

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА
ПОЯСНЮВАЛЬНА ЗАПИСКА
з дисципліни “Об’єктно -орієнтоване програмування”
Автосалон

Керівник , Проф.

Бондарєв В.М.

Студент гр. ПЗПІ 22-1

Джежела Н.Р.

Комісія:

Проф. Бондарєв В.М.,
Ст. викл. Черепанова Ю.Ю.,
Ст. викл. Ляпота В.М.

Харків 2023

Аркуш завдання та календарний план

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра *програмної інженерії*

Рівень вищої освіти *перший (бакалаврський)*

Дисципліна *Об'єктно-орієнтоване програмування*

Спеціальність *121 Інженерія програмного забезпечення*

Освітня програма: *Програмна інженерія*

Курс 1.

Група ПЗПІ-22 -1.

Семестр 2.

ЗАВДАННЯ

на курсовий проект студента

Джежели Нікіти Руслановича

1 Тема проекту:

Автосалон

2 Термін здачі студентом закінченого проекту: ***“5” - червня - 2023 р.***

3 Вихідні дані до проекту:

Специфікація програми, методичні вказівки до виконання курсової роботи

4 Зміст розрахунково-пояснювальної записки:

Вступ, опис вимог, проектування програми, інструкція користувача, висновки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапу	Термін виконання
1	Видача теми, узгодження і затвердження теми	13.04.2023 - 14.04.2023 р.
2	Формулювання вимог до програми	21.04.2023 – 30.04.2023 р.
3	Розробка підсистеми зберігання та пошуку даних.	30.04.2023 – 20.05.2023 р.
4	Розробка функцій вибору користувача та повідомлень	20.05.2023 – 22.05.2023 р.
5	Розробка функцій зберігання та завантаження даних	22.05.2023 – 24.05.2023 р.
6	Тестування і доопрацювання розробленої програмної системи	24.05.2023 – 26.05.2023 р.
7	Оформлення пояснювальної записки, додатків, графічного матеріалу	28.05.2023 – 31.05.2023 р.
8	Захист	01.06.2023 – 05.06.2023 р.

Студент _____

Керівник _____

(Прізвище, Ім'я, По батькові)

« 13 » квітня 2023 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 32 с., 13 рис., 1 додаток, 3 джерела.

ПОКУПЕЦЬ, АВТОСАЛОН, АВТОМОБІЛІ, ООП, .NET, МОВА C#.

Метою роботи є розробка програми «Автосалон», за допомогою якої, користувач матиме змогу продавати та купувати автомобілі.

В результаті розробки отримана програма, що дозволяє обрати поточного користувача, переглянути список наявних автомобілів, кожен з яких має свої характеристики: марка, модель, ціна, рік випуску, стан, об'єм двигуна, тип трансмісії та ім'я продавця. Є можливість фільтрувати за всіма параметрами та переглядати окремі автомобілі, повідомляти власника про намір придбати його автомобіль. Є можливість виставляти свій автомобіль на продаж та переглядати свої повідомлення.

В процесі розробки використано середовища Microsoft Visual Studio 2022, фреймворк Windows Forms, платформи .NET 7.0, мова програмування C#

ЗМІСТ

ВСТУП.....	6
1 Опис вимог.....	7
1.1 Сценарії використання програми.....	7
1.2 Функції програми.....	11
2 Проектування програми.....	14
3 Інструкція користувача.....	17
3.1 Інструкція по розгортанню програми.....	17
3.2 Інструкція по роботі з програмою.....	17
3.3 Інструкція по видаленню програми.....	19
Висновки.....	20
Перелік джерел посилання.....	21
Додаток А.....	22

ВСТУП

Мета роботи : створити зручний застосунок для пошуку та продажу автомобілів, який надаватиме змогу користувачеві знайти необхідний йому транспортний засіб за багатьма критеріями та зв'язатись з його власником для подальшої купівлі, або навпаки, розмістити оголошення про продаж свого автомобілю та мати змогу зв'язатись з покупцями.

Спосіб використання програми поділяється на два випадки: продаж та купівля. У обох випадках спочатку треба обрати ім'я користувача, або продовжити роботу в гостьовому режимі.

При продажу треба вказати необхідні дані про свій автомобіль та очікувати повідомлень.

При купівлі можна скористатися загальним списком для пошуку, або відфільтрувати його за параметрами : бренд, модель, рік (діапазон), ціна (діапазон), переглянути детальну інформацію та зв'язатись з продавцем.

Програма може стаціонарно використовуватись у автосалонах, або виконувати роль маркетплейса для широкого використання по території України. Вона вирішує проблему пошуку автомобілів при купівлі, та спрощує процес пошуку покупців.

1 ОПИС ВИМОГ

1.1 Сценарії використання програми

Сценарій 1. Зміна поточного користувача

Передумова

Користувач запустив програму.

Основний сценарій

1. Користувач натискає кнопку “My profile” на верхній панелі кнопок головної форми.
2. Відкривається додаткова форма на якій відображається юзернейм поточного користувача (якщо змін до цього не було - “default”), текстове поле для вводу даних та кнопка “Set!”.
3. Користувач вводить новий юзернейм та натискає кнопку “Set!”.
4. Програма перевіряє, якщо такий користувач існує, то змінює поточного користувача на вже існуючого, якщо такого користувача не існує, то створює нового, з введенням юзернеймом, та встановлює його.
5. Користувач бачить оновлений юзернейм поточного користувача.

