відгуки клієнтів, після запуску на виконання показано на рисунку 3.35.

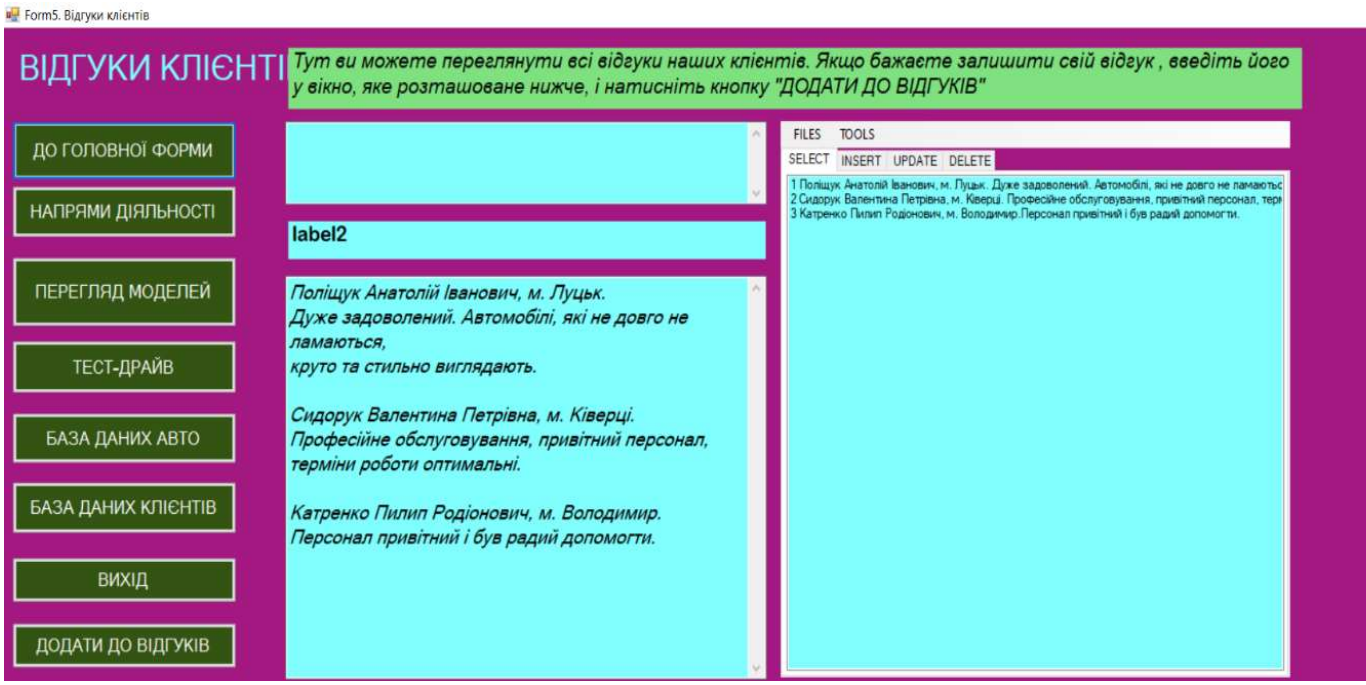

Рисунок 3.35 – Остаточний вигляд форми Form5 після запуску її на виконання

3.6 База даних автомобілів

Форма Form6 (рис. 3.36) надає клієнтам доступ до бази даних автомобілів, якими торгує салон, що дозволяє клієнтам ознайомитися з важливою для них інформацією. Компоненти, які розміщені на формі Form6, їх категорія та призначення, наведені у таблиці 3.24. Зауважимо, що в таблиці 3.24 наведені тільки ті компоненти, які ми бачимо перед запуском форми Form6 на виконання. Однак форма Form6 оперує зі значно більшою кількістю компонент, які розміщені на чотирьох сторінках tabPages об'єкта tabControl1, які ми розглянемо пізніше.

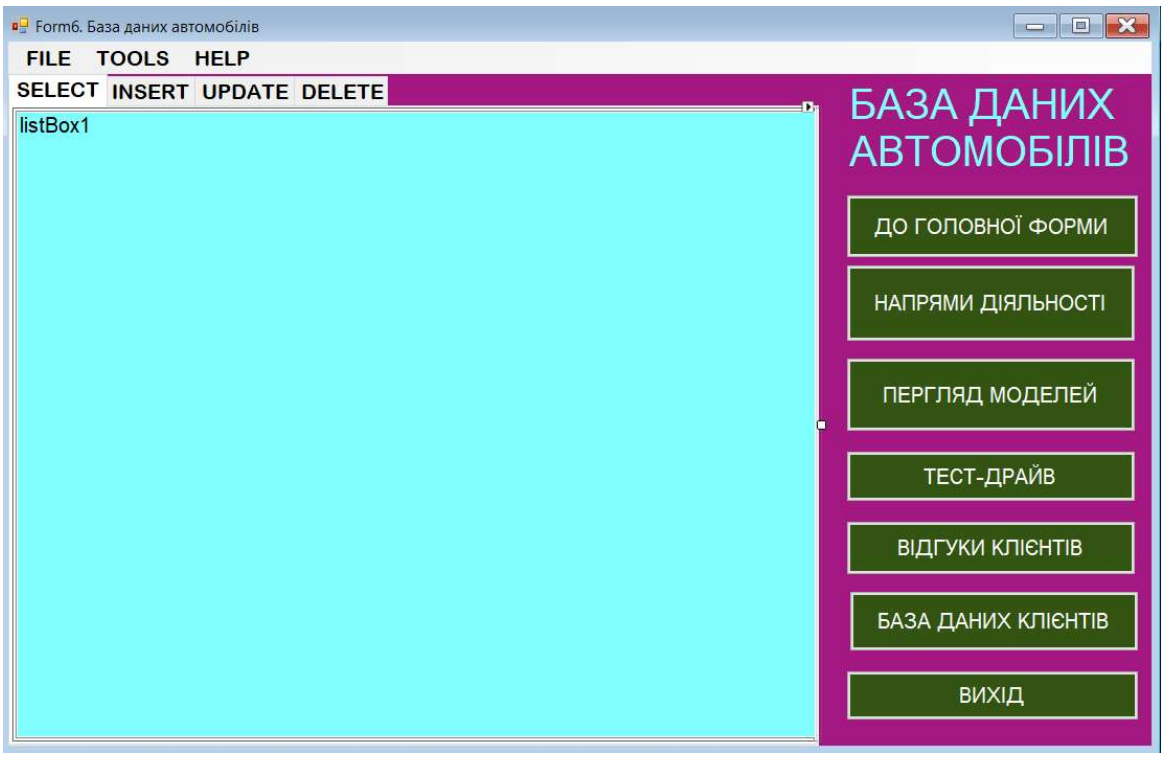

Рисунок 3.36 – Початковий вигляд форми Form6

Таблиця 3.24 – Призначення компонент форми Form4

№ з/п	Ім’я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button1 – button7	button	Перехід на інші форми проекту або виконання певних дій
2	label1 – label2	label	Відображення надписів
3	menuStrip1	menuStrip	Створення меню команд
4	tabControl1	tabControl	Створення підлеглих сторінок бази даних

Властивості компонента menuStrip1 показані у таблиці 3.25.

Таблиця 3.25 – Властивості компонента menuStrip1

№ п/п	Назва властивості	Значення властивості
1	Name	menuStrip1
2	AutoSize	True
3	Items	Collection (FILE, TOOLS, HELP)

Властивості компонента tabControl1 показані у таблиці 3.26.

