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-апаратна система інформування клієнтів аптеки засобами C# ASP .NET CORE MVC

Software and hardware system for informing customers of the pharmacy using C# ASP .NET CORE MVC

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

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Янковський Богдан Олександрович

(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

Янковському Богдану Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Програмно-апаратна система інформування клієнтів аптеки засобами C# ASP .NET CORE MVC*

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):
Вступ
Аналіз проблеми за темою роботи та постановка завдань дослідження розробки програмно-апаратної системи інформування клієнтів аптеки засобами C# ASP .NET CORE MVC
Інструменти для розробки Програмно-апаратної системи інформування клієнтів аптеки засобами C# ASP .NET CORE MVC
Розробка програмно-апаратної системи інформування клієнтів аптеки
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Огляд програмних засобів розробки системи інформування клієнтів аптеки	*Пех П.А., доцент*		
Інструменти для розробки програмно-апаратної системи інформування клієнтів аптеки	*Пех П.А., доцент*		
Розробка програмно-апаратної системи інформування клієнтів аптеки	*Пех П.А., доцент*		
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.	*Інструменти для розробки програмно-апаратної системи інформування клієнтів аптеки*	до 02.03.2025 р.	Виконано
3.	*Розробка програмно-апаратної системи інформування клієнтів аптеки*	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.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Янковський Б.О.
(підпис) (прізвище, ініціали)

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

АНОТАЦІЯ

Янковський Б.О. Програмно-апаратна система інформування клієнтів аптеки засобами C# ASP .NET CORE MVC.

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, списку використаних джерел та додатків.

У роботі обгрунтовано вибір мови та середовища програмування для розробки програмно-апаратної системи інформування клієнтів аптеки засобами C# ASP .NET CORE MVC. Представлено детальну характеристику інструментів середовища ASP.NET Core MVC для створення веб-додатків і технології їх використання для розробки інформаційних систем. У якості сховища даних використано файли JSON для зберігання інформації про клієнтів, ліки та інші аспекти діяльності аптеки. Розроблено програмний проєкт, що включає систему реєстрації та авторизації клієнтів, можливість перегляду списку ліків з описом, ціною та зображенням, із можливістю пошуку та сортування, а також адміністраторський інтерфейс для управління ліками та для перегляду списку клієнтів.

Ключові слова: середовище C# ASP .NET CORE MVC, програмно-апаратна система, інформування клієнтів аптеки, файли JSON

ANNOTATION

Yankovskyi B. Software and Hardware System for Informing Pharmacy Clients Using C# ASP.NET CORE MVC.

Bachelor's qualification work of the Educational Program «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The work justifies the choice of programming language and environment for the development of a software and hardware system for informing pharmacy clients using C# ASP.NET CORE MVC. A detailed description of ASP.NET Core MVC environment tools for creating web applications and technologies for developing information systems is presented. JSON files are used as a data storage system to store information about clients, medicines, and other aspects of pharmacy operations. A software project has been developed that includes a client registration and authentication system, the ability to view a list of medicines with a description, price, and image, with search and sorting functionality, as well as an administrator interface for managing medicines and viewing the client list.

Keywords: C# ASP.NET CORE MVC environment, software and hardware system, informing pharmacy clients, JSON files.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі інформаційні технології відіграють важливу роль у всіх сферах діяльності людини, включаючи сферу охорони здоров'я та фармацевтичну галузь. Ефективне управління інформацією про лікарські засоби та взаємодія між аптекою і клієнтами є важливими завданнями для забезпечення якісного обслуговування.

На сьогоднішній день особливу актуальність набуває розробка програмно-апаратних рішень, які дозволяють клієнтам отримувати актуальну інформацію про лікарські засоби, акційні пропозиції та нові надходження, а також здійснювати пошук необхідних медикаментів у режимі реального часу. Використання технологій C# та ASP.NET Core MVC у створенні такої системи надає можливість розробити зручний та гнучкий веб-додаток, що відповідає сучасним вимогам безпеки та продуктивності.

Метою роботи є розробка програмно-апаратної системи інформування клієнтів аптеки засобами C# ASP.NET Core MVC.

Об'єкт дослідження – інформаційні системи аптек, що забезпечують інформування клієнтів про лікарські засоби.

Предмет дослідження – процеси та засоби розробки веб-додатків для інформування клієнтів аптек засобами C# ASP.NET Core MVC.

Завдання дослідження:

– провести огляд програмних засобів розробки програмно-апаратної системи інформування клієнтів аптеки засобами C# ASP.NET Core MVC;

– розробити функціонал реєстрації та авторизації клієнтів з використанням номеру телефону та імені;

– реалізувати інтерфейс для перегляду списку ліків з можливістю пошуку, сортування, відображення ціни, опису та зображення;

– розробити адміністраторську панель для керування списком ліків та перегляду зареєстрованих клієнтів;

– використати JSON-файли як альтернативу базі даних для зберігання інформації про ліки та клієнтів;

Методи розробки. Проєкт базується на використанні сучасних технологій веб-розробки, зокрема C# ASP.NET Core MVC, що дозволяє створювати масштабовані, продуктивні та безпечні веб-додатки. Для зберігання інформації використано файлову систему JSON, що забезпечує простоту в реалізації та управлінні даними.

Наукова новизна одержаних результатів полягає у розробці інноваційної програмно-апаратної системи інформування клієнтів аптеки, яка поєднує сучасні технології веб-розробки та зручність доступу до інформації.

Практичне значення одержаних результатів. Розроблена система може бути використана в аптеках для покращення обслуговування клієнтів, а також у навчальному процесі для вивчення веб-технологій та інформаційних систем у сфері фармацевтики.

РОЗДІЛ 1
ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ ІНФОРМУВАННЯ КЛІЄНТІВ АПТЕКИ

1.1 Програмно-апаратна система інформування клієнтів аптеки та завдання щодо її розробки

Програмно-апаратна система інформування клієнтів аптеки розробляється для забезпечення зручного доступу користувачів до інформації про ліки. Головною метою проєкту є створення веб-додатку, який дозволить клієнтам аптеки отримувати актуальну інформацію про доступні ліки, а адміністраторам – керувати асортиментом ліків та інформувати клієнтів.

Основні завдання проєкту:

– реалізація системи реєстрації та авторизації клієнтів за номером телефону та ім’ям;

– надання клієнтам можливості перегляду списку ліків із пошуком та сортуванням (за ціною, назвою і по id);

– відображення детальної інформації про кожен препарат (зображення, опис, ціна, назва);

– розробка панелі адміністратора для керування асортиментом ліків (додавання і видалення ліків) і перегляду списку клієнтів;

– додавання стилів, фонових звуків, зображеннь і динамічних елементів для естетичності проєкту.

1.2 Огляд мов програмування для розробки проєкту

Для розробки даної програмно-апаратної системи можна використати різні мови програмування. Розглянемо деякі з них.

«C# – це об’єктно-орієнтована мова програмування, створена Microsoft, яка працює на платформі .NET» [1].

C# була створена під керівництвом Андерса Гейлсберга, Скотта Вілтамута та Пітера Гольде в Microsoft Research. Синтаксис цієї мови схожий на C++ та

Java, що робить її доступною для розробників, знайомих із цими мовами. C# є строго типізованою мовою з підтримкою статичної типізації та містить ряд потужних можливостей, таких як поліморфізм, перевантаження операторів, події, властивості, обробка винятків та коментарі в XML-форматі. Вона також запозичила кращі практики з таких мов, як-от C++, Object Pascal, Модул і Smalltalk. Однак, на відміну від C++, C# не підтримує множинне успадкування класів, що спрощує розробку та уникає деяких проблем, пов'язаних із цією функціональністю [2].

«JavaScript – це мова сценаріїв або програмування, яка дозволяє реалізовувати складні функції на веб-сторінках» [3].

JavaScript є мовою програмування, що реалізує стандарт ECMAScript і найчастіше використовується для створення інтерактивних вебсторінок. Він працює на боці клієнта, дозволяючи користувачам взаємодіяти з інтерфейсом, керувати поведінкою браузера, динамічно змінювати контент сторінки та здійснювати асинхронний обмін даними із сервером. Ця мова належить до скриптових і використовує динамічну типізацію, що дає можливість змінним приймати різні типи даних під час виконання. JavaScript базується на прототипному наслідуванні, що є одним із варіантів об'єктно-орієнтованого підходу, але також підтримує імперативну та частково функційну парадигми програмування. Серед ключових особливостей JavaScript можна виділити слабку типізацію, автоматичне керування пам'яттю, прототипне наслідування, а також можливість використання функцій як об'єктів першого класу, що відкриває широкий спектр можливостей для гнучкого програмування [4].

«HTML – це стандартна мова розмітки для документів, призначених для відображення у веб-браузері» [5].

HTML (HyperText Markup Language) є основною мовою розмітки, яка визначає структуру та зміст веб-сторінок. Він використовується для створення елементів сторінки, таких як текст, зображення, посилання, таблиці та мультимедійний контент. На відміну від HTML, для оформлення веб-сторінок застосовується CSS, а для додавання динамічної поведінки – JavaScript. Однією

з ключових особливостей HTML є підтримка гіпертекстових посилань, які об'єднують веб-сторінки між собою, створюючи зручну навігацію в Інтернеті. HTML використовує спеціальні елементи розмітки, такі як <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img>, <nav>, <video> та багато інших. Завдяки їм веб-браузери правильно відображають структуру сторінки та надають можливість взаємодії користувачів із контентом [6].

«CSS – це мова таблиць стилів, що використовується для опису презентації документа, написаного мовою розмітки, такою як HTML або XML» [7].

«CSS відповідає за стильове оформлення веб-сторінки, роблячи її візуально привабливою та завершеною. На відміну від HTML, CSS не створює нові елементи, а лише керує їхнім зовнішнім виглядом. Сама по собі CSS не має сенсу без HTML-документа, адже саме HTML визначає структуру сторінки, а CSS – її оформлення, розташування елементів та їхній стиль. Завдяки каскадним таблицям стилів можна змінювати кольори, шрифти, відступи, анімацію та інші параметри, що роблять веб-проєкт цілісним і зручним для користувачів» [8].

Враховуючи вимоги проєкту, було вирішено використовувати C# у поєднанні з ASP.NET Core MVC. C# є оптимальним вибором для серверної частини завдяки високій продуктивності та безпеці, тоді як JavaScript забезпечує інтерактивність веб-інтерфейсу. HTML і CSS відповідають за коректне відображення контенту та естетичний вигляд сайту.

1.3 Вибір середовища програмування для розробки проєкту

Для реалізації проєкту обране середовище розробки Visual Studio 2022, оскільки воно надає наступні переваги:

– підтримка ASP.NET Core MVC і C#;

– інтеграція з Git для контролю версій;

– зручні інструменти налагодження та тестування коду;

– розширена підтримка бібліотек та пакетів .NET;

Отже, використання C# у середовищі Visual Studio 2022 із застосуванням технології ASP.NET Core MVC дозволить створити ефективну та надійну програмно-апаратну систему для інформування клієнтів аптеки.

РОЗДІЛ 2
ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ ІНФОРМУВАННЯ КЛІЄНТІВ АПТЕКИ ЗАСОБАМИ C# ASP .NET CORE MVC

2.1 Компоненти елеменів середовища C# ASP .NET CORE MVC

ASP.NET Core MVC – це фреймворк для створення вебзастосунків, який реалізує шаблон Model-view-controller [9]. Основні компоненти середовища ASP.NET Core MVC наведені у таблиці 2.1.

Таблиця 2.1 – Компоненти середовища ASP.NET Core MVC.

Компоненти	Опис
Model (Модель)	Відповідає за управління даними та логікою бізнес-процесів.
View (Представлення)	Представляє інтерфейс користувача та відповідає за відображення інформації.
Controller (Контролер)	Обробляє запити користувачів, взаємодіє з моделями та передає дані у вигляді представленнь.
Middleware (Проміжне програмне забезпечення)	Набір компонентів, які обробляють HTTP-запити та відповіді, такі як автентифікація, авторизація, логування тощо.
Routing (Маршрутизація)	Механізм, що визначає, які контролери та методи будуть обробляти певні URL-запити.
Dependency Injection (Впровадження залежностей)	Використовується для підвищення модульності та керованості кодом.

Моделі в ASP.NET Core MVC відповідають за структуру даних і бізнес-логіку. Моделі містять інформацію, яку передають до контролерів, а також відповідають за валідацію і збереження даних.

Представлення відповідає за відображення інформації користувачу. Воно використовує Razor-синтаксис для динамічного генерування HTML-коду з даних, що передаються контролером. Представлення містить HTML, CSS і JavaScript і визначає, як виглядатиме інтерфейс вашого веб-додатку.

Контролер обробляє HTTP-запити від користувача, взаємодіє з моделями для отримання або зміни даних і передає результат у вигляді представлення. Контролери в ASP.NET Core реалізують логіку обробки запитів і визначають, які дії виконувати в залежності від запиту.

Проміжне програмне забезпечення є важливою частиною обробки запитів у ASP.NET Core. Це компоненти, які обробляють запити і відповіді перед тим, як вони потраплять до контролера або після того, як контролер їх обробить. Middleware включає в себе автентифікацію, авторизацію, обробку помилок, логування тощо.