Сценарій 2. Перегляд повідомлень

Передумова

Користувач запустив програму та змінив поточного користувача.

Основний сценарій

1. Користувач натискає кнопку “Notifications” на верхній панелі кнопок головної форми.

2. Відкривається додаткова форма у заголовку якої відображається юзернейм поточного користувача, та список усіх отриманих повідомлень.

3. Користувач бачить свої повідомлення.

Додатковий сценарій

1. Користувач натискає кнопку “Notifications”, але поточний юзер - “default”.

2. Форма “Notifications” не відкривається. Відкривається вікно з помилкою, де зазначено що користувач має змінити поточного користувача, щоб скористатися цією функцією.

Сценарій 3. Фільтрація списку автомобілів

Передумова

Користувач запустив програму.

Основний сценарій

1. Користувач бачить шість підписаних текстових полів для фільтрації (марка, модель, рік(діапазон), ціна(діапазон) та список усіх наявних автомобілів, у списку відображені короткі відомості про конкретний автомобіль: марка, модель, ціна, рік.

2. Користувач заповнює поля.

3. Програма у режимі реального часу фільтрує список за всіма параметрами.

4. Користувач бачить список автомобілів, які відповідають заданим параметрам.

Додатковий сценарій

1. Користувач заповнив поля, але за заданими параметрами не знайдено жодного автомобіля.

2. Користувач бачить пустий список.

Сценарій 4. Перегляд повної інформації про автомобіль

Передумова

Користувач запустив програму.

Основний сценарій

1. Користувач обирає зі списку автомобіль, клікнувши на необхідний мишкою, один раз, після чого натискає кнопку “Get Info”, яка знаходиться на головній формі.

2. Відкривається додаткова форма у заголовку якої відображається коротка інформація про обраний автомобіль. У тілі форми знаходиться текстова інформація про обраний автомобіль : бренд, модель, ціна, рік, стан, об’єм двигуна, тип трансмісії, та ім’я власника.

Сценарій 5. Сповіщення власника про намір придбати автомобіль

Передумова

Користувач запустив програму та змінив поточного користувача, після чого знайшов необхідний йому автомобіль, та відкрив перегляд повної інформації.

Основний сценарій

1. Користувач натискає кнопку ”Send message to owner”.

2. Власник обраного автомобіля отримає повідомлення, в якому буде зазначено юзернейм користувача який бажає придбати автомобіль та коротку інформацію про те, який автомобіль хочуть придбати.

Додатковий сценарій

1. Користувач натискає кнопку "Send message to owner", але поточний юзер - "default".

2. Повідомлення не надсилається. Відкривається вікно з помилкою, де зазначено що користувач має змінити поточного користувача, щоб скористатися цією функцією.

Сценарій 6. Створення нового оголошення про продаж автомобілю

Передумова

Користувач запустив програму та змінив поточного користувача.

Основний сценарій

1. Користувач наводить мишкою на меню "Cars" та натискає кнопку "Sale car" у випадаючому меню.

2. Відкривається форма, в якій 5 текстових полів та 2 списки, кнопка "Sale".

3. Користувач заповнює поля та натискає кнопку "Sale".

4. Програма перевіряє введені дані та додає новий автомобіль у список.

5. Форма закривається

Додатковий сценарій

1. Користувач натискає кнопку "Sale", але поточний юзер - "default".

2. Автомобіль не додається. Відкривається вікно з помилкою, де зазначено що користувач має змінити поточного користувача, щоб скористатися цією функцією.

Додатковий сценарій

1. Користувач не заповнив обов'язкові поля, або ввів об'єм двигуна у невірному форматі. Та натиснув кнопку "Sale".

2. Відкривається вікно з помилкою, де зазначено які поля необхідно заповнити та у якому форматі треба вводити дані.

1.2 Функції програми

Функція 1. Зміна поточного користувача

Форма зміни поточного користувача (рис. 1.1) містить одне текстове поле, інформацію про поточного користувача та кнопку.

Юзернейм користувача - текст.

The image shows a rectangular form with a light gray border. Inside, there is a text input field with a vertical cursor on the left. Above the input field, there is a label 'Юзернейм користувача - текст.' and a button labeled 'Set!'.

Рисунок 1.1 – Форма зміни поточного користувача

Максимальна довжина текстового поля – 32 символи. Якщо рядок пустий, при натисканні на кнопку нічого не відбувається.

Під текстовим полем міститься кнопка “Set!”.

Функція 2. Фільтрація списку автомобілів

Головна форма (рис. 1.2) містить 6 текстових полів.

Brand - фрагмент назви бренду.

Model - фрагмент назви моделі.

Year From - рік від - включно.

Year To - рік до - включно.

Price From - ціна від.

Price To - ціна до.

The image shows a web application window with a search interface. At the top, there are three horizontal bars. Below them, there are input fields for 'Brand:', 'From:', 'From:', 'Model:', 'To:', and 'To:'. To the right of the 'To:' fields is a small black square button. Below the input fields is a list of results, represented by horizontal bars of varying lengths, indicating a list of items.