Таблиця 3.26 – Властивості компонента tabControl1

№ п/п	Назва властивості	Значення властивості
1	Name	tabControl1
2	Dock	Left
3	TabPages	Collection (SELECT, INSERT, UPDATE, DELETE)
4	Size	780; 640

Розробку форми починаємо зі створення бази даних автомобілів. Клацнувши правою кнопкою миші по імені проекту CarSalesPromotion у вікні Solution Explorer, виконуємо команду Add/Components… У вікні Add New Item, вибираємо компонент Service-based Database, залишаємо стандартне ім’я файла бази даних Database1.mdf і клацаємо по кнопці Add. У браузері проектів з’явиться ім’я файла бази даних Database1.mdf.

У вікні Solution Explorer двічі клацаємо по імені Database1.mdf. У вікні Server Explorer з’являється структура порожньої БД (рис. 3.37).

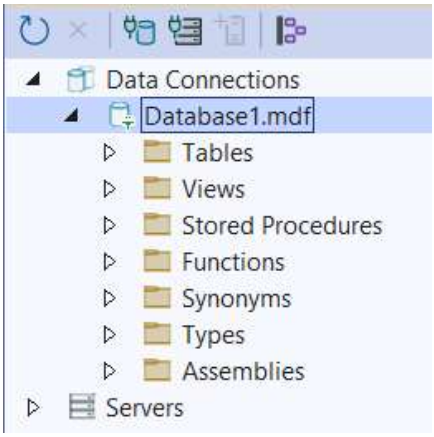

Рисунок 3.37 – Структура створеної БД Database1

Клацаємо правою кнопкою миші по вкладці Tables, і переходимо на вкладку Add New Table. Ім’я нашої таблиці залишимо стандартним, тобто Table. Виконуємо команду Save All і завершуємо процес створення першої таблиці БД.

Оновлюємо вміст БД і викликаємо таблицю Table, двічі клацнувши по ній, для створення її структури та заповнення її конкретними даними (рис. 3.38).

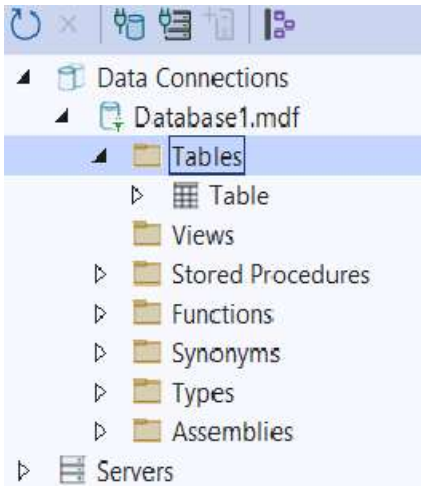

Рисунок 3.38 – Вибір таблиці Table БД Database1

Наша БД має такі поля: model, year, mileage, price, engine, transmission. Поле Id встановлюємо ключовим, а для решти полів вибираємо тип даних nvarchar(50). База даних автомобілів Database1, заповнена конкретними даними, показана на рисунку 3.39

Max Rows: 1000

Id	model	year	mileage	price	engine
1	Octavia	2020	200000	7000	2.4 дизель
2	Octavia	2022	100000	8000	2.8 бензин
3	Fabia	2016	220000	5000	2.4 дизель
4	Favorit	2018	250000	6000	1.8 бензин
5	Rapid	2022	150000	7000	2.8 дизель
6	Felicia	2023	50000	7500	3.8 бензин
7	Forman	2020	200000	6000	2.8 дизель
8	Fabia	2020	150000	6500	1.8 бензин
9	Octavia	2022	100000	6000	3.2 дизель
10	Fabia	2022	150000	4000	3.8 бензин
11	Octavia	2023	100000	7000	3.6 дизель
12	Octavia	2023	100000	8000	2.8 бензин
13	Fabia	2024	50000	9000	1.4 дизель
14	Fabia	2025	0	12000	1.4 дизель

Рисунок 3.39 – Вміст бази даних Database1

Далі на формі Form6 створюємо меню команд, передбачивши у ньому команди FILE, TOOLS та HELP. Для цього виконуємо команду Toolbox/Menus & Toolbars/MenuStrip, вибираємо компонент MenuStrip і перетягуємо його на форму Form6, щоб створити об’єкт menuStrip1. Це і є меню команд, яке заповнюємо згаданими вище командами. У команді FILE передбачаємо підлеглу команду EXIT, а в команді TOOLS передбачаємо підлеглу команду RENOVATE.

У вікні Toolbox вибираємо вкладку Containers, перетягуємо компонент TabControl на форму Form6 і створюємо об’єкт tabControl1, у якому передбачаємо чотири сторінки для організації виконання різних операцій з нашою базою даних. Властивість Dock цього об’єкта встановлюємо Left, у результаті він буде притиснутий до лівого боку вікна форми. Використовуючи об’єкт tabControl1, встановлюємо на кожній з його сторінок за допомогою вікна Collection… різні значення властивості Text:

- для сторінки tabPages1 Text =SELECT;
- для сторінки tabPages2 Text =INSERT;
- для сторінки tabPages3 Text = UPDATE;
- для сторінки tabPages4 Text = DELETE.

Таким чином, кожна сторінка об’єкта tabControl1 реалізує одну команду для роботи з базою даних.

Переходимо до розробки кожної сторінки об’єкта tabControl1. Першою розробляємо сторінку tabPages1, тобто, команду SELECT, яка виконується під час завантаження форми Form6. На сторінці розміщуємо лише один об’єкт listBox1, за допомогою якого відображуватимуться записи бази даних. Код обробника події Form6_Load (рис. 3.40) забезпечує виведення на екран вмісту бази даних, тобто він реалізує команду SELECT. Зауважимо, що попередньо в класі class Form6 необхідно створити змінну (об’єкт) sqlConnection класу SqlConnection. Цей об’єкт використовується для задання шляху до файлу бази даних, за яким він розташований. Цей шлях задається за допомогою стрічкової змінної connectionString. Її значення можна визначити так. У вікні Server Solution клацаємо по файлу Database1.mdf, далі клацаємо по вкладці Properties, і тоді у

вікні Properties визначаємо значення змінної connectionString.

```
private async void Form6_Load(object sender, EventArgs e)
{
    string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;
          AttachDbFilename=""C:\Users\Desktop\Бакалаври 2024 2025
          \A Serpukhov Bohdan 2025\CarSalesPromotion\CarSalesPromotion\Database1.mdf"";
          Integrated Security=True";
    sqlConnection = new SqlConnection(connectionString);
    await sqlConnection.OpenAsync();
    SqlDataReader sqlReader = null;
    SqlCommand command = new SqlCommand("SELECT * FROM [Table]", sqlConnection);
    try
    {
        sqlReader = await command.ExecuteReaderAsync();
        while (await sqlReader.ReadAsync())
        {
            listBox1.Items.Add(Convert.ToString(sqlReader["Id"]) + " " +
                Convert.ToString(sqlReader["model"]) + " " + Convert.ToString(sqlReader["year"]) + " " +
                Convert.ToString(sqlReader["mileage"]) + " " + Convert.ToString(sqlReader["price"]) +
                " " + Convert.ToString(sqlReader["engine"]) + " " + Convert.ToString(sqlReader["transmission"]));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (sqlReader != null)
            sqlReader.Close();
    }
}
```

Рисунок 3.40 – Код обробника події Form6_Load (команда SELECT)

Другою розробляємо сторінку tabPages2, яка виконується шляхом натискання на кнопку INSERT (рис. 3.41). Склад та призначення компонентів сторінки tabPages2 показано у таблиці 3.27.