Маршрутизація визначає, як запити HTTP зі специфічними URL-адресами співвідносяться з контролерами і методами дій у додатку. В ASP.NET Core маршрутний механізм дозволяє налаштовувати, які контролери та методи будуть обробляти конкретні запити.

Впровадження залежностей (DI) – це патерн проектування, який використовується для забезпечення звязків між компонентами додатку через контейнер залежностей. У ASP.NET Core DI дозволяє підключати сервіси та компоненти в контролери та інші об’єкти, підвищуючи тестованість і гнучкість коду.

2.2 Об’єкти проєкту

Для розробки програмно-апаратної системи інформування клієнтів аптеки необхідно створити основні об’єкти проєкту, які забезпечують коректну роботу веб-застосунку. Основні об’єкти та їхнє призначення наведені в таблиці 2.2.

Кожен із цих об’єктів виконує важливу роль у структурі застосунку. Моделі відповідають за управління даними, контролери обробляють запити, а

представлення відображають інформацію користувачам.

Таблиця 2.2 – Об’єкти проєкту

Об’єкт	Призначення
Проєкт (Project)	Створюється у середовищі Visual Studio 2022 на основі шаблону ASP.NET Core Empty.
Моделі (Models)	Класи, що містять структуру даних і бізнес-логіку.
Контролери (Controllers)	Обробляють HTTP-запити, взаємодіють з моделями та передають дані у вигляді представлення.
Представлення (Views)	Файли .cshtml, що містять HTML-розмітку з Razor-синтаксисом для динамічного відображення даних у веб-інтерфейсі.
Сервіси (Services)	Додаткові класи для бізнес-логіки, які впроваджуються через Dependency Injection (DI).
wwwroot	Каталог, що містить статичні файли (css, data, images, sounds).
Файл appsettings.json	Зберігає конфігураційні параметри
Файл Program.cs	Головний файл запуску програми, у якому налаштовується сервер та конфігурація сервісів.

Розробка програмно-апаратної системи інформування клієнтів аптеки передбачає використання різних об’єктів, кожен з яких виконує важливу роль у структурі застосунку. Моделі визначають структуру даних і бізнес-логіку, контролери обробляють запити, а представлення відповідають за відображення інформації користувачам. Сервіси сприяють розділенню логіки, що підвищує масштабованість проєкту [10].

Файли Program.cs та appsettings.json відповідають за налаштування середовища, а каталог wwwroot містить ресурси для коректного відображення інтерфейсу. Завдяки правильному розподілу обов’язків між цими об’єктами система буде ефективною, легко підтримуваною та зручною у використанні [11].

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ ІНФОРМУВАННЯ КЛІЄНТІВ АПТЕКИ

3.1 Початок створення програмно-апаратної системи інформування клієнтів аптеки

Програмно-апаратна система інформування клієнтів аптеки реалізована у кваліфікаційній роботі у вигляді сайту під ім'ям PharmacyInfoSystem за допомогою технології ASP .NET CORE MVC. Структура проєкту складається з двох частин – адміністратора та звичайного користувача. Звичайний користувач після реєстрації зможе переглядати список ліків, в той час, як адміністратор може додавати нові ліки до списку, а також переглядати список зареєстрованих клієнтів.

Створення ASP .NET CORE MVC-проєкту починається зі завантаження середовища програмування Visual Studio 2022 і появи його головного вікна (рис. 3.1). У ньому клацаємо по вкладці Create a new program, що приводить до появи наступного вікна (рис. 3.2). У ньому ми вибираємо мову програмування C#, та із списку платформ вибираємо ASP .NET CORE Empty (у списку є готовий шаблон ASP .NET CORE Veb App MVC, але ми будемо використовувати чистий шаблон з метою мати можливість впливати на всі особливості проєкту). Клацнувши по кнопці Next, викликаємо третє вікно, у якому присвоюємо проєкту ім'я PharmacyInfoSystem та вибираємо папку C:\Users\bohda\Desktop\Дипломна робота для його збереження. Далі натискаємо кнопку Next, після чого відкривається вікно з додатковою інформацією, де обираємо Framework .NET 9.0, і натискаємо Create, щоб створити новий проєкт.

Процес створення проєкту засобами Visual Studio 2022 завершується відкриттям головної сторінки сайту з початковою структурою файлів. Зазначимо, що цей процес є подібним і для інших технологій, а не лише для технології ASP .NET CORE Veb App MVC.

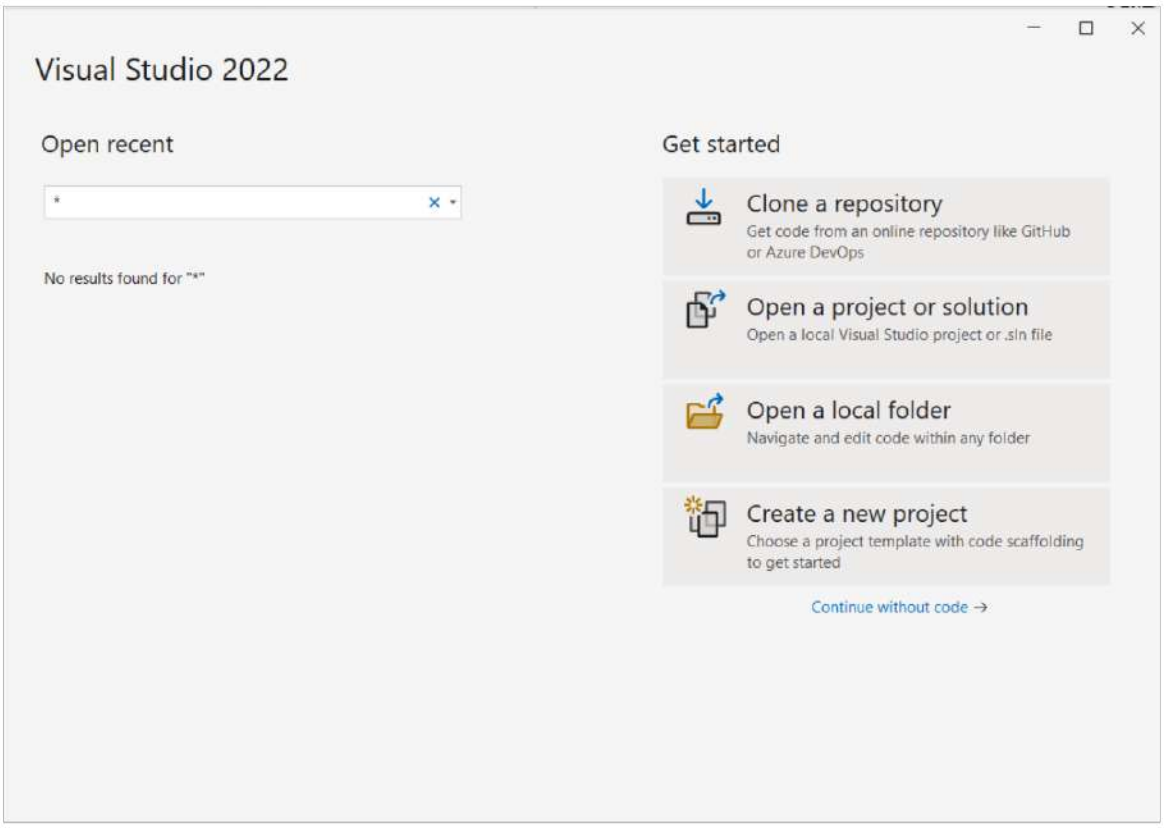

Рисунок 3.1 – Вигляд головного вікна середовища Visual Studio

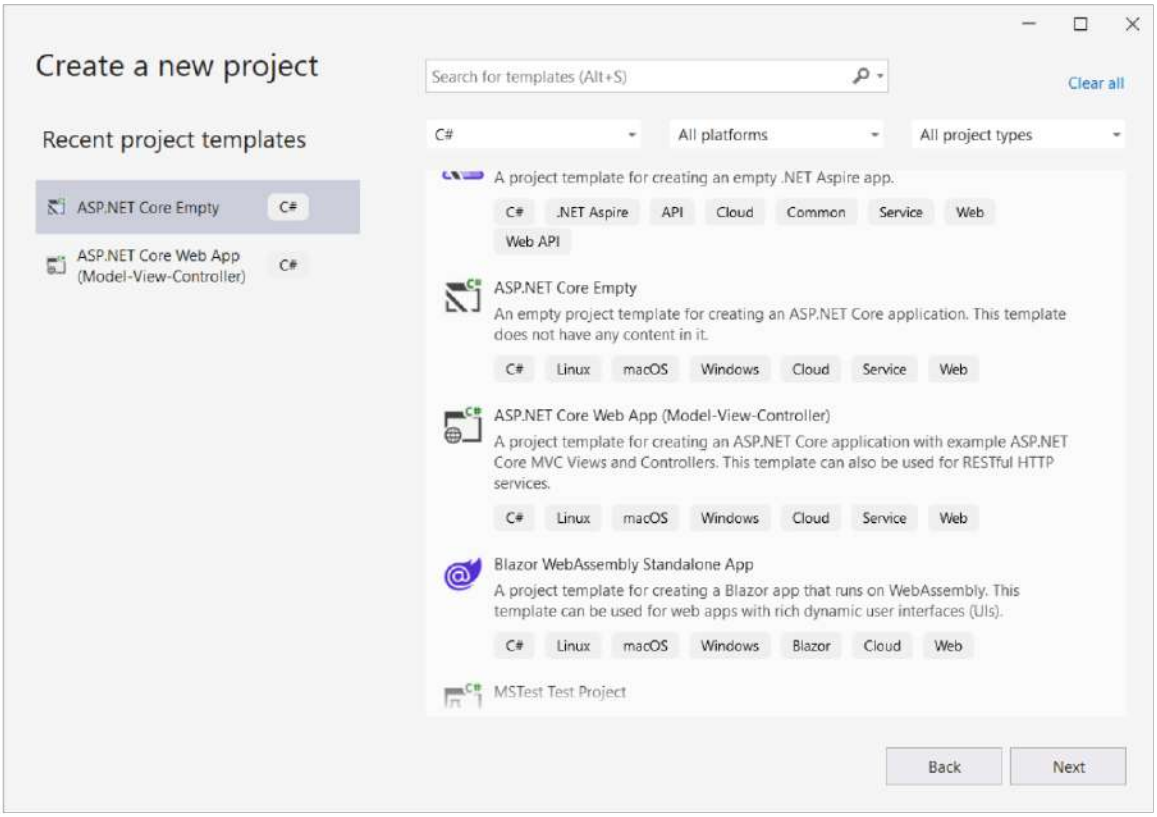

Рисунок 3.2 – Вибір мови C# та фреймворку .NET Framework для створення проєкту

Третє вікно зображено на рисунку 3.3. Вікно з додатковою інформацією на рисунку 3.4.