Рисунок 1.2 – Головна форма

Максимальна довжина для полів бренду та моделі – 32 символи. Для ціни та року використовується чисельний ввід, максимальна довжина для року - 4, для ціни - 10.

Якщо рядок пустий, фільтрування за цим параметром не відбувається.

Результат пошуку буде одразу відображатись у списку нижче. Елементами списку буде коротка інформація про автомобіль, що включатиме в себе інформацію про: бренд, модель, рік та ціну.

Якщо за заданими параметрами жодного автомобіля не знайдено, список буде порожнім.

Функція 3. Додавання нового автомобілю

Форма (рис. 1.3) містить 7 текстових полів.

Brand - назва бренду.

Model - назва моделі.

Year - рік.

Price - ціна.

Engine - об'єм двигуна у літрах.

Condition - стан автомобіля: новий або б/в.

Gearbox - тип трансмісії.

A rectangular form with a thin border. Inside, there are seven rows, each consisting of a text label 'Lorem ipsum' on the left and a rectangular input field on the right. The input fields have a small vertical line on their left side. Below these seven rows is a single, wider rectangular button with a thick black horizontal bar in the center.

Рисунок 1.3 – Форма додавання нового автомобілю

Максимальна довжина для полів бренду та моделі – 32 символи. Для ціни та року використовується чисельний ввід, максимальна довжина для року - 4, для ціни - 10. Для полів Condition та Gearbox використовуються випадаючі меню з визначеними параметрами: для Condition - “New” або “Used”, для Gearbox - “Mechanical” або “Automatic”. Для поля Engine використовується дробний ввід з дільником – “,”.

Якщо поточний користувач - “default”, відкриється вікно з повідомленням про це, оголошення не створиться.

Обов’язкові поля: бренд, модель, ціна, рік. Якщо їх не ввести відкриється вікно з помилкою, де будуть вказані не заповнені поля.

2 ПРОЕКТУВАННЯ ПРОГРАМИ

Після аналізу вхідних даних, була обрано розробку настільного застосунку з GUI та архітектурою MVC. GUI буде розроблятися за допомогою “Win Forms”.

Зовнішні дані будуть зберігатися у форматі “Json”. Доступ до зовнішніх даних буде здійснюватись за допомогою включеної до .NET 7.0 бібліотеки “System.Text.Json”.

Проектування моделей варто почати з головного – модель автомобіля. Вона повинна включати в себе усі основні дані (рис. 2.1).

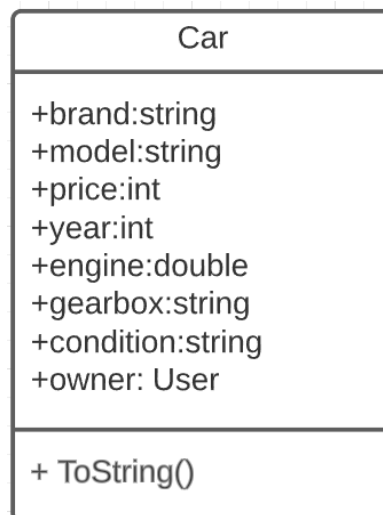


Рисунок 2.1 – Діаграма класу “Car”

Він залежить від класу “User”, який повинен містити в собі юзернейм користувача та його повідомлення (рис. 2.2).

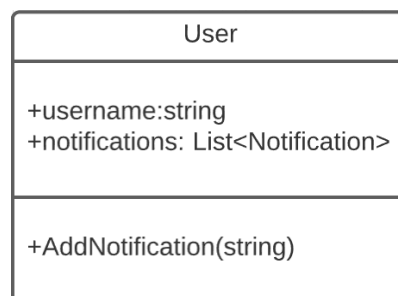


Рисунок 2.2 – Діаграма класу “User”

Клас “User” залежить від класу “Notification”, який містить в собі текст повідомлення (рис. 2.3).

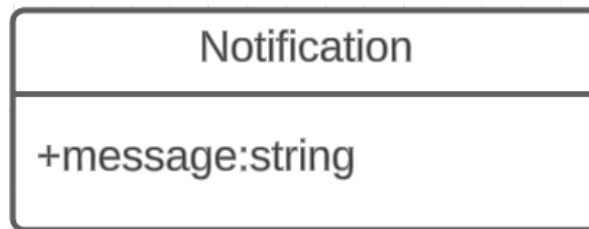


Рисунок 2.3 – Діаграма класу “Notification”

Для зручного зберігання даних, створимо ще два класи, які будуть містити у собі список усіх користувачів та автомобілів, а також методи для фільтрування, поточного користувача та додаткові методи (рис. 2.4 та 2.5).