Таблиця 3.27 – Призначення компонент сторінки tabPages2

№ з/п	Ім’я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button8	button	Виконання операції вставки
2	label3 – label8	label	Відображення імен полів
3	textBox1 – textBox6	textBox	Введення значень полів
4	label17	label	Повідомлення про помилку

Код обробника події button8_Click (рис. 3.42), яка забезпечує виконання команди вставки нового запису у кінець бази даних.

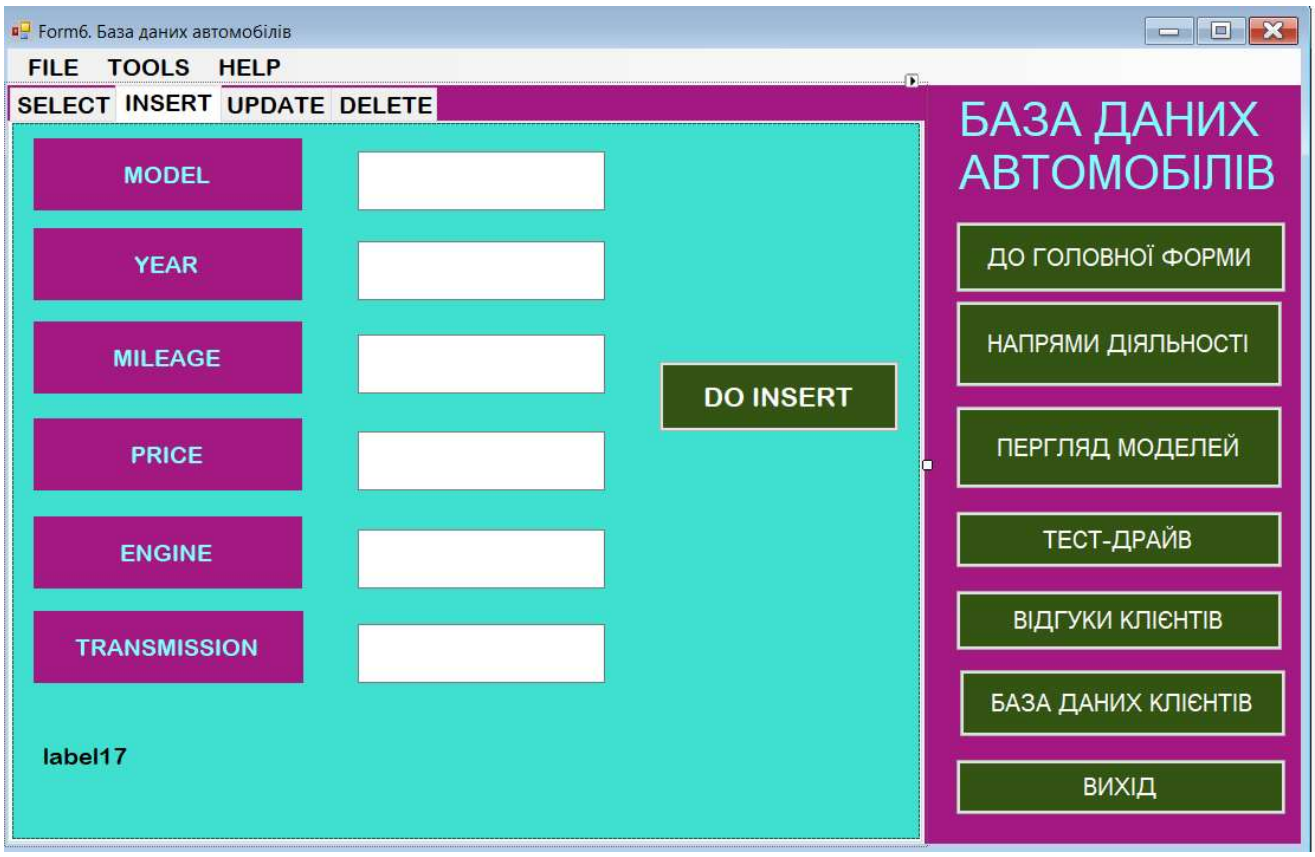

Рисунок 3.41 – Вигляд сторінки tabPages2 (команда INSERT)

```
private async void button8_Click(object sender, EventArgs e)
{
    if (label17.Visible) label7.Visible = false;
    if (!string.IsNullOrEmpty(textBox1.Text) && !string.IsNullOrWhiteSpace(textBox1.Text) &&
        !string.IsNullOrEmpty(textBox2.Text) && !string.IsNullOrWhiteSpace(textBox2.Text) &&
        !string.IsNullOrEmpty(textBox3.Text) && !string.IsNullOrWhiteSpace(textBox3.Text) &&
        !string.IsNullOrEmpty(textBox4.Text) && !string.IsNullOrWhiteSpace(textBox4.Text) &&
        !string.IsNullOrEmpty(textBox5.Text) && !string.IsNullOrWhiteSpace(textBox5.Text) &&
        !string.IsNullOrEmpty(textBox5.Text) && !string.IsNullOrWhiteSpace(textBox5.Text))
    {
        SqlCommand command = new SqlCommand("INSERT INTO [Table] (model, year, mileage, price," +
            " engine, transmission)VALUES(@model, @year, @mileage, @price, @engine, @transmission)",
            sqlConnection);
        command.Parameters.AddWithValue("model", textBox1.Text);
        command.Parameters.AddWithValue("year", textBox2.Text);
        command.Parameters.AddWithValue("mileage", textBox3.Text);
        command.Parameters.AddWithValue("price", textBox4.Text);
        command.Parameters.AddWithValue("engine", textBox5.Text);
        command.Parameters.AddWithValue("transmission", textBox6.Text);
        await command.ExecuteNonQueryAsync();
        MessageBox.Show("Команда INSERT виконана!", "Підтвердження виконання команди INSERT.",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        label17.Visible = true;
        label17.Text = "Ввести поля 'model', 'year', 'mileage', 'price', 'engine' , 'transmission'!";
    }
}
```

Рисунок 3.42 – Код обробника події button8_Click

Третьою розробляємо сторінку tabPages3, яка виконується шляхом натискання на кнопку UPDATE (рис. 3.43). Склад та призначення компонентів сторінки tabPages3 показано у таблиці 3.28.

Таблиця 3.28 – Призначення компонент сторінки tabPages3

№ з/п	Ім’я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button9	button	Виконання операції оновлення
2	label9 – label5	label	Відображення імен полів
3	textBox7 – textBox13	textBox	Введення значень полів
4	label18	label	Повідомлення про помилку