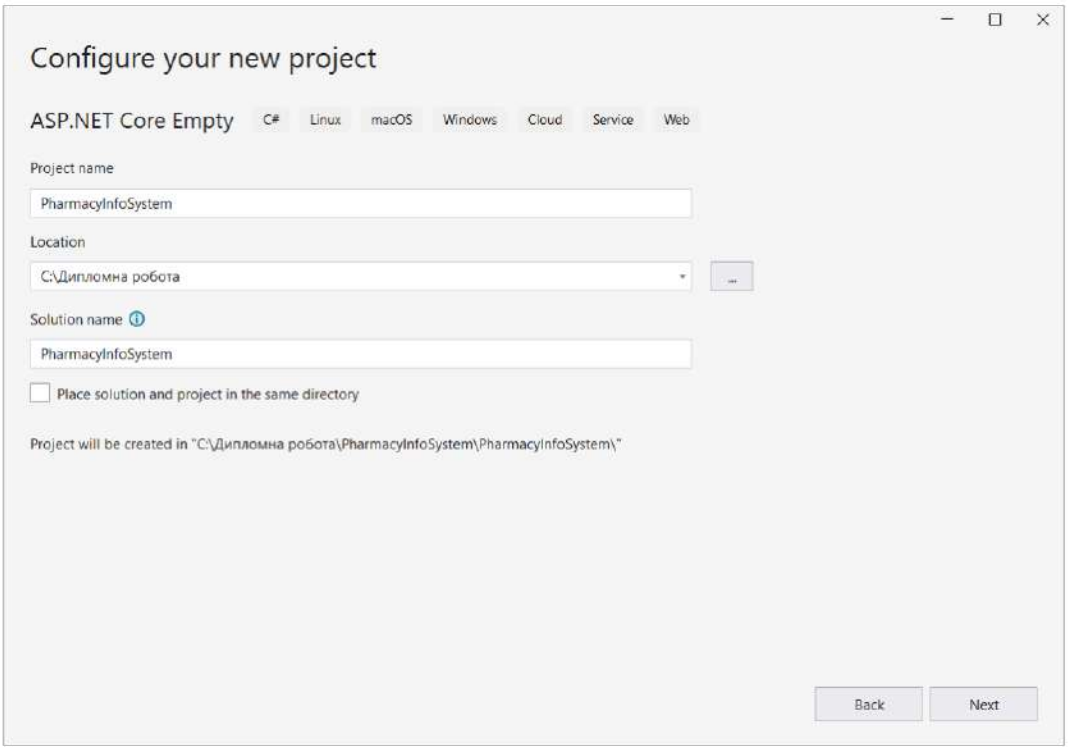

Рисунок 3.3 – Найменування проєкту та вибір папки для його збереження

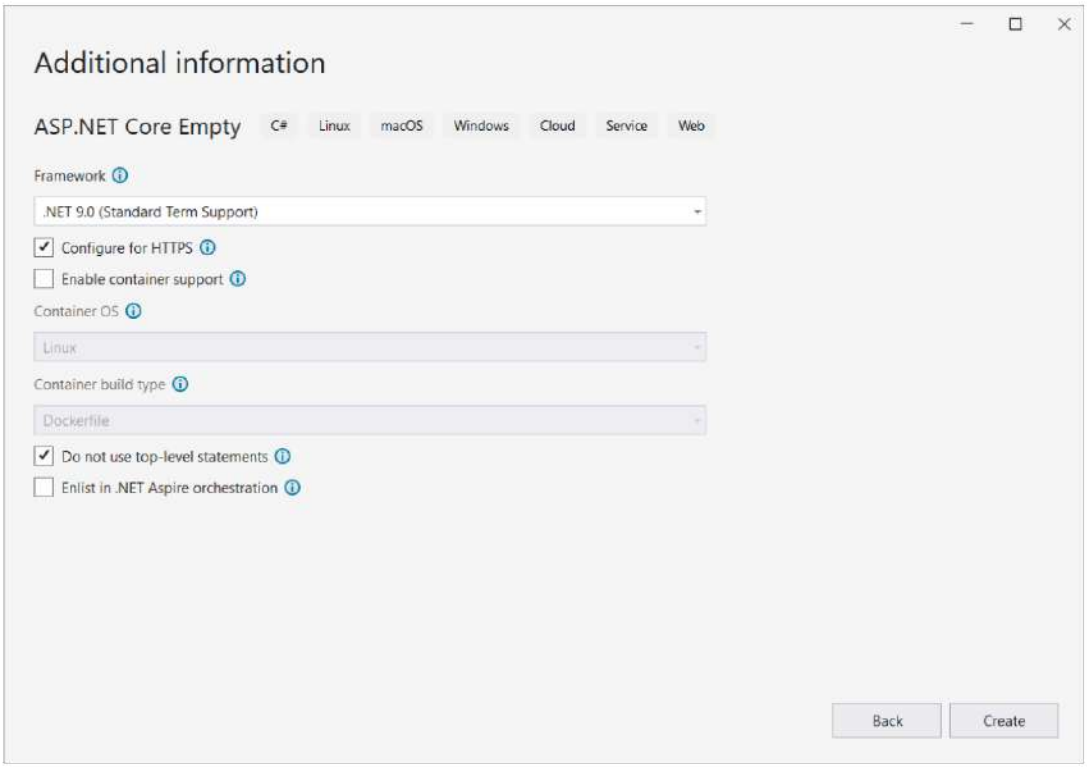

Рисунок 3.4 – Вікно з додатковою інформацією

3.2 Створення структури проєкту

Після створення нового проєкту його структура буде виглядати, як показано на рисунку 3.5.

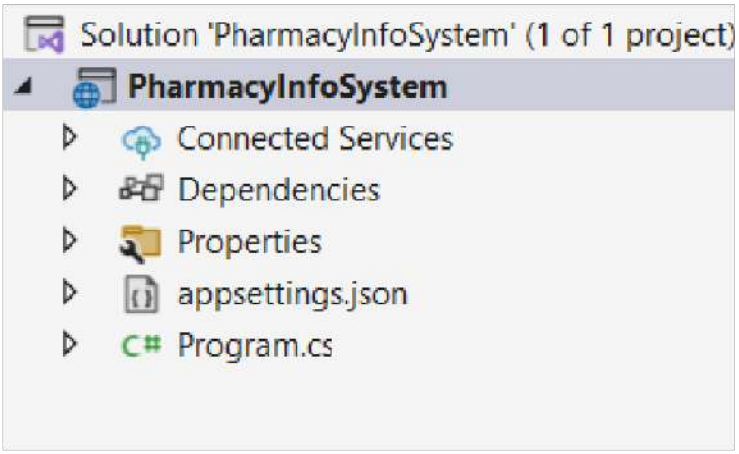

Рисунок 3.5 – Структура нового проєкту

У структурі нового проєкту на даний момент є лише кілька елементів (папок та файлів):

– папка Connected Services використовується для додавання зовнішніх служб;

– папка Dependencies містить зовнішні бібліотеки та пакети, які використовує проєкт;

– папка Properties містить файли налаштувань проєкту;

– файл appsettings.json – це файл конфігурації, в якому зберігаються налаштування додатку;

– файл Program.cs – це головний файл програми, з якого запускається додаток;

Перші три елементи при розробці проєкту створюються самим середедовищем, і в подальшому їх вміст ми змінювати не будемо.

Далі приступаємо до безпосередньої розробки структури нашого проєкту. З цією метою нам потрібно додати папки для моделей (Models), для представлень (Views), для контролерів (Controllers), папку, в якій будуть зберігатися стилі та

картинки (wwwroot), а також папку Services для програм обробки даних користувачів та ліків, яка працюватиме у режимах адміністратора та звичайного клієнта. Після додавання нових елементів структура буде виглядати, як показано на рисунку 3.6.

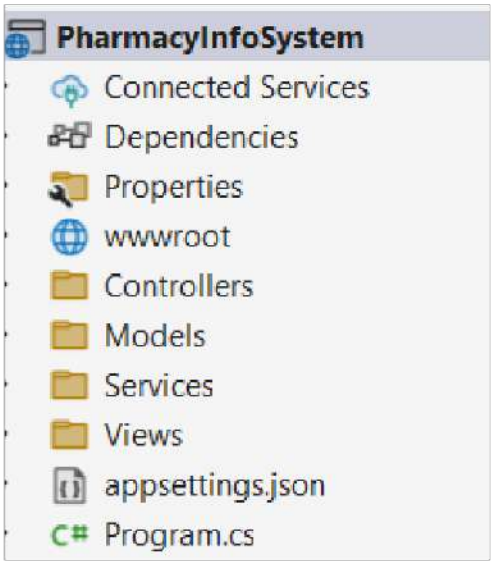

Рисунок 3.6 – Структура проєкту після додавання нових елементів

3.3 Створення та налаштування контролерів (Controllers) проєкту

Контролери в ASP.NET Core MVC є одним із ключових компонентів, які відповідають за обробку HTTP-запитів, взаємодію з моделями та повернення відповідних представлень (View) [12].

У нашому проєкті є два основних контролери – AdminController.cs та HomeController.cs, які потрібно створити в папці Controllers.

Для створення контролера HomeController.cs натискаємо правою кнопкою миші по папці Controllers, натискаємо Add/Class, після чого відкриється вікно додавання файлів (рис. 3.7). У цьому вікні натискаємо вкладку Class, вводимо ім’я контролера HomeController.cs, після чого натискаємо кнопку Add.

Для створення контролера AdminController.cs слід зробити теж саме. Як бачимо, обидва контролери є класами. Методи цих класів повинні забезпечити потрібний для розробника функціонал.

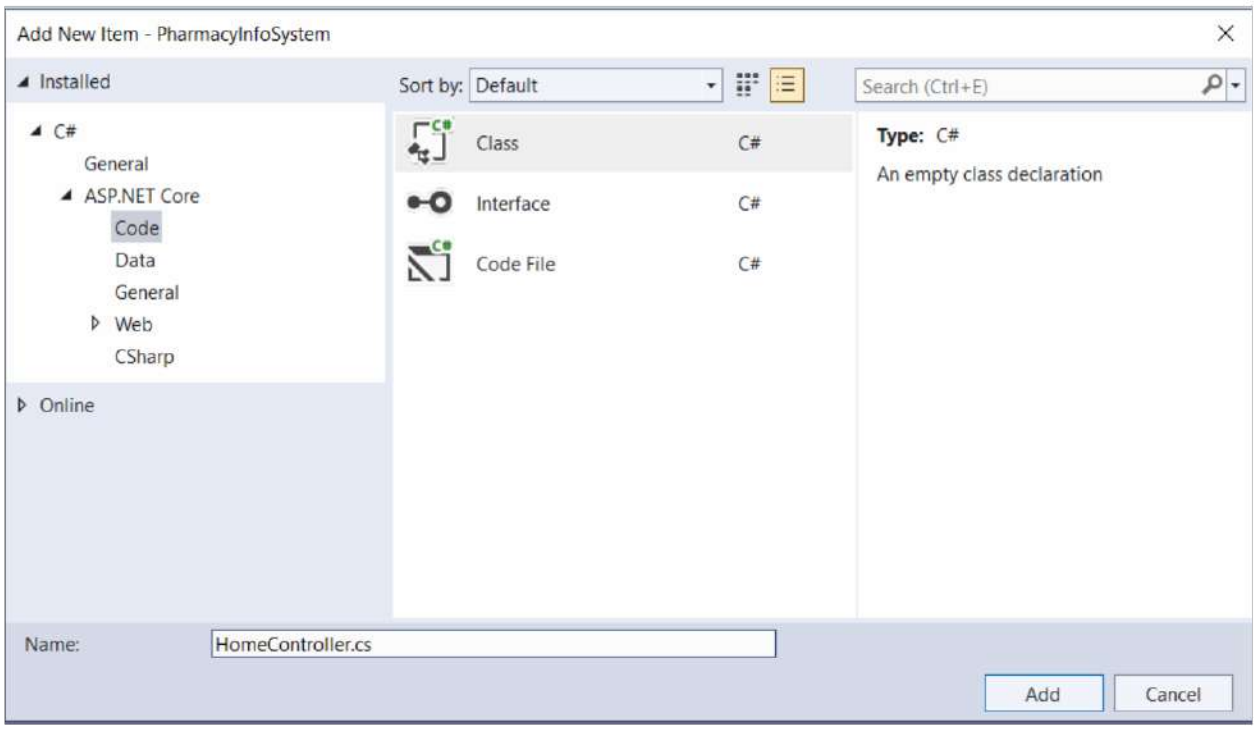

Рисунок 3.7 – Вікно додавання файлів

Контролер HomeController.cs відповідає за керування головною сторінкою, реєстрацією, авторизацію клієнтів та відображенням списку ліків.

Конструктор HomeController (PharmacyService pharmacyService) контролера HomeController.cs отримує екземпляр класу PharmacyService через ін'єкцію залежностей (рис. 3.8).

```
public class HomeController : Controller
{
    private readonly PharmacyService _pharmacyService;

    public HomeController(PharmacyService pharmacyService)
    {
        _pharmacyService = pharmacyService;
    }
```

Рисунок 3.8 – Конструктор отримує екземпляр класу PharmacyService

Далі розглянемо методи класу HomeController.cs.

Метод IActionResult Index() контролера HomeController повертає головну сторінку. Він отримує список ліків через об'єкт _pharmacyService і передає їх у

представлення (рис. 3.9).

```
public IActionResult Index()
{
    var medicines = _pharmacyService.Medicines;
    return View(medicines);
}
```

Рисунок 3.9 – метод який повертає головну сторінку.

Метод IActionResult Register() відображає форму реєстрації клієнта (рис. 3.10).

```
[HttpGet]
public IActionResult Register()
{
    return View();
}
```

Рисунок 3.10 – Метод IActionResult Register() з HTTP-методом GET

Метод IActionResult Register() з HTTP-методом POST обробляє дані форми реєстрації клієнта (рис. 3.11).

```
[HttpPost]
public IActionResult Register(Customer customer)
{
    if (ModelState.IsValid)
    {
        _pharmacyService.AddCustomer(customer);
        HttpContext.Session.SetString("Phone", customer.Phone);
        return RedirectToAction("Medicines");
    }

    return View(customer);
}
```

Рисунок 3.11 – Метод IActionResult Register() з HTTP-методом POST

Метод IActionResult LoginC() з HTTP-методом GET відображає форму авторизації (рис. 3.12).

```
[HttpGet]
public IActionResult LoginC()
{
    return View();
}
```

Рисунок 3.12 – Метод IActionResult LoginC() з HTTP-методом GET

Метод IActionResult LoginC() з HTTP-методом POST обробляє авторизацію клієнта (рис. 3.13). Він отримує введені користувачем ім’я (name) та номер телефону (phoneNumber), перевіряє їх у списку зареєстрованих клієнтів (_pharmacyService.Customers).

```
[HttpPost]
public IActionResult LoginC(string name, string phoneNumber)
{

    var customer = _pharmacyService.Customers
    .FirstOrDefault(c => c.Name == name && c.Phone == phoneNumber);

    if (customer != null)
    {

        HttpContext.Session.SetString("Phone", phoneNumber);
        return RedirectToAction("Medicines");
    }
    else
    {

        ViewBag.ErrorMessage = "Невірне ім'я або номер телефону.";
        return View();
    }
}
```

Рисунок 3.13 – Метод IActionResult LoginC()з HTTP-методом POST

Метод IActionResult Medicines() відображає список ліків, а також дозволяє шукати та сортувати їх (рис. 3.14). Він перевіряє, чи користувач авторизований через сесію (HttpContext.Session.GetString("Phone")). Якщо ні, виводить повідомлення про необхідність авторизації.

```
public IActionResult Medicines(string searchTerm, string sortOrder)
{

    var phone = HttpContext.Session.GetString("Phone");
    if (string.IsNullOrEmpty(phone) ||
    !_pharmacyService.IsCustomerRegistered(phone))
    {
        ViewBag.ErrorMessage = "Для перегляду ліків потрібно авторизуватися.";
        return View();
    }

    var medicines = _pharmacyService.Medicines;
```

```
//Пошук
if (!string.IsNullOrEmpty(searchTerm))
{
    medicines = medicines.Where(m => m.Name.Contains(searchTerm,
    StringComparison.OrdinalIgnoreCase) ||
    m.Description.Contains(searchTerm,
    StringComparison.OrdinalIgnoreCase)).ToList();
}
// Перевіряємо, чи є ліки після пошуку
if (medicines == null || !medicines.Any())
{
    ViewBag.ErrorMessage = "Ліки не знайдені.";
}
```

```
// Сортування
switch (sortOrder)
{
    case "name_asc":
        medicines = medicines.OrderBy(m => m.Name).ToList();
        break;
    case "name_desc":
        medicines = medicines.OrderByDescending(m => m.Name).ToList();
        break;
    case "price_asc":
        medicines = medicines.OrderBy(m => m.Price).ToList();
        break;
    case "price_desc":
        medicines = medicines.OrderByDescending(m => m.Price).ToList();
        break;
    default:
        break;
}
ViewData["SortOrder"] = sortOrder;
ViewData["SearchTerm"] = searchTerm;
return View(medicines);
```

Рисунок 3.14 – Метод IActionResult Medicines()

HomeController координує процес реєстрації, авторизації клієнтів та відображення списку ліків. Він виконує лише керуючі функції, а обробку даних передає сервісу PharmacyService. (Повний код HomeController в додатку А).