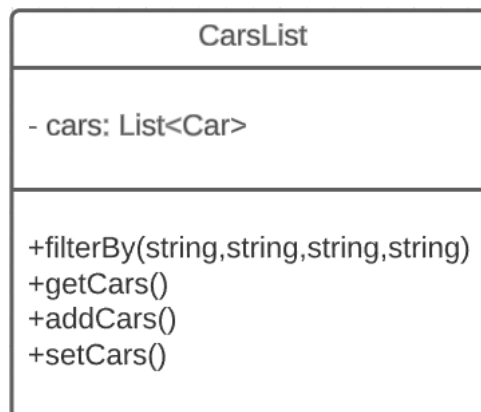


Рисунок 2.4 – Діаграма класу “CarsList”

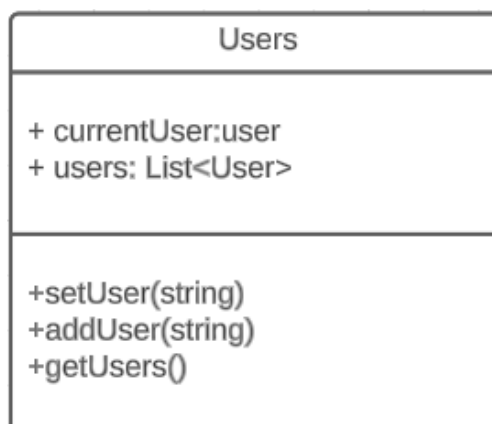


Рисунок 2.5 – Діаграма класу “Users”

Встановлюємо залежності між формами та моделями і отримаємо діаграму класів (рис. 2.6)

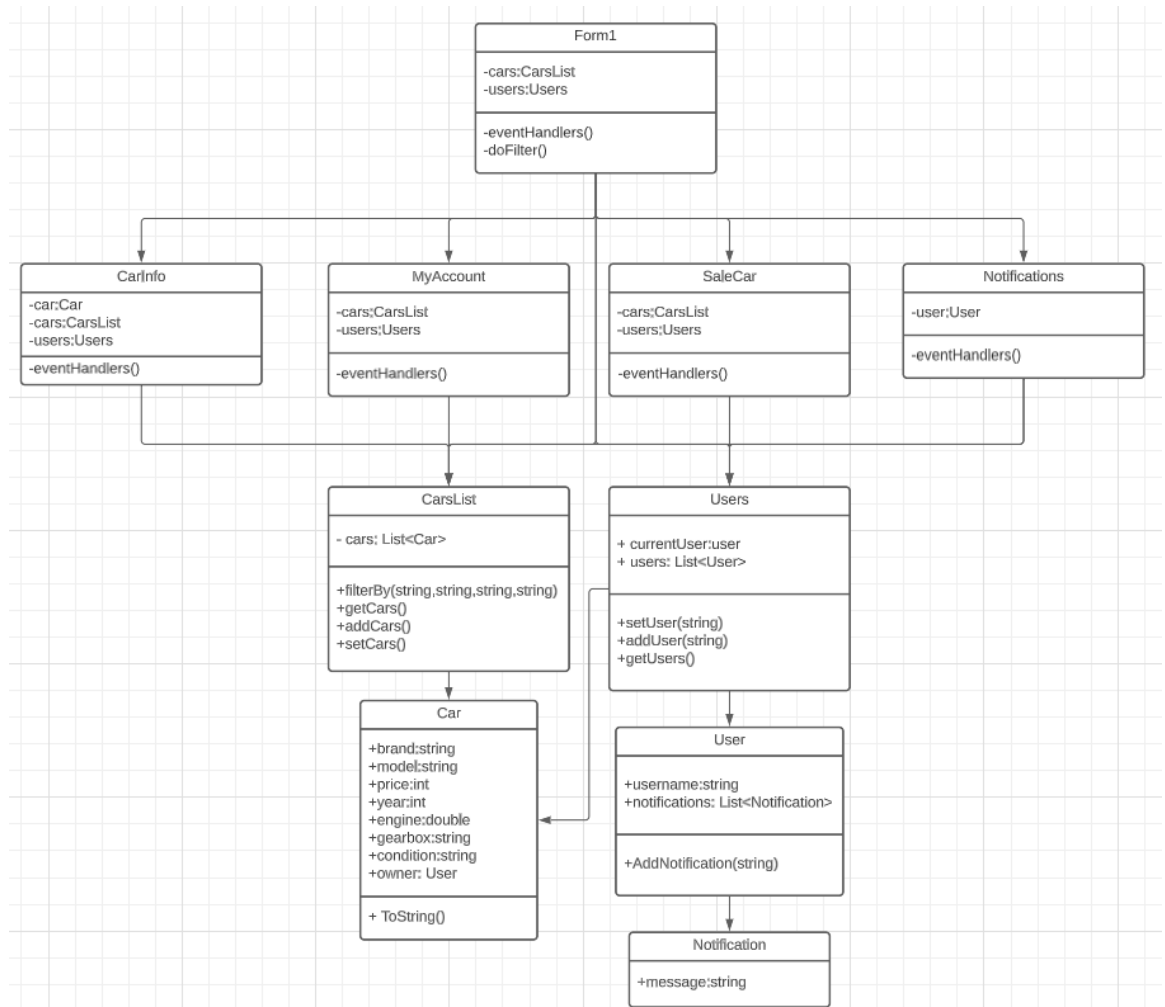


Рисунок 2.6 – Діаграма класів

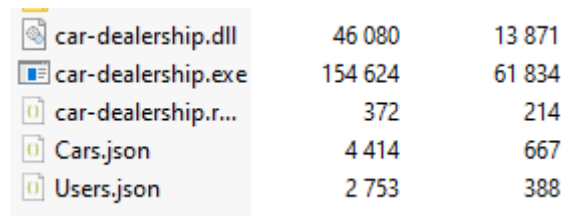
Головна форма ініціює виклик інших форм, які виконують різні функції. Форми взаємозв'язані екземплярами класів “CarsList” та “Users”, які ініціюються в головній формі та через параметри передаються в інші форми.