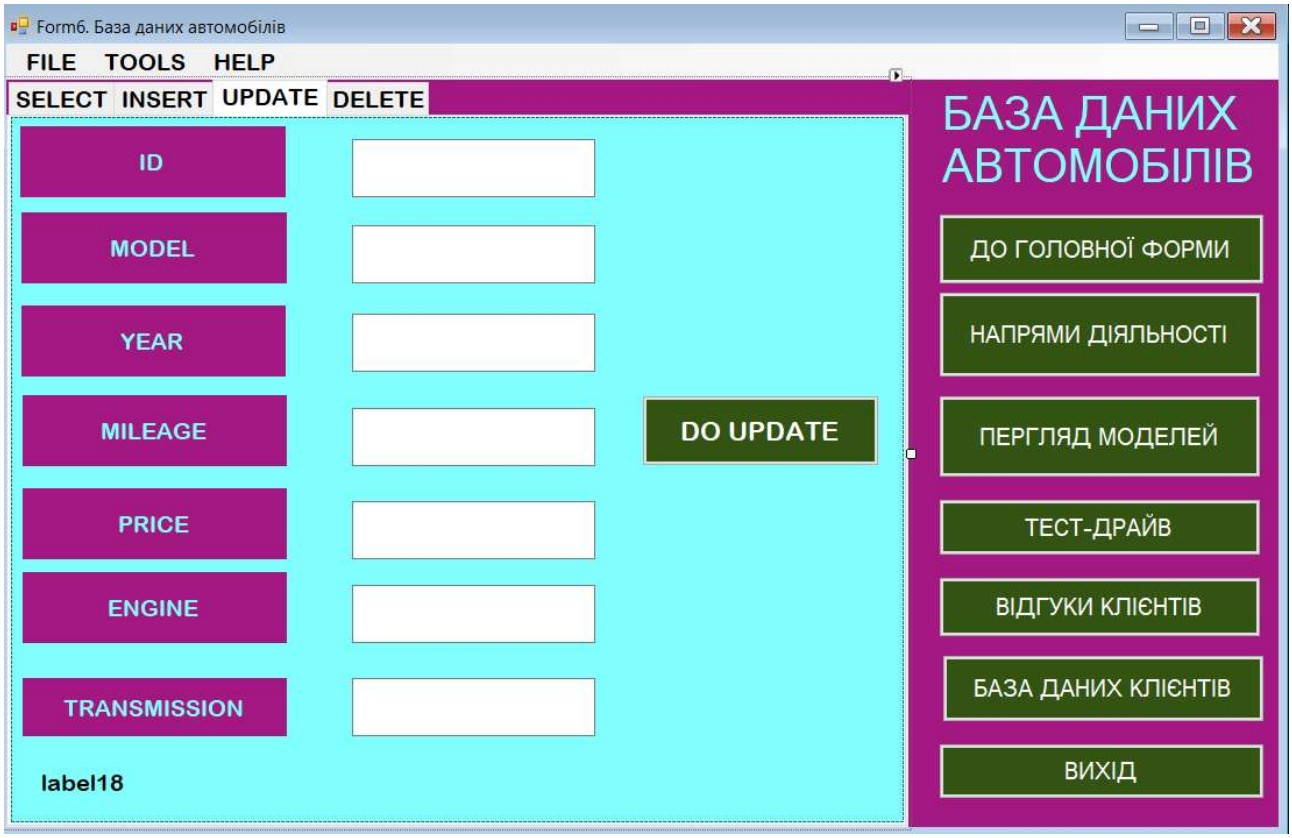

Рисунок 3.43 – Вигляд сторінки tabPages3 (команда UPDATE)

Код обробника події button9_Click, яка забезпечує виконання команди оновлення (редагування) певного запису, значення Id якого ми задаємо, наведено на рисунку 3.44. Якщо одне з полів на сторінці не буде задано, з’явиться повідомлення заповнити всі поля бази даних.

```
private async void button9_Click(object sender, EventArgs e)
{   if (label18.Visible)
        label18.Visible = false;
    if (!string.IsNullOrEmpty(textBox7.Text) && !string.IsNullOrWhiteSpace(textBox7.Text) &&
        !string.IsNullOrEmpty(textBox8.Text) && !string.IsNullOrWhiteSpace(textBox8.Text) &&
        !string.IsNullOrEmpty(textBox9.Text) && !string.IsNullOrWhiteSpace(textBox9.Text) &&
        !string.IsNullOrEmpty(textBox10.Text) && !string.IsNullOrWhiteSpace(textBox10.Text) &&
        !string.IsNullOrEmpty(textBox11.Text) && !string.IsNullOrWhiteSpace(textBox11.Text) &&
        !string.IsNullOrEmpty(textBox12.Text) && !string.IsNullOrWhiteSpace(textBox12.Text) &&
        !string.IsNullOrEmpty(textBox13.Text) && !string.IsNullOrWhiteSpace(textBox13.Text))
    {
        SqlCommand command = new SqlCommand("UPDATE [Table] SET [model] = @model, [year] = @year, " +
            "[mileage] = @mileage, [price] = @price, [engine] = @engine, " +
            "[transmission] = @transmission WHERE [Id] = @Id", sqlConnection);
        command.Parameters.AddWithValue("Id", textBox7.Text);
        command.Parameters.AddWithValue("model", textBox8.Text);
        command.Parameters.AddWithValue("year", textBox9.Text);
        command.Parameters.AddWithValue("mileage", textBox10.Text);
        command.Parameters.AddWithValue("price", textBox11.Text);
        command.Parameters.AddWithValue("engine", textBox12.Text);
        command.Parameters.AddWithValue("transmission", textBox13.Text);
        await command.ExecuteNonQueryAsync();
        MessageBox.Show("Команда UPDATE виконана!", "Підтвердження виконання команди UPDATE.",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else if (!string.IsNullOrEmpty(textBox7.Text) && !string.IsNullOrWhiteSpace(textBox7.Text))
    {   label18.Visible = true;
        label18.Text = "Id має бути заповнений!";
    }
    else
    {   label18.Visible = true;
        label18.Text = "Заповнити поля 'model', 'year', 'mileage', 'price', 'engine','transmission'!";
    }
```

Рисунок 3.44 – Код обробника події button9_Click

Четвертою розробляємо сторінку tabPages4, яка виконується шляхом натискання на кнопку DELETE (рис. 3.45).

Склад та призначення компонентів сторінки tabPages3 показано у таблиці 3.29.

Таблиця 3.29 – Призначення компонент сторінки tabPages4

№ з/п	Ім’я компонента або їх групи	Категорія компонента	Призначення компонента або групи компонентів
1	button10	button	Виконання операції вилучення
2	label6	label	Відображення імені поля
3	textBox14	textBox	Введення значення поля
4	label19	label	Повідомлення про помилку

Код обробника події button10_Click (виконання команди вилучення певного запису, значення Id якого ми задаємо), наведено на рисунку 3.46.