Контроллер AdminController.cs відповідає за керування панеллю адміна (додавання ліків, перегляд клієнтів тощо).

Конструктор AdminController приймає PharmacyService через ін'єкцію залежностей і перевіряє, щоб він не був null, інакше викидає виняток (рис. 3.15).

```
public class AdminController : Controller
{
    private readonly PharmacyService _pharmacyService;

    public AdminController(PharmacyService pharmacyService)
    {
        _pharmacyService = pharmacyService ?? throw new
        ArgumentNullException(nameof(pharmacyService));
    }
```

Рисунок 3.15 – Конструктор AdminController

Метод IActionResult Dashboard() перевіряє, чи адміністратор залогінений через сесію, отримує список ліків, а якщо вказано searchTerm, виконує пошук за назвою або описом, а потім передає список ліків у представлення (рис. 3.16).

```
[HttpGet]
public IActionResult Dashboard(string searchTerm)
{

    if (HttpContext.Session.GetString("IsAdmin") != "true")
    {
        return RedirectToAction("Login");
    }
    var medicines = _pharmacyService.GetMedicines();
    if (!string.IsNullOrEmpty(searchTerm))
    {
        medicines = medicines.Where(m => m.Name.Contains(searchTerm,
        StringComparison.OrdinalIgnoreCase) ||
        m.Description.Contains(searchTerm,
        StringComparison.OrdinalIgnoreCase)).ToList();
    }
    foreach (var medicine in medicines)
    {
        Console.WriteLine($"Medicine in dashboard: {medicine.Name},
        {medicine.Description}");
    }
    ViewData["SearchTerm"] = searchTerm;
    return View(medicines);
}
```

Рисунок 3.16 – Метод IActionResult Dashboard()

Метод IActionResult CustomerList() отримує список зареєстрованих клієнтів із PharmacyService і передає його у представлення (рис. 3.17).

```
public IActionResult CustomerList()
{
    var customers = _pharmacyService.Customers;
    return View(customers);
}
```

Рисунок 3.17 – Метод IActionResult CustomerList()

Метод IActionResult AddMedicine() з HTTP-методом GET перевіряє, чи адміністратор залогінений, якщо так, відображає сторінку додавання ліків, якщо ні, перенаправляє на сторінку входу (рис. 3.18).

```
[HttpGet]
public IActionResult AddMedicine()
{
    if (HttpContext.Session.GetString("IsAdmin") != "true")
    {
        return RedirectToAction("Login");
    }
    return View();
}
```

Рисунок 3.18 – Метод IActionResult AddMedicine() з HTTP-методом GET

Метод AddMedicine() з HTTP-методом POST приймає модель Medicine і зображення, перевіряє авторизацію адміністратора, виводить інформацію у консоль, перевіряє, чи зображення є коректним файлом, генерує унікальне ім’я для нього, зберігає у папці wwwroot/images, оновлює шлях до зображення в моделі, перевіряє коректність введених даних і якщо все добре, додає ліки у PharmacyService, а потім перенаправляє на Dashboard.

Крім того, метод перевіряє наявність папки для зображень, що запобігає можливим помилкам під час збереження файлу. У разі виникнення виняткових ситуацій при завантаженні зображення, в консоль виводиться повідомлення про помилку, а користувач отримує відповідне повідомлення через ModelState.

Якщо модель містить некоректні дані, користувач залишається на сторінці

додавання ліків, де може виправити введену інформацію. Після успішного додавання нового препарату оновлений список ліків можна переглянути в панелі адміністратора, всі його дані зберігаються у JSON-файлі, що дозволяє зберігати інформацію між перезапусками програми (рис. 3.19-3.20).

```
[HttpPost]
public async Task<IActionResult> AddMedicine(Medicine medicine,
IFormFile image)
{
    Console.WriteLine("Метод AddMedicine викликано");

    if (HttpContext.Session.GetString("IsAdmin") != "true")
    {
        return RedirectToAction("Login");
    }

    Console.WriteLine($"Received medicine: {medicine.Name},
    {medicine.Description}, {medicine.Price}");
```

```
if (image == null || image.Length == 0)
{
    Console.WriteLine("Зображення не вибрано або файл пустий.");
    ModelState.AddModelError("ImageUrl", "Зображення є обов'язковим.");
}
else if (image.ContentType.StartsWith("image"))
{
    Console.WriteLine($"Зображення отримано: {image.FileName},
    Тип: {image.ContentType}");

    string fileName = Guid.NewGuid().ToString() +
    Path.GetExtension(image.FileName);
    var filePath = Path.Combine(Directory.GetCurrentDirectory(),
    "wwwroot/images", fileName);

    Console.WriteLine($"Унікальне ім'я для зображення згенеровано:
    {fileName}");

    var imageDirectory = Path.Combine(Directory.GetCurrentDirectory(),
    "wwwroot/images");
    if (!Directory.Exists(imageDirectory))
    {
        Console.WriteLine($"Папка {imageDirectory} не існує, створюємо
        її...");
        Directory.CreateDirectory(imageDirectory);
    }
    else
    {
        Console.WriteLine($"Папка для зображень існує: {imageDirectory}");
    }
```

Рисунок 3.19 – Метод IActionResult AddMedicine з методом POST (початок)

```
try
{
   using (var stream = new FileStream(filePath, FileMode.Create))
   {
     await image.CopyToAsync(stream);
   }
   Console.WriteLine($"Зображення збережено за шляхом: {filePath}");

   medicine.ImageUrl = $"/images/{fileName}";
   Console.WriteLine($"Присвоєно шлях зображення до моделі:
   {medicine.ImageUrl}");
   ModelState.Remove("ImageUrl");
 }
 catch (Exception ex)
 {
   Console.WriteLine($"Помилка при збереженні зображення:
   {ex.Message}");
   ModelState.AddModelError("ImageUrl", "Помилка при збереженні
   зображення.");
 }
}
else
{
    Console.WriteLine($"Невірний формат файлу: {image.ContentType}");
    ModelState.AddModelError("ImageUrl", "Файл не є зображенням.");
}
```

```
 foreach (var modelError in ModelState.Values.SelectMany(v => v.Errors))
 {
     Console.WriteLine($"Помилка: {modelError.ErrorMessage}");
 }

 if (!ModelState.IsValid)
 {
     Console.WriteLine("Модель некоректна, повертаємо на форму.");
     return View(medicine);
 }

 _pharmacyService.AddMedicine(medicine);

 var medicines = _pharmacyService.GetMedicines();
 foreach (var med in medicines)
 {
     Console.WriteLine($"Medicine added to memory: {med.Name},
     {med.Description}, {med.ImageUrl}");
 }

 return RedirectToAction("Dashboard");
}
```

Рисунок 3.20 – Метод IActionResult AddMedicine з методом POST (закінчення)

Метод IActionResult Login з HTTP-методом GET повертає сторінку входу (рис. 3.21).

```
[HttpGet]
public IActionResult Login()
{
    return View();
}
```

Рисунок 3.21 – Метод IActionResult Login з HTTP-методом GET

Метод IActionResult Login з HTTP-методом POST перевіряє, чи введені логін і пароль співпадають із admin/admin, якщо так, то перенаправляє на Dashboard, а якщо ні, передає повідомлення про помилку (рис. 3.22).

```
[HttpPost]
public IActionResult Login(string username, string password)
{
    if (username == "admin" && password == "admin")
    {
        HttpContext.Session.SetString("IsAdmin", "true");
        return RedirectToAction("Dashboard");

    }

    ViewBag.ErrorMessage = "Невірний логін або пароль";
    return View();
}
```

Рисунок 3.22 – метод Login з HTTP-методом POST.

Метод Delete отримує id ліків, знаходить відповідний об’єкт у списку, після чого викликає DeleteMedicine у PharmacyService і видаляє ліки (рис. 3.23).

```
[HttpPost]
public JsonResult Delete(int id)
{
    var medicineToRemove = _pharmacyService.GetMedicines()
    .FirstOrDefault(m => m.Id == id);
    if (medicineToRemove != null)
    {
        _pharmacyService.DeleteMedicine(id);
        return Json(new { success = true });
    }
    return Json(new { success = false });
}
```

Рисунок 3.23 – Метод IActionResult Delete

AdminController реалізує функціонал адміністратора для керування списком ліків, реєстрацією клієнтів, входом у систему та видаленням ліків (повний код AdminController в додатку Б).

3.4 Створення та налаштування моделей (Models) проєкту

Моделі в ASP.NET Core MVC відповідають за представлення даних на сайті [13].

У нашому проєкті моделі описують структуру даних, які використовуються для збереження інформації про клієнтів в класі Customer.cs, та ліків в класі Medicine.cs.

В папці Models нам потрібно створити два класи – Medicine.cs (рис. 3.24) і Customer.cs (рис. 3.25).

```
namespace PharmacyInfoSystem.Models
{
    public class Medicine
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public decimal Price { get; set; }
        public string? ImageUrl { get; set; }
    }
}
```

Рисунок 3.24 – Клас Medicine.cs

Модель Medicine.cs є основою для збереження інформації про ліки. Вона містить основні дані: назву, опис, ціну та шлях до зображення.

```
namespace PharmacyInfoSystem.Models
{
    public class Customer
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Phone { get; set; }
    }
}
```

Рисунок 3.25 – Клас Customer.cs.

Модель Customer.cs містить інформацію про клієнтів, які зареєструвалися в системі. Вона використовується для зберігання ідентифікатора клієнта, його імені та номера телефону, що дозволяє вести облік користувачів аптеки.

3.5 Створення та налаштування представлень (Views) проєкту

Представлення (Views) в ASP.NET Core MVC відповідають за відображення даних користувачеві [14].

Перед початком створення представленнь нам потрібно створити в папці Views кілька папок, які будуть працювати з різними контролерами. Створимо папки Admin, Home і Shared. В папці Admin створимо представлення AddMedicine.cshtml. Для його створення натискаємо правою кнопкою миші по папці Admin, виконуємо Add/Class; відкриється вікно додавання файлів (рис. 3.26), натискаємо Web, вибираємо Razor View – Empty, називаємо файл і натискаємо Add. Так само створимо ще три файли в папці Admin: CustomerList.cshtml, Dashboard.cshtml і Login.cshtml. В папці Home створюємо Index.cshtml, Medicines.cshtml, Register.cshtml і LoginC.cshtml. В папці Shared потрібно створити файл _Layout.cshtml, а в папці Views потрібно створити файл _ViewStart.cshtml.