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

3.1 Інструкція по розгортанню програми

Для початку роботи з програмою необхідно загрузити архів з трьома файлами: “car-dealership.exe”, “Cars.json”, “Users.json”, “car-dealership.dll”, “car-dealership.runtimeconfig.json” (рис. 3.1).

Розархівувати архів в будь-яку папку, два рази клікнути по файлу “car-dealership.exe”, після чого відкриється головна форма та можна буде розпочинати роботу з програмою.



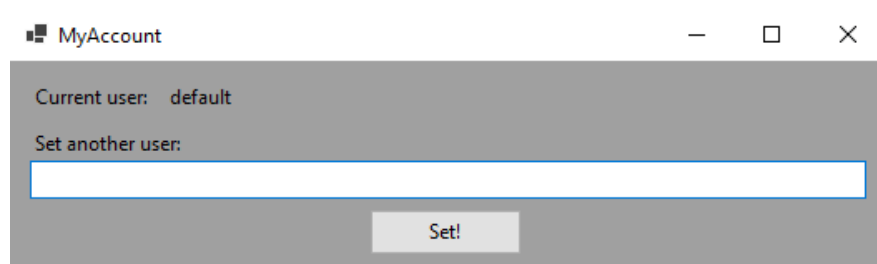
car-dealership.dll	46 080	13 871
car-dealership.exe	154 624	61 834
car-dealership.r...	372	214
Cars.json	4 414	667
Users.json	2 753	388

Рисунок 3.1 – Вигляд архіву з програмою

3.2 Інструкція по роботі з програмою

1 Зміна поточного користувача

Після запуску програми користувач бачить головну форму. Для зміни поточного користувача необхідно натиснути кнопку “My profile” в меню. Після чого відкриється форма (рис 3.2), куди необхідно вписати новий юзернейм користувача та натиснути кнопку “Set!”.



MyAccount

Current user: default

Set another user:

Set!

Рисунок 3.2 – Вигляд форми “MyAccount”

2 Покупка автомобіля

Змінивши поточного користувача можна надіслати повідомлення продавцю про намір придбати автомобіль. Для цього необхідно знайти необхідний автомобіль, виділити його, натиснувши на нього в списку, натиснути кнопку “Get Info”. Після цього відкриється додаткове вікно з повною інформацією про обраний автомобіль (рис. 3.3), де необхідно натиснути кнопку “Send message to owner”.

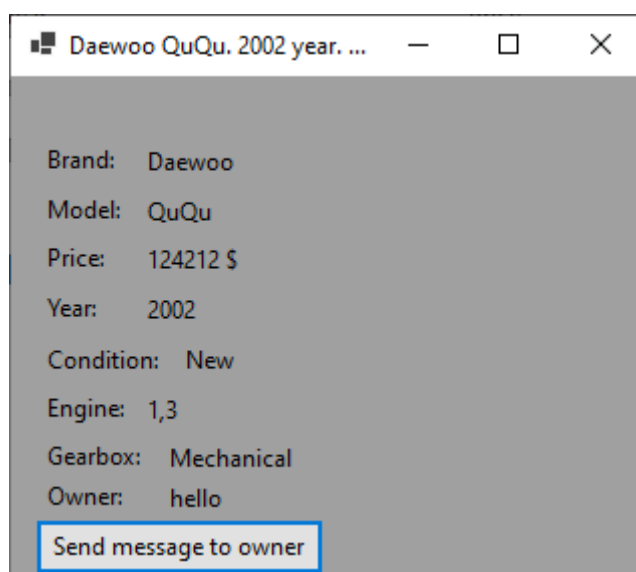
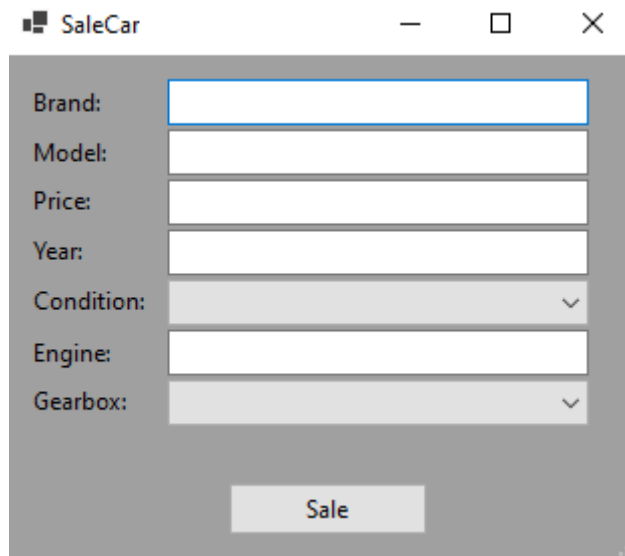


Рисунок 3.3 – Вигляд форми з детальною інформацією.

3 Продаж автомобіля

Змінивши поточного користувача можна виставити своє авто на продаж. Для цього необхідно в верхній частині головної форми навести мишкою на кнопку “Cars” після цього у випадаючому меню обрати пункт “Sale Car”. Відкриється додаткова форма для введення даних (рис. 3.4). Після заповнення текстових текстових полів, треба натиснути кнопку “Sale”, якщо операція вдала, то форма закриється а авто з’явиться у списку.