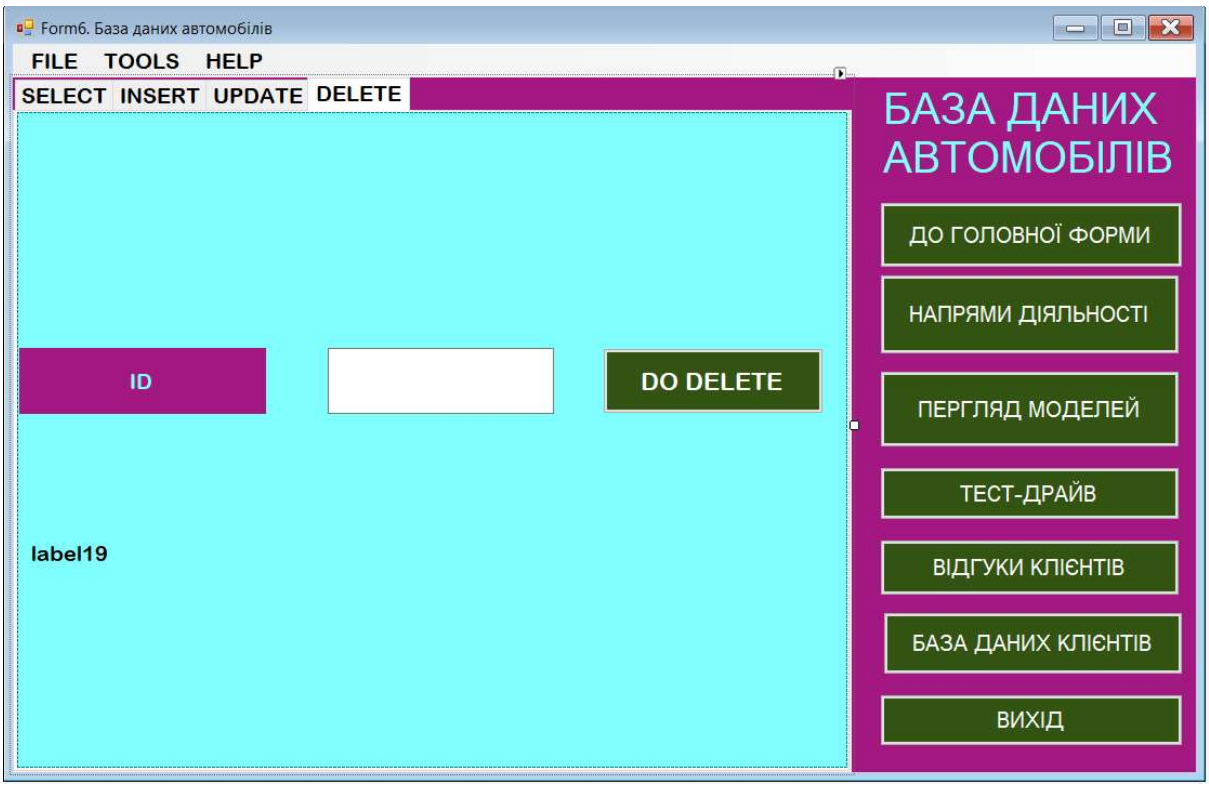

Рисунок 3.45 – Вигляд сторінки tabPages4 (команда DELETE)

```
private async void button10_Click(object sender, EventArgs e)
{
    if (label19.Visible)
        label19.Visible = false;
    if (!string.IsNullOrEmpty(textBox14.Text) && !string.IsNullOrWhiteSpace(textBox14.Text))
    {
        SqlCommand command = new SqlCommand("DELETE FROM [Table] WHERE [Id] = @Id", sqlConnection);

        command.Parameters.AddWithValue("Id", textBox14.Text);

        await command.ExecuteNonQueryAsync();
        MessageBox.Show("Команда DELETE виконана!", "Підтвердження виконання команди DELETE.",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        label19.Visible = true;
        label19.Text = "Поле Id має бути заповнене!";
    }
}
```

Рисунок 3.46 – Код обробника події button10_Click (команда DELETE)

Після внесення змін у хоча би в одне поле запису бази даних, цю базу потрібно оновити. Це досягається виконанням команди RENOVATE нашого меню команд. Код обробника події Click відповідного пункту меню наведено на рисунку 3.47.

Остаточний вигляд форми Form6 після запуску неї на виконання показано

на рисунку 3.48. Повний код форми Form6 наведено у додатку А.

```
private async void uPDATEToolStripMenuItem_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();

    SqlDataReader sqlReader = null;
    SqlCommand command = new SqlCommand("SELECT * FROM [Table]", sqlConnection);
    try
    {
        sqlReader = await command.ExecuteReaderAsync();
        while (await sqlReader.ReadAsync())
        {
            listBox1.Items.Add(Convert.ToString(sqlReader["Id"]) + " " +
                Convert.ToString(sqlReader["model"]) + " " + Convert.ToString(sqlReader["year"]) +
                " " + Convert.ToString(sqlReader["mileage"]) + " " + Convert.ToString(sqlReader["price"]) +
                " " + Convert.ToString(sqlReader["engine"]) + " " + Convert.ToString(sqlReader["transmission"]));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (sqlReader != null)
            sqlReader.Close();
    }
}
```

Рисунок 3.47 – Код обробника події Click кнопки RENOVATE

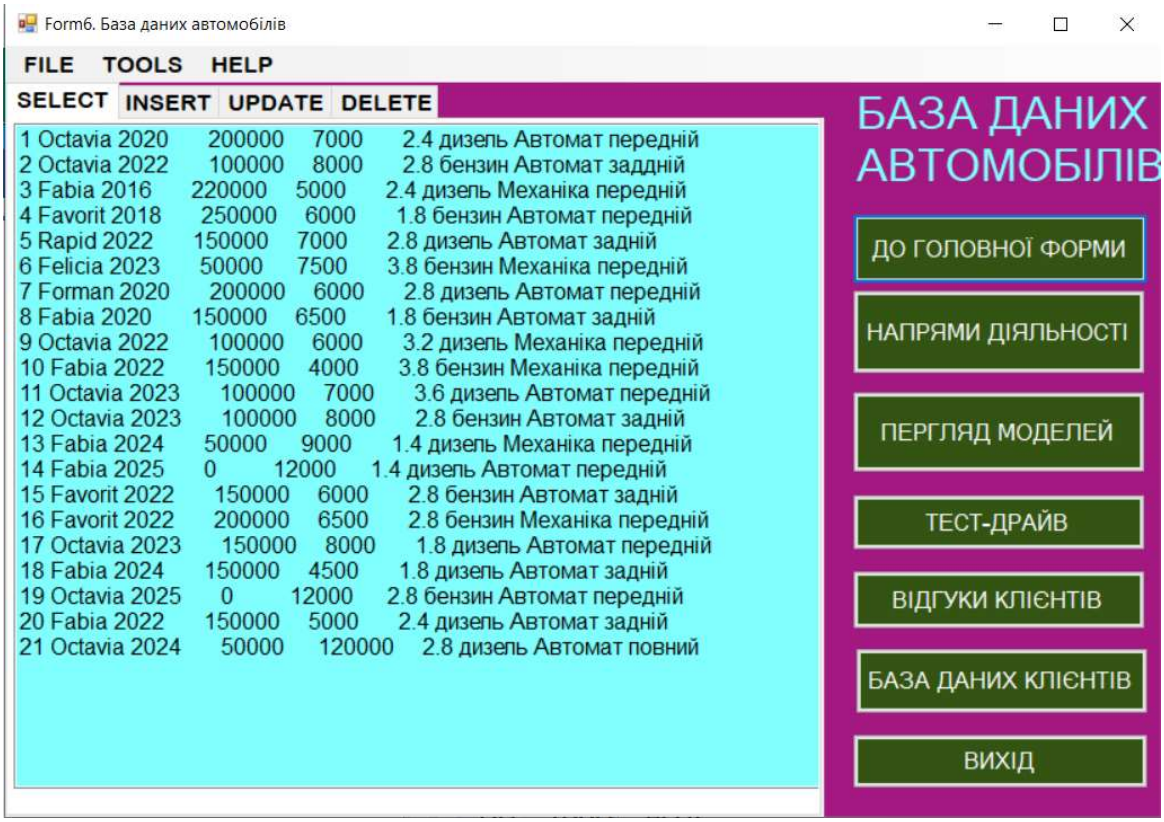

Рисунок 3.48 – Вигляд форми Form6 після запуску неї на виконання

3.7 База даних клієнтів салону

Форма Form7 (рис. 3.49-3.50) надає працівникам салону доступ до бази даних клієнтів, що дозволяє ділитися з ними важливою інформацією. База даних клієнтів розроблена за тією ж технологією, що й база даних клієнтів Повний код форми Form6 наведено у додатку Б.