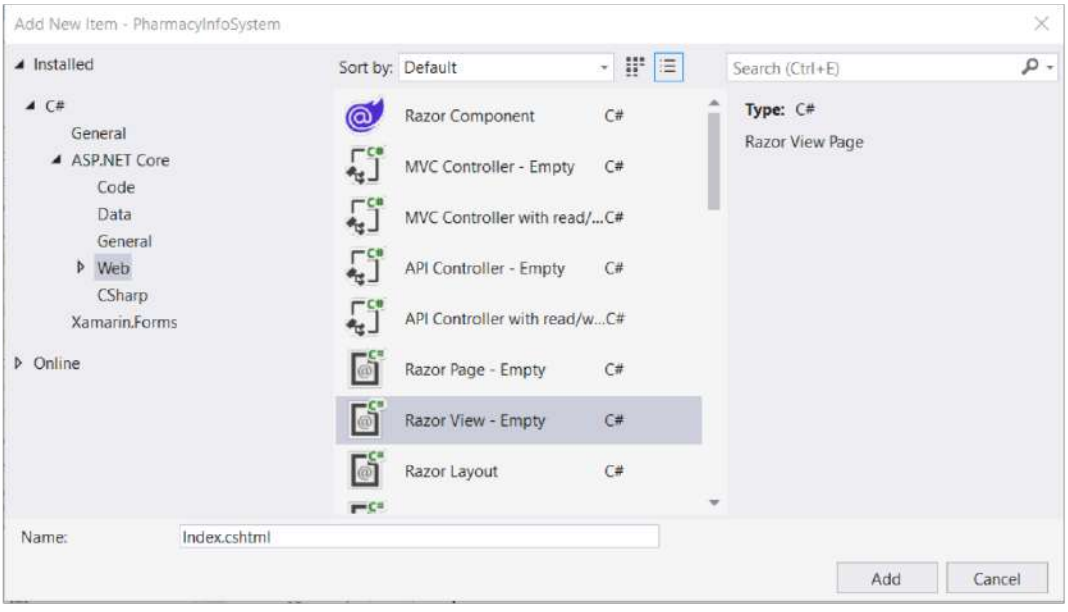

Рисунок 3.26 – Вікно додавання файлів(Razor View – Empty)

Тепер, коли ми створили всі необхідні представлення, слід налаштувати їх відповідно до потреб проєкту.

Представлення AddMedicine.cshtml відповідає за додавання ліків до списку (рис. 3.27-3.28).

```
@model PharmacyInfoSystem.Models.Medicine
<h2>Додавання ліків</h2>
<form asp-action="AddMedicine" method="post"
enctype="multipart/form-data">
    <div>
        <label for="Name">Назва:</label>
        <input type="text" id="Name" name="Name"
value="@Model?.Name" required />
    </div>
    <div>
        <label for="Description">Опис:</label>
        <input type="text" id="Description" name="Description"
        value="@Model?.Description" required />
    </div>
    <div>
        <label for="Price">Ціна:</label>
        <input type="number" id="Price" name="Price"
value="@Model?.Price"
        step="0.01" required />
    </div>
    <div class="form-group">
        <label for="image">Зображення:</label>
        <input type="file" class="form-control" id="image" name="image"
        accept="image/*" required />
        <span asp-validation-for="image" class="text-danger"></span>
    </div>
    @if (!ViewData.ModelState.IsValid)
    {
     <div class="alert alert-danger">
      <ul>
       @foreach (var error in ViewData.ModelState.Values
       .SelectMany(v => v.Errors))
       {
         <li>@error.ErrorMessage</li>
       }
      </ul>
     </div>
    }
    <button type="submit">Додати ліки</button>
</form>

<a href="/Admin" class="btn btn-primary">Повернутися до списку</a>
```

Рисунок 3.27 – Код представлення AddMedicine.cshtml (початок)

```
<script>
    const form = document.querySelector("form");
    form.addEventListener("submit", function(event) {
        const imageInput = document.querySelector("input[name='image']");
        if (!imageInput.files.length) {
            alert("Будь ласка, виберіть зображення.");
            event.preventDefault();
        }
    });
</script>
```

Рисунок 3.28 – Код представлення AddMedicine.cshtml (закінчення)

Представлення AddMedicine.cshtml дозволяє адміністратору додавати нові ліки з усіма необхідними параметрами. Воно забезпечує зручний інтерфейс введення інформації про ліки та її подальшого збереження. Завдяки використанню JavaScript перевірка відбувається ще до відправлення даних, що підвищує ефективність роботи адміністративної панелі.

Також передбачена обробка можливих помилок через ModelState, що дозволяє виводити повідомлення про некоректно заповнені поля безпосередньо у веб-інтерфейсі. Використання атрибутів required забезпечує базову перевірку введених даних ще на рівні браузера. Кнопка повернення на сторінку адміністративної панелі дозволяє швидко скасувати введення та повернутися до списку ліків без оновлення сторінки.

Представлення CustomerList.cshtml показує табличку з ктієнтами (рис. 3.29).

```
@model IEnumerable<PharmacyInfoSystem.Models.Customer>
<a href="/Admin" class="btn btn-primary">Повернутися до панелі адміністратора</a>
<h2>Список клієнтів</h2>
<table class="table table-striped table-bordered">
    <thead>
        <tr>
            <th>Ім'я</th>
            <th>Телефон</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var customer in Model)
        {
            <tr>
                <td>@customer.Name</td>
                <td>@customer.Phone</td>
            </tr>
        }
    </tbody>
</table>
```

Рисунок 3.29 – Представлення CustomerList.cshtml

Представлення CustomerList.cshtml дозволяє адміністратору переглядати список зареєстрованих клієнтів у табличному вигляді. Це представлення є важливим для адміністратора, оскільки дозволяє контролювати реєстрацію клієнтів і працювати з їхніми даними.

Представлення Dashboard.cshtml показує панель адміністратора (рис. 3.30-3.31).

```
@model IEnumerable<PharmacyInfoSystem.Models.Medicine>
@using PharmacyInfoSystem.Models

<h2>Панель адміністратора</h2>
<a href="/Admin/AddMedicine" class="btn btn-primary">Додати ліки</a>
<a href="/Admin/CustomerList" class="btn btn-primary">Переглянути клієнтів</a>
<h3>Список ліків</h3>

<form method="get" action="/Admin/Dashboard">
    <input type="text" name="searchTerm" value=
    "@ViewData["SearchTerm"]"  placeholder="Пошук ліків..." />
    <button type="submit" class="btn">Пошук</button>
</form>
```

```
<div class="medicine-list">
 @foreach (var medicine in Model)
 {
   <div class="medicine-item" id="medicine-@medicine.Id">
      <div class="medicine-img">
           @if (!string.IsNullOrEmpty(medicine.ImageUrl))
           {
               <img src="@medicine.ImageUrl" alt="@medicine.Name" />
           }
      </div>
      <div class="medicine-details">
          <h4 class="medicine-name">@medicine.Name</h4>
          <p class="medicine-description">@medicine.Description</p>
          <span class="medicine-price">@medicine.Price грн</span>
      </div>
      <div class="medicine-actions">
          <a href="#" onclick="deleteMedicine(@medicine.Id)"
          class="delete-btn">Видалити</a>
      </div>
    </div>
  }
</div>
```

Рисунок 3.30 – Представлення Dashboard.cshtml (початок)

```
<div class="medicine-list">
 @foreach (var medicine in Model)
 {
   <div class="medicine-item" id="medicine-@medicine.Id">
      <div class="medicine-img">
           @if (!string.IsNullOrEmpty(medicine.ImageUrl))
           {
               <img src="@medicine.ImageUrl" alt="@medicine.Name" />
           }
      </div>
      <div class="medicine-details">
          <h4 class="medicine-name">@medicine.Name</h4>
          <p class="medicine-description">@medicine.Description</p>
          <span class="medicine-price">@medicine.Price грн</span>
      </div>
      <div class="medicine-actions">
          <a href="#" onclick="deleteMedicine(@medicine.Id)"
          class="delete-btn">Видалити</a>
      </div>
    </div>
  }
</div>
```

```
<script>
  function deleteMedicine(id) {
    if (confirm("Ви впевнені, що хочете видалити цей препарат?")) {
        fetch(`/Admin/Delete?id=${id}`, {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            }
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                document.getElementById(`medicine-${id}`).remove();
                alert("Ліки успішно видалено!");
            } else {
                alert("Не вдалося видалити ліки!");
            }
        })
        .catch(() => alert("Помилка при видаленні!"));
    }
  }
</script>
```

Рисунок 3.31 – Представлення Dashboard.cshtml (закінчення)

Представлення Dashboard.cshtml забезпечує зручний інтерфейс для адміністратора аптеки. Він дозволяє: переглядати ліки, зображення, опис і ціну; додавати нові ліки; видаляти ліки без перезавантаження сторінки завдяки використанню JavaScript для взаємодії з сервером через асинхронні запити. Це

спрощує роботу адміністратора, роблячи інтерфейс зручним для управління ліками.

Представлення Login.cshtml для авторизації адміна (рис. 3.32).

```
@{
    ViewData["Title"] = "Login";
}
<h2>Підтвердження</h2>
<form method="post">
    <div>
        <label for="username">Логін:</label>
        <input type="text" id="username" name="username" required />
    </div>
    <div>
        <label for="password">Пароль:</label>
        <input type="password" id="password" name="password" required />
    </div>
    <div>
        <button type="submit">Увійти</button>
    </div>
</form>
@if (ViewBag.ErrorMessage != null)
{
    <div style="color: red;">
        @ViewBag.ErrorMessage
    </div>
}
```

Рисунок 3.32 – Представлення Login.cshtml

Представлення Login.cshtml є ефективним і простим інтерфейсом для авторизації користувачів. Вона включає поля для введення логіну (admin) та пароля (admin), а також обов'язкову перевірку заповнення цих полів перед відправленням форми. У разі невдалого входу, користувач отримує повідомлення про помилку, що дозволяє зручніше взаємодіяти з системою.

Index.cshtml – це представлення головної сторінки (рис. 3.33).

```
@{
    ViewData["Title"] = "Аптека";
}
<h1>@ViewData["Title"]</h1>
```

Рисунок 3.33 – Представлення Index.cshtml

Представлення Index.cshtml виводить надпис «Аптека» на головну сторінку.

LoginC.cshtml виконує роль представлення для авторизації клієнтів (рис. 3.34).

```
@{
    ViewData["Title"] = "Авторизація";
}
<h2>Авторизація</h2>
<form method="post">
  <div>
      <label for="name">Ім'я:</label>
      <input type="text" id="name" name="name" required />
  </div>
  <div>
      <label for="phoneNumber">Номер телефону:</label>
      <input type="text" id="phoneNumber" name="phoneNumber" required />
  </div>
  <button type="submit">Увійти</button>
</form>
@if (ViewBag.ErrorMessage != null)
{
    <div style="color:red; margin-top: 10px;">
        @ViewBag.ErrorMessage
    </div>
}
```

Рисунок 3.34 – Представлення LoginC.cshtml

Представлення LoginC.cshtml – це форма авторизації, яка приймає ім’я та номер телефону користувача та передає їх у контролер для перевірки. Якщо дані правильні, користувач авторизується, а якщо ні – отримує повідомлення про помилку.

Представлення Medicines.cshtml паказує список ліків (рис. 3.35-3.36).

```
@model IEnumerable<PharmacyInfoSystem.Models.Medicine>
<a href="/Home" class="btn btn-primary">Повернутися на головну сторінку</a>
<h3>Список ліків</h3>
<form method="get" action="/Home/Medicines">
  <input type="text" name="searchTerm" value="@ViewData["SearchTerm"]"
  placeholder="Пошук ліків..." />
  <button type="submit" class="btn">Пошук</button>
</form>
```

Рисунок 3.35 – Представлення Medicines.cshtml (початок)

```
<form method="get" action="/Home/Medicines">
 <div>
  <label for="sortOrder">Сортувати за:</label>
  <select name="sortOrder" id="sortOrder" onchange="this.form.submit()">
    <option value="" @(string.IsNullOrEmpty(ViewData["SortOrder"]?
    .ToString()) ? "selected" : "")>Замовчуванням</option>
    <option value="name_asc" @(ViewData["SortOrder"]?
    .ToString() == "name_asc" ? "selected" : "")>Назвою (А-Я)</option>
    <option value="name_desc" @(ViewData["SortOrder"]?
    .ToString() == "name_desc" ? "selected" : "")>Назвою (Я-А)</option>
    <option value="price_asc" @(ViewData["SortOrder"]?
    .ToString() == "price_asc" ? "selected" : "")>Ціною (по зростанню)</option>
    <option value="price_desc" @(ViewData["SortOrder"]?
    .ToString() == "price_desc" ? "selected" : "")>Ціною (по спаданню)</option>
  </select>
 </div>
</form>
```

```
@if (ViewBag.ErrorMessage != null)
{
    <div class="alert alert-danger">
        @ViewBag.ErrorMessage
    </div>
}
@if (Model != null && Model.Any())
{
 <div class="medicine-list">
  @foreach (var medicine in Model)
  {
   <div class="medicine-item">
    <div class="medicine-img">
        @if (!string.IsNullOrEmpty(medicine?.ImageUrl))
        {
         <img src="@medicine.ImageUrl" alt="@medicine.Name" />
        }
    </div>
    <div class="medicine-details">
      <div class="medicine-name">@medicine.Name</div>
      <div class="medicine-description">@medicine.Description</div>
      <span class="medicine-price">@medicine.Price грн</span>
    </div>
   </div>
  }
 </div>
}
```

Рисунок 3.36 – Представлення Medicines.cshtml (закінчення)

Представлення Medicines.cshtml виводить список ліків з можливістю пошуку та сортування. Користувач може бачити інформацію про кожен препарат, включаючи назву, опис, ціну та зображення. У разі виникнення помилки з’являється відповідне повідомлення.

Представлення Register.cshtml – це представлення для реєстрації клієнта (рис. 3.37).

```
@model PharmacyInfoSystem.Models.Customer
@{
    ViewData["Title"] = "Реєстрація";
}
<h2>Реєстрація клієнта</h2>
<form method="post">
 <div class="form-group">
  <label for="Name">Ім'я</label>
  <input type="text" class="form-control" id="Name" name="Name" required />
 </div>
 <div class="form-group">
  <label for="Phone">Номер телефону</label>
  <input type="text" class="form-control" id="Phone" name="Phone" required />
 </div>
 <button type="submit" class="btn btn-primary">Зареєструватися</button>
</form>
```

Рисунок 3.37 – Представлення Register.cshtml

Представлення Register.cshtml є простим інтерфейсом для реєстрації клієнтів. Він дозволяє користувачу ввести ім’я та номер телефону для створення нового запису в системі.