The image shows a window titled "SaleCar" with standard Windows window controls (minimize, maximize, close). Inside the window is a form with the following fields and controls:

- Brand:
- Model:
- Price:
- Year:
- Condition:
- Engine:
- Gearbox:

At the bottom center of the form is a button labeled "Sale".

Рисунок 3.4 – Вигляд форми для продажу автомобіля

3.3 Інструкція по видаленню програми

Для видалення програми необхідно видалити всі, розархівовані раніше, файли, проте треба пам'ятати, що інформація зберігається локально у файлах "Users.json" та "Cars.json", і при їх видаленні інформація про користувачі та автомобілі буде знищена.

ВИСНОВКИ

У результаті розробки отримали застосунок для покупки та продажу автомобілів. Який містить в собі два незалежних сценарії, покупка автомобіля та його продаж. Він значно спростить процес купівлі/продажу автомобіля, бо дозволяє централізувати усі варіанти та зв'язати покупця з продавцем.

Застосунок знайде своє місце як інструмент для очних автосалонів, де необхідно знати які автомобілі зараз є в наявності, або може бути перероблений для широкого використання по території України, що дозволить користувачам з різних міст купувати то продавати свої автомобілі.

Якщо оцінювати ступінь готовності цього застосунку, можна дати оцінку мінімально працюючого варіанту, бо на достатньому рівні реалізован основний функціонал, який дає змогу користуватись та ознайомлюватись з продуктом.

Якщо стане мета покращити цей застосунок, необхідно почати з повноцінної системи аутентифікації користувача, надати можливість переглядати свої автомобілі ,які зараз знаходяться у продажу, та можливість покупцям робити підписку за обраними фільтрами, щоб отримувати повідомлення, якщо у наявності з'являтиметься новий автомобіль, проте, в залежності від подальшого місця використання цього застосунку, цей список може бути змінений.

Під час розробки цього застосунку, я здобув навички використання “WinForms” та закріпив отриманні під час вивчення курсу “Об’єктно-орієнтоване програмування” знання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондарєв В.М. Електронний учбовий посібник з дисципліни «Основи програмування» URL: <http://tss.co.ua:5555/>

2. Методичні вказівки до виконання курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» для студентів першого (бакалаврського) рівня вищої освіти усіх форм навчання спеціальності 121 «Інженерія програмного забезпечення», освітньо-професійна програма «Програмна інженерія» / Упоряд.: В.М. Бондарєв, Ю.Ю. Черепанова – Харків: ХНУРЕ, 2022. – 39 с.

3. Microsoft. .NET documentation. URL: <https://learn.microsoft.com/uk-ua/dotnet/>

ДОДАТОК А

Вихідний код програми

1 Папка Models:

- Файл "Car.cs"

```
public class Car
{
    public string brand { get; set; }
    public string model { get; set; }
    public int price { get; set; }
    public int year { get; set; }
    public double engine { get; set; }
    public string gearbox { get; set; }
    public string condition { get; set; }
    public User owner { get; set; }
    public Car(){}
    public Car(string brand, string model, int price, int
year, double engine, string gearbox, string condition, User
owner)
    {
        this.brand = brand; this.model = model; this.price
= price;
        this.year = year; this.engine = engine;
this.gearbox = gearbox;
        this.condition = condition; this.owner = owner;
    }
    public override string ToString()
    {
        return $"{brand} {model}. {year} year. {price} $.";
    }
}
```

- Файл "CarsList.cs"

```
public class CarsList
{
    private List<Car> cars;
    public CarsList()
    {
        this.cars = new List<Car>();
    }

    public CarsList(List<Car> cars)
    {
        this.cars = cars;
    }
}
```

```

        public List<Car> filterBy(string model, string brand,
string yearFrom, string yearTo, string priceFrom, string
priceTo)
        {
            List<Car> result = new List<Car>();

            string m = model.ToLower();
            string b = brand.ToLower();
            int yF = 1900;
            int yT = 3000;
            int pF = 0;
            int pT = 10000000;

            if (yearFrom != "")
            {
                yF = int.Parse(yearFrom);
            };
            if (yearTo != "")
            {
                yT = int.Parse(yearTo);
            }
            if (priceFrom != "")
            {
                pF = int.Parse(priceFrom);
            };
            if (priceTo != "")
            {
                pT = int.Parse(priceTo);
            }

            foreach (Car car in cars)
            {
                if (car.model.ToLower().Contains(m)
                    && car.brand.ToLower().Contains(b)
                    && car.year >= yF
                    && car.year <= yT
                    && car.price >= pF
                    && car.price <= pT)
                {
                    result.Add(car);
                }
            }
            return result;
        }

        public List<Car> getCars() { return cars; }
        public void addCar(Car car)
        {
            cars.Add(car);
        }

```

```

    }
    public void setCars(List<Car> cars)
    {
        this.cars = cars;
    }
}

```

- Файл "Notification.cs"

```

public class Notification
{
    public string message { get; set; }
    public Notification()
    {
    }
    public Notification(string msg)
    {
        message = msg;
    }
    public override string ToString()
    {
        return message;
    }
}