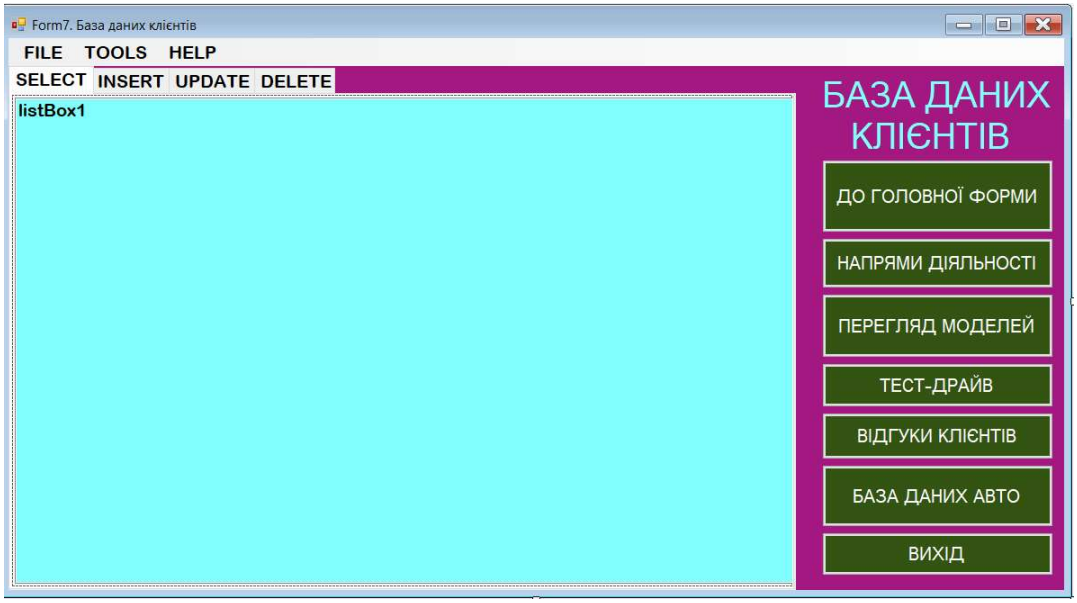

Рисунок 3.49 – Початковий вигляд форми Form7

Рисунок 3.50 – Вигляд форми Form7 після запуску неї на виконання

3.8 Вимоги до апаратної частини програмно-апаратного комплексу

Програмно-апаратний комплекс розроблений для реалізації його на персональних комп’ютерах з такими основними характеристиками:

– процесор Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz на базі архітектури x64;

– оперативно-запам’ятовуючий пристрій ОЗП 8,00 ГБ (доступно для використання: 7,86 ГБ);

– операційна система Windows 10 Pro, версія 22H2, 64-розрядна операційна система;

– ввід за допомогою пера та сенсорний ввід на дисплеї недоступні.

ВИСНОВКИ

Проведені у кваліфікаційній роботі бакалавра дослідження та розроблений у ній програмно-апаратний комлекс засвідчили актуальність розв’язаної у роботі проблеми. Встановлено, що просування продаж автомобілів за допомогою програмно-апаратного комплекс є на сьогодні актуальним завданням, що має чітке практичне спрямування.

У кваліфікаційнй роботі бакалавра:

– виконано огляд середовищ та мов програмування і обгрунтовано їх вибір для розробки програмно-апаратного комплексу з просування продаж автомобілів;

– дано детальну характеристику інструментів для розробки програмно-апаратного комплексу у вигляді проекту C# WinForms;

– розроблено форму «Напрями діяльності салону»;

– розроблено форму «Перегляд марок та моделей автомобілів»;

– розроблено форму «Тест-драйв автомобілів»;

– розроблено форму «Відгуки клієнтів салону»;

– розроблено форму «База даних автомобілів»;

– розроблено форму «База даних клієнтів салону».

Усі форми комплексу належним чином розроблені, відлагоджені та протестовані.

Розроблений у кваліфікаційній роботі програмно-апаратний комплекс (проект) може бути використаний салоном з продажу автомобілів збільшення обсягів продаж та студентами університетів під час вивчення технологій розробки подібних комлексів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bjarne Stroustrup. Principles and Practice Using C++. Addison-Wesley: Pearson. 2023. 320 p. URL: https://www.stroustrup.com/PPP3.html (дата звернення: 29.03.2025).

2. Josh Lospinoso. C++ CRASH COURSE. San Francisko: NoStarch Press, 2022. 794 p. URL: https://surli.cc/rkzcyd (дата звернення: 29.03.2025).

3. Візуальне середовище програмування. URL: https://surl.li/mlmhgp (дата звернення: 29.03.2025).

4. Microsoft Visual Studio. URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio (дата звернення: 29.03.2025).

5. Коноваленко І.В., Марущак П.О. Платформа .NET та мова програмування C# 8.0. Тернопіль: ФОП Паляниця В.А., 2020. 320 с.

6. .NET | Build. URL: https://dotnet.microsoft.com/en-us/ (дата звернення: 29.03.2025).

7. Що таке .NET? https://abitap.com/1-1/ (дата звернення: 29.03.2025).

8. Переваги та недоліки .NET: швидкий розвиток, велика поширеність і середні зарплати. URL: https://dou.ua/lenta/articles/pros-and-cons-of-dotnet/ (дата звернення: 29.03.2025).

9. ASP.NET core blazor. URL: https://surl.li/fhzorn (дата звернення: 29.03.2025).

10. The clean coder blog. URL: https://surl.li/trnzdj (дата звернення: 29.03.2025).

11. Pcharm. Вікіпедія. URL: https://uk.wikipedia.org/wiki/PyCharm (дата звернення: 29.03.2025).

12. Eclipse. URL: https://uk.wikipedia.org/wiki/Eclipse (дата звернення: 29.03.2025).

13. Мова програмування. URL: https://uk.wikipedia.org/wiki/Мова_програмування (дата звернення: 29.03.2025).

14. C Sharp. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 29.03.2025).

15. Що таке C#? Кому підходить програмування на сі шарп? Beetroot academy. https://surl.li/dxxcfb (дата звернення: 29.03.2025).

16. 9 причин вивчати мову C# FoxmindEd. FoxmindEd. URL: https://foxminded.ua/ru/9-prichin-vivchiti-movu-c (дата звернення: 29.03.2025).

17. Янковський А. C# book. URL: https://surl.li/mnzhgt (дата звернення: 29.03.2025).

18. ISO/IEC JTC1 SC22 WG21 N4860. Programming Languages C++. ISO/IEC. 2020. 1841 p. URL: https://isocpp.org/files/papers/N4860.pdf (дата звернення: 29.03.2025).

19. Michael D. Exercises for Programming in C++. 2021. URL: https://www.ece.uvic.ca/~frodo/cppbook/downloads/exercises_for_programming_in_cpp-2021-04-01.pdf (дата звернення: 29.03.2025).

20. Changkun Ou. Modern C++ Tutorial: C++11/14/17/20 On the Fly. 2023. 92 p. URL: https://changkun.de/modern-cpp/pdf/modern-cpp-tutorial-en-us.pdf (дата звернення: 29.03.2025).

21. Marius Bancila. Modern C++ Programming Cookbook. URL: https://studylib.net/doc/25977911/modern-c---programming-cookbook---master-c---core-languag (дата звернення: 29.03.2025).

22. Дейтел Х.М., Дейтел П.Дж. Як програмувати на C++. К: ООО «Біном-Пресс», 2023. 1456 с. URL: https://studylib.net/doc/25977911/modern-c---programming-cookbook---master-c---core-languag (дата звернення: 29.03.2025).

23. Пех П.А. Програмування. Конспект лекцій. Луцьк: ЛНТУ, 2020. 324 с. URL: https://lib.lntu.edu.ua/uk/147258369/3772 (дата звернення: 29.03.2025).

24. Пех П.А. Програмування. Методичні вказівки до самостійної роботи. Луцьк: ЛНТУ, 2021. 208 с. URL: https://lib.lntu.edu.ua/uk/147258369/10323 (дата звернення: 29.03.2025).

25. Python (programming language). URL: https://en.wikipedia.org/wiki/Python_(programming_language) (дата звернення: 29.03.2025).

26. The Python Tutorial. Python documentation. URL: https://docs.python.org/3/tutorial/ (дата звернення: 29.03.2025).

27. Jotheph D. Dooth. Database Design. URL: https://dokumen.pub/database-design-succinctly-9781642002232.html (дата звернення: 29.03.2025).