Представлення _Layout.cshtml – це шаблонний макет, який використовується у всіх сторінках сайту (рис. 3.38-3.39).

```
<!DOCTYPE html>
<html lang="uk">
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>@ViewData["Title"]</title>
 <link href="~/css/site.css" rel="stylesheet" />
</head>
<script>
 window.onload = function () {
   var audio = new Audio("/sounds/pianosounds.wav");
   audio.loop = true;
    var allowSound = localStorage.getItem("allowSound");
    if (allowSound === null) {
       var userChoice = confirm("Чи дозволяєте ви відтворення звуку?");
       if (userChoice) {
          localStorage.setItem("allowSound", "true");
          audio.play().catch(error => {
              console.log("Автоматичне відтворення заблоковано: ", error);
              alert("Автоматичне відтворення звуку заблоковано вашим браузером.
              Будь ласка, увімкніть звук вручну.");
           });
```

Рисунок 3.38 – Представлення _Layout.cshtml (початок)

```
        } else {
            localStorage.setItem("allowSound", "false");
        }
    } else if (allowSound === "true") {
        audio.play().catch(error => {
            console.log("Автоматичне відтворення заблоковано: ", error);
            alert("Автоматичне відтворення звуку заблоковано вашим браузером.
            Будь ласка, увімкніть звук вручну.");
        });
    } else {
        console.log("Звук відключено користувачем.");
    }
  };
  function resetSoundPermission() {
      localStorage.removeItem("allowSound");
      location.reload();
  }
</script>
```

```
<body>
  <header class="header">
    <div class="logo">PharmacyInfoSystem</div>
    <button class="menu-btn" onclick="toggleMenu()">☰</button>
    <nav class="menu">
      <ul>
        <li><a href="/Home">Головна сторінка</a></li>
        <li><a href="/Home/Medicines">Переглянути список ліків</a></li>
        <li><a href="/Home/Register">Реєстрація</a></li>
        <li><a href="/Home/LoginC">Авторизація</a></li>
        <li><a href="/Admin">Куточок адміна</a></li>
        <li><a href="javascript:void(0)" onclick="resetSoundPermission()">Звук</a></li>
      </ul>
    </nav>
</header>
```

```
<div class="container">
  @RenderBody()
</div>

<footer class="footer">
  <div class="footer-content">
      <p id="currentTime"></p>
      <p>Виконав: КІ-41 Янковський Богдан </p>
  </div>
</footer>
<style>
  .footer {
      background-color: white;
      color: black;
      text-align: center;
      padding: 10px;
      position: relative;
      bottom: 0;
      width: 100%;
      border-top: 1px solid #ddd;
      height: 60px;
      line-height: 60px;
  }
</style>
```

Рисунок 3.39 – Представлення _Layout.cshtml (продовження)

Закінчення представлення _Layout.cshtml зображено на рисунку 3.40.

```
    <script>
      function updateTime() {
          let now = new Date();
          let formattedTime = now.toLocaleString('uk-UA', {
              year: 'numeric',
              month: '2-digit',
              day: '2-digit',
              hour: '2-digit',
              minute: '2-digit',
              second: '2-digit'
          });
          document.getElementById("currentTime").innerText = "" + formattedTime;
      }
        updateTime();
        setInterval(updateTime, 1000);
    </script>
    <script>
        function toggleMenu() {
            var menu = document.querySelector(".menu");
            if (menu.style.display === "block") {
                menu.style.display = "none";
            } else {
                menu.style.display = "block";
            }
        }
    </script>
</body>
</html>
```

Рисунок 3.40 – Представлення _Layout.cshtml (закінчення)

Файл _Layout.cshtml є основним шаблоном сайту, що визначає його загальний вигляд і структуру. Він містить заголовок сторінки, підключення стилів та скриптів, навігаційне меню, основний контейнер для контенту та футер. Однією з особливостей є вбудована система відтворення фонової музики, яка запитує дозвіл у користувача та зберігає його вибір у локальному сховищі браузера. У футері відображається поточний час, який оновлюється в реальному часі. Також реалізовано адаптивне меню, яке дозволяє зручно переглядати сайт на мобільних пристроях. Загалом, цей шаблон створює єдиний стиль для всіх сторінок сайту та робить його більш зручним і функціональним для користувачів.

_ViewStart.cshtml спрощує управління виглядом сайту, забезпечуючи єдиний стиль для всіх сторінок (рис. 3.41).

```
@{
    Layout = "_Layout";
}
```

Рисунок 3.41 – _ViewStart.cshtml.

Файл _ViewStart.cshtml встановлює макет за замовчуванням для всіх представлень у проєкті. В даному випадку він вказує, що кожна сторінка буде використовувати шаблон _Layout.cshtml. Це означає, що всі представлення автоматично підключатимуть загальну структуру сайту, включаючи шапку, меню, футер та інші спільні елементи, без необхідності вказувати макет у кожному файлі окремо.

3.6 Створення та налаштування сервісів (Services) проєкту

У проєкті ASP.NET Core MVC сервісами називаються класи, які виконують конкретні задачі або надають певну функціональність. Сервіси можуть бути використані для роботи з даними, логіки бізнес-процесів, взаємодії з API тощо. Сервіси можуть бути ін'єковані у контролери.

В папці Services потрібно створити клас PharmacyService.cs. У цьому класі оголощуємо списки для збереження даних, шляхи до файлів, змінну для генерації ID ліків, властивості для отримання списків клієнтів і ліків та створюємо конструктор класу PharmacyService(). Цей конструктор викликається під час створення об'єкта класу. Він завантажує дані про клієнтів та ліки з файлів JSON (рис. 3.42).

```
public class PharmacyService
{
    private List<Customer> _customers = new List<Customer>();
    private List<Medicine> _medicines = new List<Medicine>();

    private static readonly string DataDirectory =  Path
   .Combine(Directory.GetCurrentDirectory(), "wwwroot", "data");
    private static readonly string CustomersFile = Path
    .Combine(DataDirectory, "customers.json");
    private static readonly string MedicinesFile = Path
    .Combine(DataDirectory, "medicines.json");
    private int _nextId = 1;

    public IEnumerable<Customer> Customers => _customers;
    public IEnumerable<Medicine> Medicines => _medicines;

    public PharmacyService()
    {
        LoadCustomers();
        LoadMedicines();
    }
```

Рисунок 3.42 – Оголошення класу PharmacyService()

Метод LoadCustomers() завантажує клієнтів з фйлу JSON (рис. 3.43).

```
private void LoadCustomers()
{
  if (File.Exists(CustomersFile))
  {
    string json = File.ReadAllText(CustomersFile);
    _customers = JsonSerializer.Deserialize<List<Customer>>(json) ??
    new List<Customer>();
  }
}
```

Рисунок 3.43 – Метод LoadCustomers()

Метод SaveCustomers() зберігає клієнтів у файлі JSON (рис. 3.44).

```
private void SaveCustomers()
{
    string json = JsonSerializer.Serialize(_customers,
    new JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(CustomersFile, json);
}
```

Рисунок 3.44 – Метод SaveCustomers()

Метод IsCustomerRegistered() перевіряє, чи клієнт зареєстрований (рис. 3.45).

```
public bool IsCustomerRegistered(string phone)
{
    return _customers.Any(c => c.Phone == phone);
}
```

Рисунок 3.45 – Метод IsCustomerRegistered()

Метод AddCustomer() – це метод для додавання нового клієнта (рис. 3.46).

```
public void AddCustomer(Customer customer)
{
    _customers.Add(customer);
    SaveCustomers();
}
```

Рисунок 3.46 – Метод AddCustomer()

Метод AddMedicine() – це метод для додавання ліків (рис. 3.47).

```
public void AddMedicine(Medicine medicine)
{
    Console.WriteLine($"Adding medicine: {medicine.Name}, " +
                      $"{medicine.Description}, " +
                      $"{medicine.Price}");
    medicine.Id = _nextId++;
    _medicines.Add(medicine);
    SaveMedicines();
    Console.WriteLine($"Ліки додано: {medicine.Name}");
}
```

Рисунок 3.47 – Метод AddMedicine()

Метод GetMedicines() – це метод для отримання списку ліків (рис. 3.48).

```
public List<Medicine> GetMedicines()
{
    return _medicines;
}
```

Рисунок 3.48 – Метод GetMedicines()

Метод AuthorizeCustomer() – це метод для авторизації клієнта (рис. 3.49).

```
public string AuthorizeCustomer(string name, string phone)
{
    var customer = _customers.FirstOrDefault
        (c => c.Name == name && c.Phone == phone);
    if (customer != null)
    {
        return "Авторизація успішна";
    }
    else
    {
        return "Клієнта з таким ім'ям та номером не знайдено.";
    }
}
```

Рисунок 3.49 – Метод AuthorizeCustomer()

Метод LoadMedicines() використовується для завантаження ліків з файла JSON (рис. 3.50).

```
private void LoadMedicines()
{
 if (File.Exists(MedicinesFile))
 {
  string json = File.ReadAllText(MedicinesFile);
  _medicines = JsonSerializer.Deserialize<List<Medicine>>(json) ??
  new List<Medicine>();
    if (_medicines.Any())
    {
        _nextId = _medicines.Max(m => m.Id) + 1;
    }
  }
}
//Збереження ліків у JSON
private void SaveMedicines()
{
```

Рисунок 3.50 – Метод LoadMedicines()

Метод SaveMedicines() використовується для збереження ліків у файлі JSON (рис. 3.51).

```
private void SaveMedicines()
{
    string json = JsonSerializer.Serialize(_medicines,
    new JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText(MedicinesFile, json);
}
```

Рисунок 3.51 – SaveMedicines()

Метод DeleteMedicine() – це метод для видалення ліків за ID (рис. 3.52).

```
public void DeleteMedicine(int id)
{
    var medicine = _medicines.FirstOrDefault(m => m.Id == id);
    if (medicine != null)
    {
        _medicines.Remove(medicine);
        SaveMedicines();
    }
    else
    {
        // Логіка для обробки ситуації, коли ліки з таким ID не знайдено
        Console.WriteLine($"Medicine with ID {id} not found.");
    }
}
```

Рисунок 3.52 – Метод DeleteMedicine()

Сервіс PharmacyService.cs забезпечує збереження та обробку даних про клієнтів і ліки у файлах JSON без використання бази даних. Він дозволяє додавати, отримувати, перевіряти наявність, авторизовувати клієнтів, а також працювати зі списком ліків, включаючи додавання та видалення. Це ефективне рішення для збереження даних у вигляді файлів.(Повний код PharmacyService.cs в додатку В).

3.7 Налаштування wwwroot (стилів та зображенн) проєкту

У проєктах ASP.NET Core статичні файли (стилі, зображення тощо) зазвичай зберігаються в папці wwwroot. Це спеціальна папка, яку система використовує для надання доступу до статичних ресурсів з веб-сайту [15].

Для нашого сайту у папці wwwroot потрібно створити папки css, images, sounds і data. В папці images будуть зберігатися, зображення, які ми обираємо при додаванні ліків і фон.

В папці css ми створимо файл site.css; в ньому будуть зберігатися наші стилі сайту (код з файлу site.css в додатку Г).

У файлі site.css ми прописали стилі які оформлюють інтерфейс сайту. Основний фон сайту має світло-зелений колір і містить фонове зображення, яке розтягується на всю сторінку, фіксується під час прокрутки та не повторюється. Заголовки стилізовані в темно-зеленому кольорі та вирівнюються по центру.

Контейнер сайту має відступ знизу для врахування футера та займає всю доступну висоту екрана. Кнопки мають зелений фон, білі написи, закруглені краї та змінюють колір і додають тінь при наведенні. Посилання також виділяються зеленим кольором і стають темнішими при наведенні.

Форми мають білий фон, округлі межі та тонку сіру рамку. Вони містять відступи, заголовки, підписи до полів, а також текстові поля, що займають всю ширину форми. Кнопки у формах стилізовані під основні кнопки сайту.

Список ліків відображається у вигляді гнучких карток, що вирівнюються по центру сторінки. Кожен елемент має білий фон, закруглені краї, тінь і змінює

положення та тінь при наведенні. Зображення ліків заповнює верхню частину картки, адаптуючись до її розміру. Опис ліків містить назву, текст опису та ціну, яка виділена синім кольором.

Для таблиць застосовується стиль із світлою сіро-зеленою шапкою, межами між комірками та чергуванням кольорів рядків для зручності читання. При наведенні рядки змінюють фон на світліший.

Шапка сайту оформлена у вигляді темно-зеленого блока з білим текстом, містить логотип і кнопку відкриття меню. Меню розташоване у верхньому правому куті, з'являється при натисканні кнопки та містить посилання білого кольору, що підкреслюються при наведенні.

Стилі також передбачають адаптивність: на маленьких екранах картки ліків розтягуються на всю ширину, а всі елементи залишаються зручними для взаємодії.