```

- Файл "User.cs"

```

public class User
{
    public string username { get; set; }
    public List<Notification> notifications { get; set; }
    public User()
    {
    }
    public User(string username)
    {
        this.username = username;
        notifications = new();
    }

    public void AddNotification(string message)
    {
        notifications.Add(new Notification(message));
    }
}

```

- Файл "Users.cs"

```

public class Users
{
    public User currentUser { get; set; }

    public List<User> users { get; set; }
    public Users()
    {
    }
}

```



```

    {
        users = new List<User>();
    }
    public void setUser(string un)
    {
        foreach (User u in users)
        {
            if (u.username == un) currentUser = u;
        }
    }
    public void addUser(string username)
    {
        User newUser = new(username);
        foreach (User u in users)
        {
            if (u.username == username)
            {
                return;
            }
        }
        users.Add(newUser);
        setUser(newUser.username);
    }
    public List<User> getUsers()
    {
        return users;
    }
}

```

2 Папка Forms:

- Файл "CarInfo.cs"

```

public partial class CarInfo : Form
{
    private Car car;
    private Users users;
    private CarsList cars;
    public CarInfo(Car car, Users users, CarsList cars)
    {
        InitializeComponent();

        this.car = car;
        this.users = users;
        this.cars = cars;

        this.Text = car.ToString();
        CarInfoModel.Text = car.model;
        CarInfoBrand.Text = car.brand;
        CarInfoPrice.Text = $"{car.price} $";
        CarInfoYear.Text = $"{car.year}";
    }
}

```

```

        CarInfoCondition.Text = car.condition;
        CarInfoEngine.Text = $"{car.engine}";
        CarInfoGearbox.Text = car.gearbox;
        CarInfoOwner.Text = car.owner.username;
    }

    private void CarInfo_Load(object sender, EventArgs e)
    {

    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (car.owner.username ==
users.currentUser.username || users.currentUser.username ==
"default") return;
        foreach (User u in users.users)
        {
            if (u.username == car.owner.username)
            {
                u.AddNotification($"User
'{users.currentUser.username}', want to buy your car :
{car.ToString()}");
                DataAccess.Save(cars, users);
            }
        }
    }
}

```

- Файл “Form1.cs”

```

public partial class Form1 : Form
{
    private CarsList cars = new();
    private Users users = new();

    public Form1()
    {
        InitializeComponent();
        DataAccess.Load(cars, users);
        users.setUser("default");
        bindingSource1.DataSource = cars.getCars();
    }

    private void SaleCarStripButton_Click(object sender,
EventArgs e)
    {
        var form = new SaleCar(cars, users);
        form.ShowDialog();
        if (form.DialogResult == DialogResult.OK)

```

```

        {
            bindingSource1.DataSource =
form.cars.getCars();
            bindingSource1.ResetBindings(true);
        }
    }

    private void NotificationsStripButton_Click(object
sender, EventArgs e)
    {
        var form = new Notifications(users.currentUser);
        form.Show();
    }

    private void ProfileStripButton_Click(object sender,
EventArgs e)
    {
        var form = new MyAccount(users, cars);
        form.Show();
    }

    private void GetCarInfoButton_Click(object sender,
EventArgs e)
    {
        Car? car = CarsList.SelectedItem as Car;
        if (car == null) return;

        var form = new CarInfo(car, users, cars);
        form.Show();
    }

    private void DoFilter()
    {
        List<Car> filtered =
cars.filterBy(TextBoxFilterModel.Text, TextBoxFilterBrand.Text,
                TextBoxFilterYearFrom.Text,
                TextBoxFilterYearTo.Text,
                TextBoxFilterPriceFrom.Text,
                TextBoxFilterPriceTo.Text);
        bindingSource1.DataSource = filtered;
    }

    private void TextBoxFilterModel_TextChanged(object
sender, EventArgs e)
    {
        DoFilter();
    }

    private void TextBoxFilterBrand_TextChanged(object
sender, EventArgs e)
    {
        DoFilter();
    }

```

```

    }

    private void TextBoxFilterYearFrom_TextChanged(object
sender, EventArgs e)
    {
        if
(System.Text.RegularExpressions.Regex.IsMatch(TextBoxFilterYear
From.Text, "[^0-9]"))
        {
            TextBoxFilterYearFrom.Text =
TextBoxFilterYearFrom.Text.Remove(TextBoxFilterYearFrom.Text.Le
ngth - 1);
        }

        DoFilter();
    }

    private void TextBoxFilterYearTo_TextChanged(object
sender, EventArgs e)
    {
        if
(System.Text.RegularExpressions.Regex.IsMatch(TextBoxFilterYear
To.Text, "[^0-9]"))
        {
            TextBoxFilterYearTo.Text =
TextBoxFilterYearTo.Text.Remove(TextBoxFilterYearTo.Text.Length
- 1);
        }

        DoFilter();
    }

    private void TextBoxFilterPriceFrom_TextChanged(object
sender, EventArgs e)
    {
        if
(System.Text.RegularExpressions.Regex.IsMatch(TextBoxFilterPric
eFrom.Text, "[^0-9]"))
        {
            TextBoxFilterPriceFrom.Text =
TextBoxFilterPriceFrom.Text.Remove(TextBoxFilterPriceFrom.Text.
Length - 1);
        }

        DoFilter();
    }

    private void TextBoxFilterPriceTo_TextChanged(object
sender, EventArgs e)
    {