28. Павловський В.І., Петрашенко А.В., Побєда Д.В. Бази даних та засоби управління., К.: КПІ ім. Ігоря Сікорського, 2021. 112 с.

ДОДАТКИ

Додаток А

Повний код форми F6

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace CarSalesPromotion
{
    public partial class Form6 : Form
    {
        SqlConnection sqlConnection;

        public Form6()
        {
            InitializeComponent();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form3 f3 = new Form3();
            f3.Show();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form7 f7 = new Form7();
            f7.Show();

        }

        private void button5_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form5 f5 = new Form5();
            f5.Show();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 f4 = new Form4();
            f4.Show();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button2_Click(object sender, EventArgs e)
```

```
        {
            this.Hide();
            Form2 f2 = new Form2();
            f2.Show();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 f1 = new Form1();
            f1.Show();
        }

        private async void Form6_Load(object sender, EventArgs e)
        {
            string connectionString = @»Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=««C:\Users\Петро\Desktop\Бакалаври 2024
2025\A Serpukhov Bohdan
2025\CarSalesPromotion\CarSalesPromotion\Database1.mdf»«;Integrated Security=True»;
            sqlConnection = new SqlConnection(connectionString);
            await sqlConnection.OpenAsync();
            SqlDataReader sqlReader = null;
            SqlCommand command = new SqlCommand(«SELECT * FROM [Table]»,
sqlConnection);
            try
            {
                sqlReader = await command.ExecuteReaderAsync();
                while (await sqlReader.ReadAsync())
                {
                    listBox1.Items.Add(Convert.ToString(sqlReader[«Id»]) + « « +
Convert.ToString(sqlReader[«model»]) + « « + Convert.ToString(sqlReader[«year»]) + « «
+ Convert.ToString(sqlReader[«mileage»]) + « « + Convert.ToString(sqlReader[«price»])
+ « « + Convert.ToString(sqlReader[«engine»]) + « « +
Convert.ToString(sqlReader[«transmission»]));
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlReader != null)
                    sqlReader.Close();
            }
        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private async void button8_Click(object sender, EventArgs e)
        {
            if (label17.Visible) label7.Visible = false;
            if (!string.IsNullOrEmpty(textBox1.Text) &&
!string.IsNullOrWhiteSpace(textBox1.Text) &&
                !string.IsNullOrEmpty(textBox2.Text) &&
!string.IsNullOrWhiteSpace(textBox2.Text) &&
                !string.IsNullOrEmpty(textBox3.Text) &&
!string.IsNullOrWhiteSpace(textBox3.Text) &&
                !string.IsNullOrEmpty(textBox4.Text) &&
!string.IsNullOrWhiteSpace(textBox4.Text) &&
                !string.IsNullOrEmpty(textBox5.Text) &&
```

```
!string.IsNullOrWhiteSpace(textBox5.Text) &&
                !string.IsNullOrEmpty(textBox5.Text) &&
!string.IsNullOrWhiteSpace(textBox5.Text))
            {
                SqlCommand command = new SqlCommand(«INSERT INTO [Table] (model, year,
mileage, price, engine, transmission)VALUES(@model, @year, @mileage, @price, @engine,
@transmission)», sqlConnection);
                command.Parameters.AddWithValue(«model», textBox1.Text);
                command.Parameters.AddWithValue(«year», textBox2.Text);
                command.Parameters.AddWithValue(«mileage», textBox3.Text);
                command.Parameters.AddWithValue(«price», textBox4.Text);
                command.Parameters.AddWithValue(«engine», textBox5.Text);
                command.Parameters.AddWithValue(«transmission», textBox6.Text);
                await command.ExecuteNonQueryAsync();
                MessageBox.Show(«Команда INSERT виконана!», «Підтвердження виконання
команди INSERT.», MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                label17.Visible = true;
                label17.Text = «Ввести поля 'model', 'year', 'mileage', 'price',
'engine' , 'transmission'!»;
            }
        }

        private async void button9_Click(object sender, EventArgs e)
        {
            if (label18.Visible)
                label18.Visible = false;
            if (!string.IsNullOrEmpty(textBox7.Text) &&
!string.IsNullOrWhiteSpace(textBox7.Text) && !string.IsNullOrEmpty(textBox8.Text) &&
!string.IsNullOrWhiteSpace(textBox8.Text) && !string.IsNullOrEmpty(textBox9.Text) &&
!string.IsNullOrWhiteSpace(textBox9.Text) && !string.IsNullOrEmpty(textBox10.Text) &&
!string.IsNullOrWhiteSpace(textBox10.Text) && !string.IsNullOrEmpty(textBox11.Text) &&
!string.IsNullOrWhiteSpace(textBox11.Text) && !string.IsNullOrEmpty(textBox12.Text) &&
!string.IsNullOrWhiteSpace(textBox12.Text) && !string.IsNullOrEmpty(textBox13.Text) &&
!string.IsNullOrWhiteSpace(textBox13.Text))
            {
                SqlCommand command = new SqlCommand(«UPDATE [Table] SET [model] =
@model, [year] = @year, [mileage] = @mileage, [price] = @price, [engine] = @engine,
[transmission] = @transmission WHERE [Id] = @Id», sqlConnection);

                command.Parameters.AddWithValue(«Id», textBox7.Text);
                command.Parameters.AddWithValue(«model», textBox8.Text);
                command.Parameters.AddWithValue(«year», textBox9.Text);
                command.Parameters.AddWithValue(«mileage», textBox10.Text);
                command.Parameters.AddWithValue(«price», textBox11.Text);
                command.Parameters.AddWithValue(«engine», textBox12.Text);
                command.Parameters.AddWithValue(«transmission», textBox13.Text);
                await command.ExecuteNonQueryAsync();

                MessageBox.Show(«Команда UPDATE виконана!», «Підтвердження виконання
команди UPDATE.», MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else if (!string.IsNullOrEmpty(textBox7.Text) &&
!string.IsNullOrWhiteSpace(textBox7.Text))
            {
                label18.Visible = true;
                label18.Text = «Id має бути заповнений!»;
            }
            else
            {
                label18.Visible = true;
                label18.Text = «Заповнити поля 'Name', 'GRN', 'USD', 'EUR', 'PLN'!»;
            }
```

```
        }

        private void label18_Click(object sender, EventArgs e)
        {

        }

        private async void button10_Click(object sender, EventArgs e)
        {
            if (label19.Visible)
                label19.Visible = false;
            if (!string.IsNullOrEmpty(textBox14.Text) &&
!string.IsNullOrWhiteSpace(textBox14.Text))
            {
                SqlCommand command = new SqlCommand(«DELETE FROM [Table] WHERE [Id] =
@Id», sqlConnection);

                command.Parameters.AddWithValue(«Id», textBox14.Text);

                await command.ExecuteNonQueryAsync();
                MessageBox.Show(«Команда DELETE виконана!», «Підтвердження виконання
команди DELETE.», MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                label19.Visible = true;
                label19.Text = «Поле Id має бути заповнене!»;
            }

        }

        private async void uPDATEToolStripMenuItem_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();