У папку sounds загрузиться фонова музика з розширенням wav (рис. 3.53).

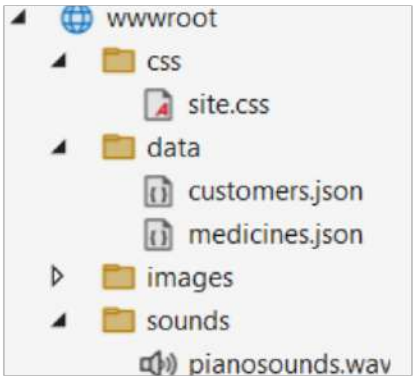

Рисунок 3.53 – Структура wwwroot

Упапці data створиться два файли, customers.json і medicines.json. В них будуть зберігатися зареєстровані клієнти та додані ліки.

3.8 Налаштування Program.cs проєкту

Program.cs є головною точкою входу у наш сайт і відповідає за ініціалізацію всієї конфігурації, сервісів та компонентів сайту. У ньому

відбувається налаштування сервісів, маршрутизації, обробка HTTP-запитів, а також інтеграція з іншими частинами програми.

Код який визначає клас Program показаний на рисунку 3.54. Метод Main – точка входу в додаток. WebApplication.CreateBuilder(args) створює об’єкт builder, який використовується для налаштування веб-додатка (рис. 3.54).

```
public class Program
{
  public static void Main(string[] args)
  {
    var builder = WebApplication.CreateBuilder(args);
```

Рисунок 3.54 – Визначення класу Program та створення об’єкту builder

Додавання логування в консоль показана на рисунку 3.55.

```
builder.Logging.AddConsole();
```

Рисунок 3.55 – Додавання логування

Додаємо служби для підтримки локалізації (builder.Services.AddLocalization()) та MVC (builder.Services.AddControllersWithViews()) (рис. 3.56).

```
builder.Services.AddLocalization(options => options.ResourcesPath = "Resources");
builder.Services.AddControllersWithViews()
                .AddDataAnnotationsLocalization()
                .AddViewLocalization();
```

Рисунок 3.56 – Служби для підтримки локалізації та MVC

Додаємо підтримку сесій (рис. 3.57). Де AddDistributedMemoryCache() – додає внутрішнє кешування для збереження сесій, а AddSession() – налаштовує сесії.

```
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});
```

Рисунок 3.57 – Підтримка сесій

Додаємо службу PharmacyService та створюємо веб-додаток (рис. 3.58).

Служба builder.Services.AddSingleton<PharmacyService>() означає що об'єкт PharmacyService буде створений один раз і використовуватиметься у всьому додатку.

app = builder.Build() означає що після конфігурації сервісів створюється веб-додаток.

```
builder.Services.AddSingleton<PharmacyService>();

var app = builder.Build();
```

Рисунок 3.58 – Додавання служби PharmacyService та створення веб-додатку

Метод app.UseSession() – активує використання сесій у додатку (рис. 3.59).

```
app.UseSession();
```

Рисунок 3.59 – Використання сесії

Додаємо використання статичних файлів (рис. 3.60).

Метод app.UseStaticFiles(new StaticFileOptions {...}); – налаштовує сервер для обслуговування статичних файлів (зображень, CSS, JS).

```
app.UseStaticFiles(new StaticFileOptions
{
    ServeUnknownFileTypes = true,
    DefaultContentType = "image/png"
});
```

Рисунок 3.60 – Використання статичних файлів

Метод app.UseRouting() додає підтримку маршрутизації (рис. 3.61).

```
app.UseRouting();
```

Рисунок 3.61 – Додавання підтримки маршрутизації

Додаємо маршрути (рис. 3.62).

Метод app.MapControllerRoute(...) – визначає, як обробляти URL-запити.

Якщо URL починається з admin/, запит буде оброблений AdminController, а за замовчуванням викликатиметься Dashboard, a"default" – визначає стандартний маршрут: {controller=Home}/{action=Index}/{id?}.

```
app.MapControllerRoute(
name: "admin",
pattern: "admin/{controller=Admin}/{action=Dashboard}/{id?}");

app.MapControllerRoute(
name: "default",
pattern: "{controller=Home}/{action=Index}/{id?}");
```

Рисунок 3.62 – Додавання маршрутів

Метод app.Run() для запуску додатку показано на рисунку 3.63.

```
app.Run();
```

Рисунок 3.63 – Запуск додатку

Цей код налаштовує ASP.NET Core MVC-додаток для програмно–апаратної системи інформування клієнтів аптеки, забезпечуючи всі необхідні елементи для його роботи. У ньому реалізовано логування в консоль, що дозволяє відстежувати події в додатку. Додається підтримка локалізації, що забезпечує можливість зміни мов інтерфейсу. Використання сесій дозволяє зберігати тимчасові дані про користувачів, наприклад, інформацію про їхню авторизацію.

До служб додається PharmacyService, який відповідає за управління даними про ліки та клієнтів. Використання статичних файлів дозволяє працювати з CSS, JavaScript, зображеннями та звуком, що є важливим для візуального оформлення сайту. Завдяки налаштуванню маршрутизації забезпечується правильна обробка запитів до сторінок, зокрема, передбачено окремий шлях для панелі адміністратора. В кінці виконується запуск додатка, що забезпечує його готовність до роботи та обробку HTTP-запитів.

Крім того, локалізація реалізована через AddLocalization та AddViewLocalization, що дає можливість легко адаптувати інтерфейс для користувачів з різних мовних груп [16].

Обробка статичних файлів у конфігурації UseStaticFiles дозволяє налаштовувати доступ до зображень, таблиць стилів і скриптів, що покращує продуктивність і зручність роботи з інтерфейсом. Використання ServeUnknownFileTypes = true розширює можливості сервера у випадках, коли потрібно обслуговувати нестандартні типи файлів [17].

Налаштування маршрутизації дозволяє гнучко визначати URL-адреси для сторінок. Це спрощує навігацію та робить структуру сайту логічною для користувачів.

Таким чином, даний код є основою для стабільної та функціональної веб-системи, що забезпечує швидкий доступ клієнтів до інформації про ліки та ефективне управління аптечним асортиментом з боку адміністратора (повний код Program.cs знаходиться в додатку Д).

3.9 Вигляд структури готового проєкту

Після налаштування всіх компонентів, таких як моделі, представлення, контролери, статичні файли, сервіси та Program.cs, програмно-апаратна система інформування клієнтів аптеки засобами C# ASP .NET CORE MVC вважається створеною, а її структура проєкту виглядає, як на рисунку 3.64.

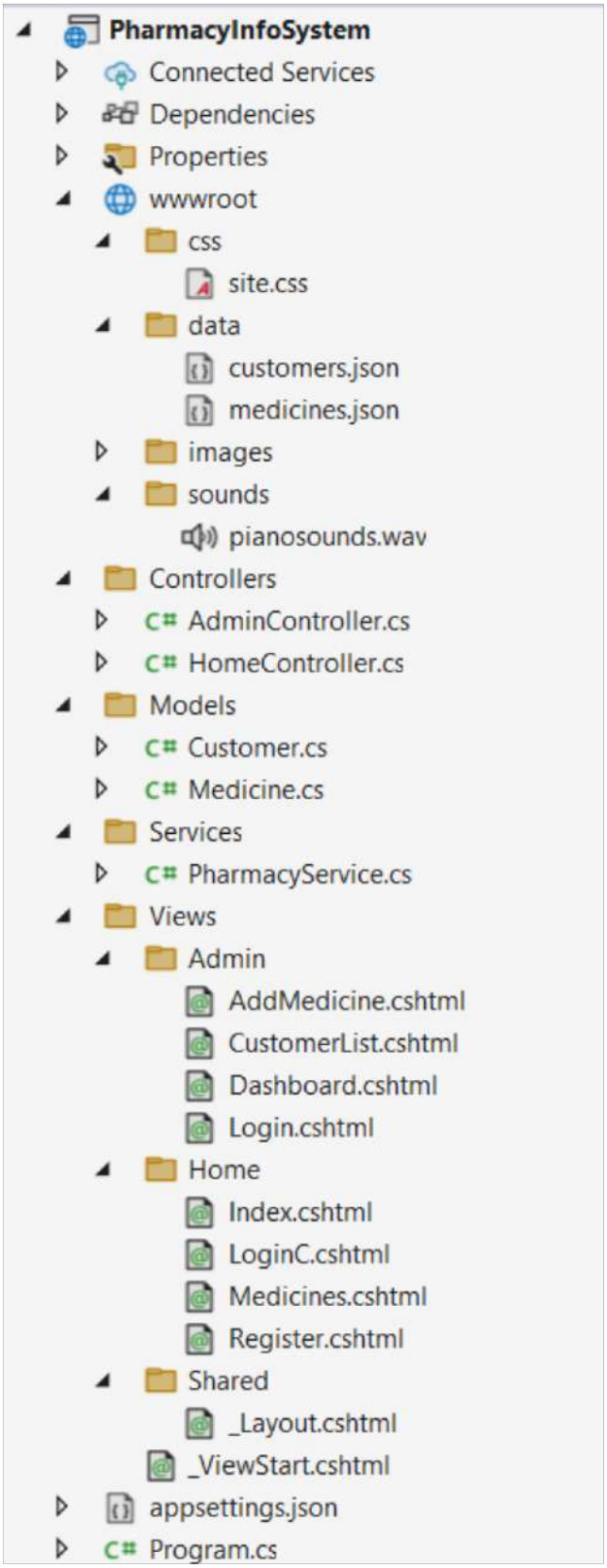

Рисунок 3.64 – Заповнена структура проєкту

Тепер можна спробувати запустити програму, для цього потрібно натиснути Start Wthout Debugging (рис. 3.65) або Ctrl+F5 у Visual Studio 2022.

Рисунок 3.65 – Start Wthout Debugging

Запуск програми пройшов успішно, відкрився сайт з назвою localhost – це ім'я хосту, що вказує на поточний комп'ютер, де працює додаток. Це означає, що він працює на локальному комп'ютері, а не в інтернеті.

При запуску програми викликається представлення Index.cshtml (рис. 3.66); воно є головною сторінкою аптеки, на якій є надпис «Аптека», також через _Layout.cshtml викликається шапка та футер (рис. 3.67) сайту.

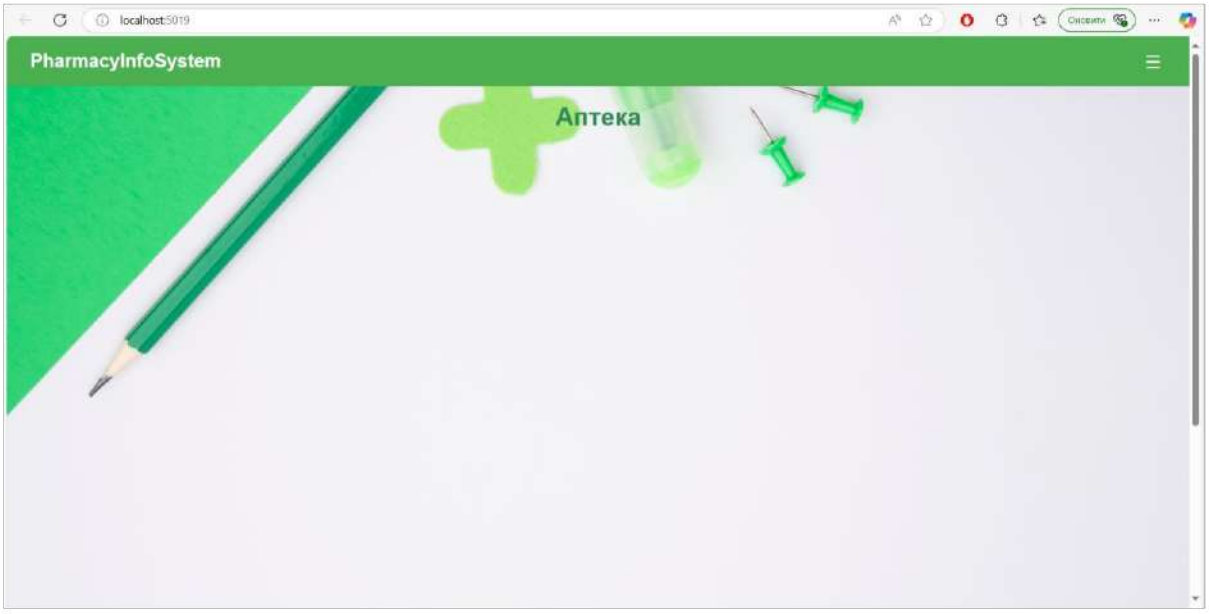

Рисунок 3.66 – Головна сторінка з шапкою

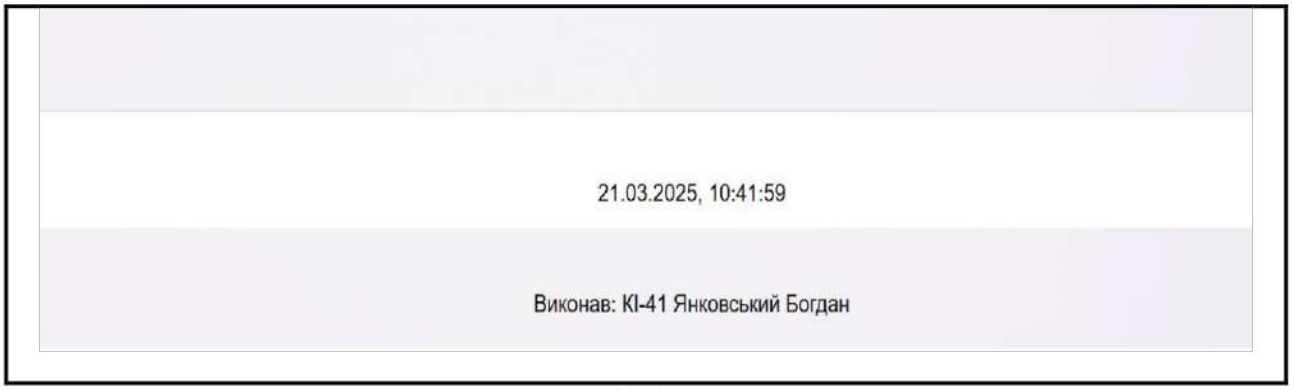

Рисунок 3.67 – Футер

У футері відображається надпис та точний час за Києвом, який динамічно змінюється. У шапці знаходиться меню (рис. 3.68), при натисканні на яке вискакує вікно для переходу між сторінками.