```

```

        if
        (System.Text.RegularExpressions.Regex.IsMatch(TextBoxFilterPriceTo.Text, "[^0-9]"))
        {
            TextBoxFilterPriceTo.Text =
            TextBoxFilterPriceTo.Text.Remove(TextBoxFilterPriceTo.Text.Length - 1);
        }

        DoFilter();
    }
}

```

- Файл "MyAccount.cs"

```

public partial class MyAccount : Form
{
    private Users users;
    private CarsList cars;
    public MyAccount(Users users, CarsList cars)
    {
        InitializeComponent();
        this.users = users;
        this.cars = cars;

        MyAccountCurrentUser.Text =
        users.currentUser.username;
    }

    private void MyAccount_Load(object sender, EventArgs e)
    {
    }

    private void button1_Click(object sender, EventArgs e)
    {
        users.addUser(MyAccountUsername.Text);
        users.setUser(MyAccountUsername.Text);
        MyAccountCurrentUser.Text =
        users.currentUser.username;
        DataAccess.Save(cars, users);
    }
}

```

- Файл "Notifications.cs"

```

public partial class Notifications : Form
{
    public Notifications(User currentUser)
    {
        InitializeComponent();
        bindingSource_notifications.DataSource =
        currentUser.notifications;
    }
}

```

```

        this.Text = currentUser.username;
    }

    private void listBox1_SelectedIndexChanged(object
sender, EventArgs e)
    {

    }

    private void Notifications_Load(object sender,
EventArgs e)
    {
    }
}

```

- Файл "Notifications.cs"

```

public partial class SaleCar : Form
{
    public CarsList cars;
    public Users users;
    public SaleCar(CarsList cars, Users users)
    {
        InitializeComponent();
        this.cars = cars;
        this.users = users;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (users.currentUser.username == "default")
        {
            MessageBox.Show("You must switch user to use
it.", "oopps...", MessageBoxButtons.OK);
            return;
        }
        string errors = "";
        double engine = 0;

        if (SaleCarBrand.Text == "") errors += "Car must
have a brand. ";
        if (SaleCarModel.Text == "") errors += "Car must
have a model. ";
        if (SaleCarPrice.Text == "") errors += "Car must
have a price. ";
        try
        {
            engine = Convert.ToDouble(SaleCarEngine.Text);
        }
        catch
        {

```

```

        errors += "Engine must be a number in format
*,* ";
    }
    if (errors != "")
    {
        MessageBox.Show(errors, "oopps...",
        MessageBoxButtons.OK);
        return;
    }

    string condition = SaleCarCondition.Text;
    if (SaleCarCondition.Text == "")
    {
        condition = "Used";
    }

    string gearbox = SaleCarGearbox.Text;
    if (SaleCarGearbox.Text == "")
    {
        gearbox = "Not specified";
    }

    cars.addCar(new Car(SaleCarBrand.Text,
    SaleCarModel.Text, int.Parse(SaleCarPrice.Text),
    int.Parse(SaleCarYear.Text), engine, gearbox, condition,
    users.currentUser));

    DataAccess.Save(cars, users);
    this.DialogResult = DialogResult.OK;
    this.Close();
}

private void SaleCarPrice_TextChanged(object sender,
EventArgs e)
{
    if
    (System.Text.RegularExpressions.Regex.IsMatch(SaleCarPrice.Text
    , "[^0-9]"))
    {
        SaleCarPrice.Text =
        SaleCarPrice.Text.Remove(SaleCarPrice.Text.Length - 1);
    }
}

private void SaleCarYear_TextChanged(object sender,
EventArgs e)
{
    if
    (System.Text.RegularExpressions.Regex.IsMatch(SaleCarYear.Text,
    "[^0-9]"))

```

```

        {
            SaleCarYear.Text =
SaleCarYear.Text.Remove(SaleCarYear.Text.Length - 1);
        }
    }

    private void SaleCarEngine_TextChanged(object sender,
EventArgs e)
    {
    }
}

```

3 Папка Controls

- Файл "Data.cs"

```

public static class DataAccess
{
    const string DATA_PATH_CARS = "Cars.json";
    const string DATA_PATH_USERS = "Users.json";

    public static void Save(CarsList cars, Users
users)
    {
        string jsonStringCars =
JsonSerializer.Serialize(cars.getCars());
        string jsonStringUsers =
JsonSerializer.Serialize(users.getUsers());
        File.WriteAllText(DATA_PATH_CARS,
jsonStringCars);
        File.WriteAllText(DATA_PATH_USERS,
jsonStringUsers);
    }

    public static void Load(CarsList cars, Users
users)
    {
        string jsonStringCars =
File.ReadAllText(DATA_PATH_CARS);
        string jsonStringUsers =
File.ReadAllText(DATA_PATH_USERS);
        var carList =
JsonSerializer.Deserialize<List<Car>>(jsonStringCars);
        var userList =
JsonSerializer.Deserialize<List<User>>(jsonStringUsers);
        cars.setCars(carList);
        users.users = userList;
    }
}

```