            SqlDataReader sqlReader = null;
            SqlCommand command = new SqlCommand(«SELECT * FROM [Table]»,
sqlConnection);
            try
            {
                sqlReader = await command.ExecuteReaderAsync();
                while (await sqlReader.ReadAsync())
                {
                    listBox1.Items.Add(Convert.ToString(sqlReader[«Id»]) + « « +
Convert.ToString(sqlReader[«model»]) + « « + Convert.ToString(sqlReader[«year»]) + « «
+ Convert.ToString(sqlReader[«mileage»]) + « « + Convert.ToString(sqlReader[«price»])
+ « « + Convert.ToString(sqlReader[«engine»]) + « « +
Convert.ToString(sqlReader[«transmission»]));
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlReader != null)
                    sqlReader.Close();
            }
        }

        private void eToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            if (sqlConnection != null && sqlConnection.State !=
ConnectionState.Closed)
                sqlConnection.Close();
            Application.Exit();
        }
    }
}
```

Додаток Б

Повний код форми F7

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace CarSalesPromotion
{
    public partial class Form7 : Form
    {
        SqlConnection sqlConnection;
        public Form7()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 f1 = new Form1();
            f1.Show();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private async void Form7_Load(object sender, EventArgs e)
        {
            string connectionString = @»Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=««C:\Users\Петро\Desktop\Бакалаври 2024
2025\A Serpukhov Bohdan
2025\CarSalesPromotion\CarSalesPromotion\Database2.mdf»«;Integrated Security=True»;
            sqlConnection = new SqlConnection(connectionString);
            await sqlConnection.OpenAsync();
            SqlDataReader sqlReader = null;
            SqlCommand command = new SqlCommand(«SELECT * FROM [Table]»,
sqlConnection);
            try
            {
                sqlReader = await command.ExecuteReaderAsync();
```

```
                while (await sqlReader.ReadAsync())
                {
                    listBox1.Items.Add(Convert.ToString(sqlReader[«Id»]) + « « +
Convert.ToString(sqlReader[«name»]) + « « + Convert.ToString(sqlReader[«adres»]) + « «
+ Convert.ToString(sqlReader[«mobile»]) + « « + Convert.ToString(sqlReader[«model»]));
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlReader != null)
                    sqlReader.Close();
            }
        }

        private void label7_Click(object sender, EventArgs e)
        {

        }

        private void tabPage1_Click(object sender, EventArgs e)
        {

        }

        private async void button8_Click_1(object sender, EventArgs e)
        {
            if (label7.Visible) label7.Visible = false;
            if (!string.IsNullOrEmpty(textBox1.Text) &&
!string.IsNullOrWhiteSpace(textBox1.Text) &&
                !string.IsNullOrEmpty(textBox2.Text) &&
!string.IsNullOrWhiteSpace(textBox2.Text) &&
                !string.IsNullOrEmpty(textBox3.Text) &&
!string.IsNullOrWhiteSpace(textBox3.Text) &&
            !string.IsNullOrEmpty(textBox4.Text) &&
!string.IsNullOrWhiteSpace(textBox4.Text))
            {
                SqlCommand command = new SqlCommand(«INSERT INTO [Table] (name, adres,
mobile, model)VALUES(@name, @adres, @mobile, @model)», sqlConnection);
                command.Parameters.AddWithValue(«name», textBox1.Text);
                command.Parameters.AddWithValue(«adres», textBox2.Text);
                command.Parameters.AddWithValue(«mobile», textBox3.Text);
                command.Parameters.AddWithValue(«model», textBox4.Text);
                await command.ExecuteNonQueryAsync();
                MessageBox.Show(«Команда INSERT виконана!», «Підтвердження виконання
команди INSERT.», MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                label7.Visible = true;
                label7.Text = «Ввести поля ’name’, ’adres’, ’mobile’, ’model’!»;
            }
        }

        private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void tabPage3_Click(object sender, EventArgs e)
        {
```

```
        }

        private void label11_Click(object sender, EventArgs e)
        {

        }

        private void textBox6_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox7_TextChanged(object sender, EventArgs e)
        {

        }

        private void label8_Click(object sender, EventArgs e)
        {

        }

        private void label9_Click(object sender, EventArgs e)
        {

        }

        private void label10_Click(object sender, EventArgs e)
        {

        }

        private void textBox8_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox5_TextChanged(object sender, EventArgs e)
        {

        }

        private async void button9_Click(object sender, EventArgs e)
        {
            if (label13.Visible)
                label13.Visible = false;
            if (!string.IsNullOrEmpty(textBox5.Text) &&
!string.IsNullOrWhiteSpace(textBox5.Text) && !string.IsNullOrEmpty(textBox6.Text) &&
!string.IsNullOrWhiteSpace(textBox6.Text) && !string.IsNullOrEmpty(textBox7.Text) &&
!string.IsNullOrWhiteSpace(textBox7.Text) && !string.IsNullOrEmpty(textBox8.Text) &&
!string.IsNullOrWhiteSpace(textBox8.Text) && !string.IsNullOrEmpty(textBox9.Text) &&
!string.IsNullOrWhiteSpace(textBox9.Text))
            {
                SqlCommand command = new SqlCommand(«UPDATE [Table] SET [name] =
@name, [adres] = @adres, [mobile] = @mobile, [model] = @model WHERE [Id] = @Id»,
sqlConnection);

                command.Parameters.AddWithValue(«Id», textBox5.Text);
                command.Parameters.AddWithValue(«name», textBox6.Text);
                command.Parameters.AddWithValue(«adres», textBox7.Text);
                command.Parameters.AddWithValue(«mobile», textBox8.Text);
                command.Parameters.AddWithValue(«model», textBox9.Text);
                await command.ExecuteNonQueryAsync();
```

```
                MessageBox.Show(«Команда UPDATE виконана!», «Підтвердження виконання
команди UPDATE.», MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else if (!string.IsNullOrEmpty(textBox5.Text) &&
!string.IsNullOrWhiteSpace(textBox5.Text))
            {
                label13.Visible = true;
                label13.Text = «Id має бути заповнений!»;
            }
            else
            {
                label13.Visible = true;
                label13.Text = «Заповнити поля 'name', 'adres', 'mobile', 'model'!»;
            }
        }

        private async void button10_Click(object sender, EventArgs e)
        {
            if (label14.Visible)
                label14.Visible = false;
            if (!string.IsNullOrEmpty(textBox14.Text) &&
!string.IsNullOrWhiteSpace(textBox14.Text))
            {
                SqlCommand command = new SqlCommand(«DELETE FROM [Table] WHERE [Id] =
@Id», sqlConnection);

                command.Parameters.AddWithValue(«Id», textBox14.Text);

                await command.ExecuteNonQueryAsync();
                MessageBox.Show(«Команда DELETE виконана!», «Підтвердження виконання
команди DELETE.», MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                label14.Visible = true;
                label14.Text = «Поле Id має бути заповнене!»;
            }
        }

        private async void uPDATEToolStripMenuItem_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();

            SqlDataReader sqlReader = null;
            SqlCommand command = new SqlCommand(«SELECT * FROM [Table]»,
sqlConnection);
            try
            {
                sqlReader = await command.ExecuteReaderAsync();
                while (await sqlReader.ReadAsync())
                {
                    listBox1.Items.Add(Convert.ToString(sqlReader[«Id»]) + « « +
Convert.ToString(sqlReader[«name»]) + « « + Convert.ToString(sqlReader[«adres»]) + « «
+ Convert.ToString(sqlReader[«mobile»]) + « « + Convert.ToString(sqlReader[«model»]));
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlReader != null)
                    sqlReader.Close();
```

```
            }

        }

        private void eXITToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (sqlConnection != null && sqlConnection.State !=
ConnectionState.Closed)
                sqlConnection.Close();
            Application.Exit();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form2 f2 = new Form2();
            f2.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form3 f3 = new Form3();
            f3.Show();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 f4 = new Form4();
            f4.Show();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form5 f5 = new Form5();
            f5.Show();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form6 f6 = new Form6();
            f6.Show();
        }
    }
}
```