Рисунок 3.68 – Вікно для переходу між сторінками

Натиснемо «Переглянути список ліків»; після натискання нас переносить на список ліків (рис. 3.69), який прописаний в представленні Medicines.cshtml. Як бачимо, на сторінці є можливість пошуку та сортування (за назвою, за ціною та по ID (за замовчуванням)), але ліки не можна переглянути, поки користувач не буде авторизований через представлення LoginC.cshtml (рис. 3.70), або, якщо користувача немає в системі, зареєстрований через Register.cshtml (рис. 3.71).

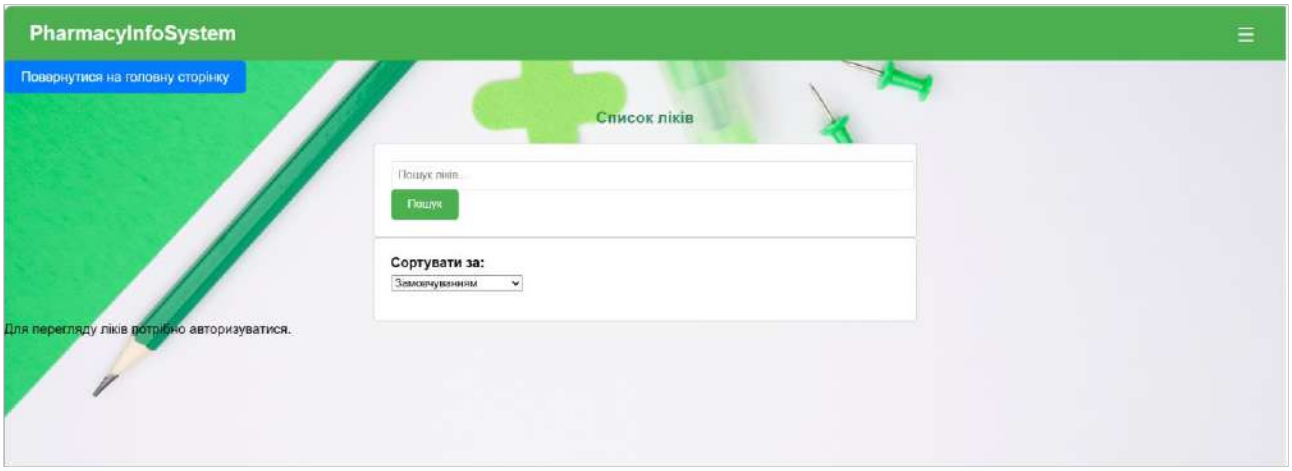

Рисунок 3.69 – Список ліків (без авторизації)

Щоб з’явився список ліків, нам потрібно авторизуватися; в меню натиснемо вкладку «Авторизація». Вводимо дані, яких немає в системі, вилізе помилка. Спробуємо зареєструватися, для цього в меню натиснемо «Реєстрація».

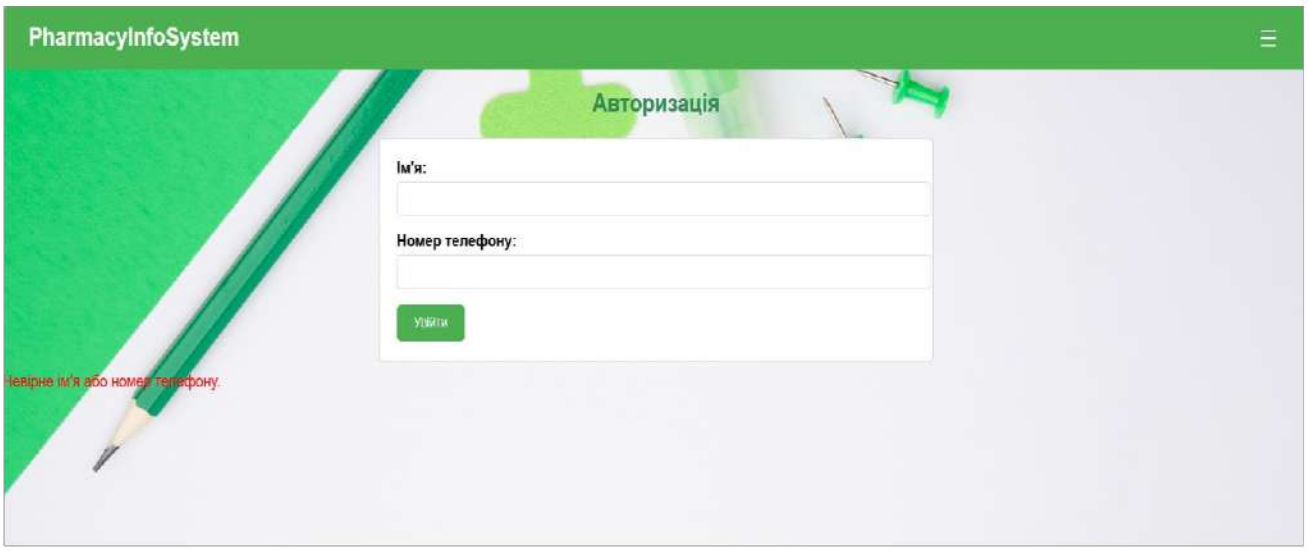

Рисунок 3.70 – Панель авторизації

В поле для реєстрації вводимо ім’я та намер телефону, натискаємо «Зареєструватися».

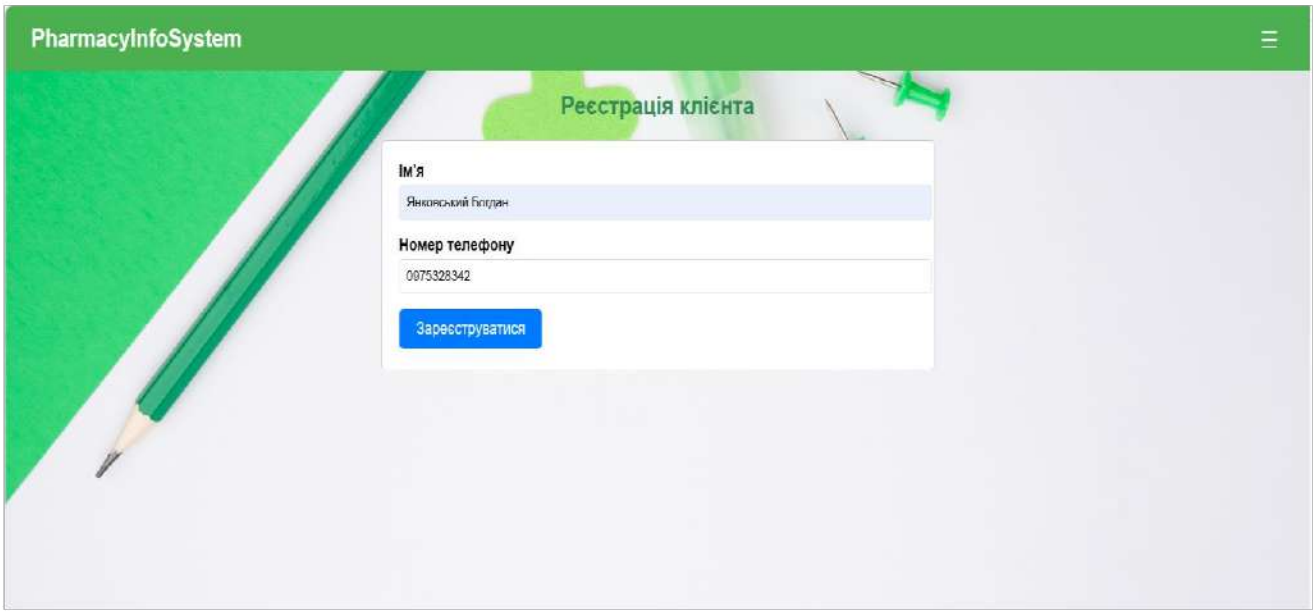

Рисунок 3.71 – Панель реєстрації

Після натискання «Зареєструватися», нас перенесе до списку ліків (рис. 3.72). Тепер користувач зможе переглядати ліки в списку, використовувати функцію пошуку (рис. 3.73) та сортування (рис. 3.74).

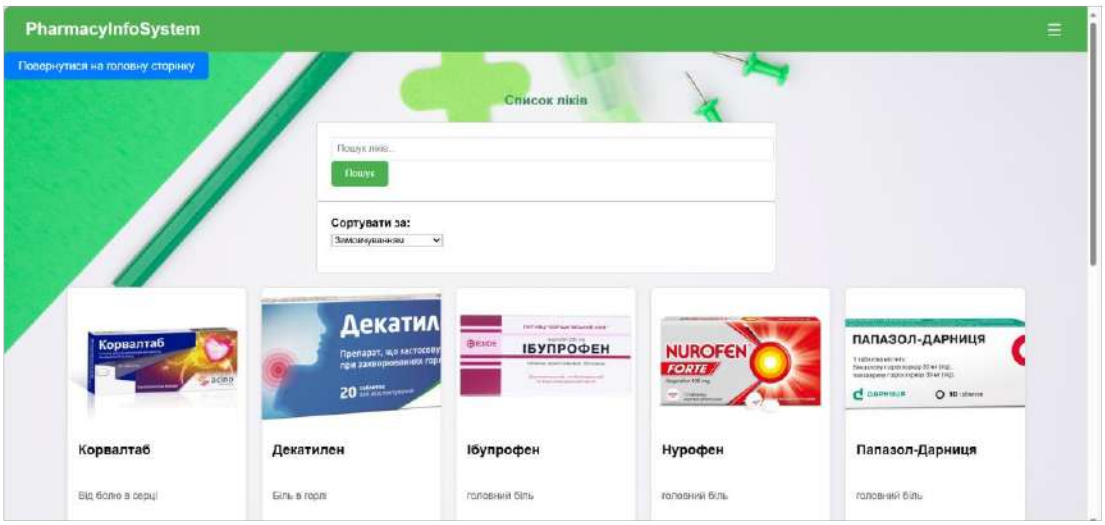

Рисунок 3.72 – Список ліків (після авторизації)

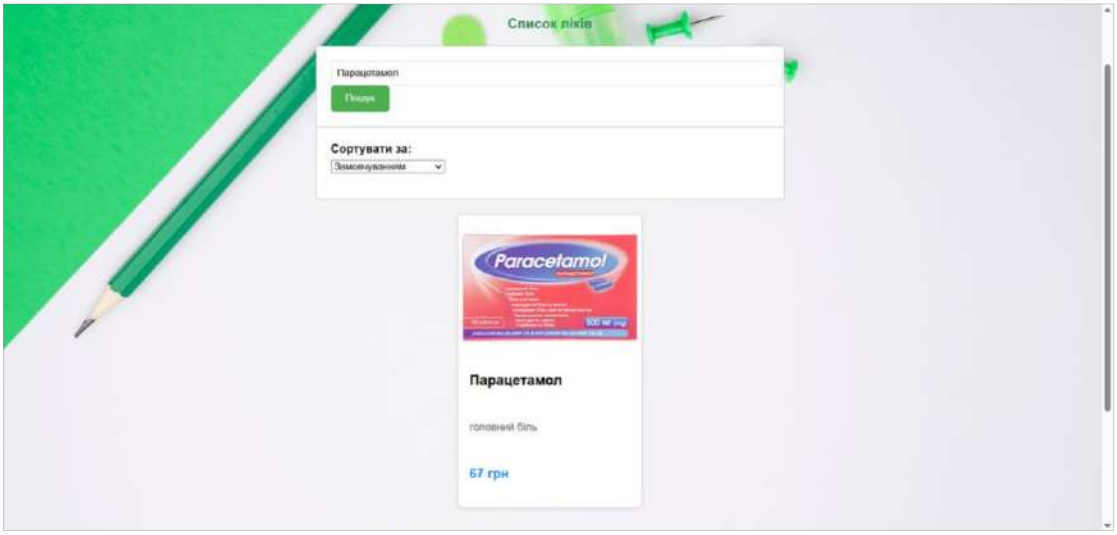

Рисунок 3.73 – Використання функції пошуку

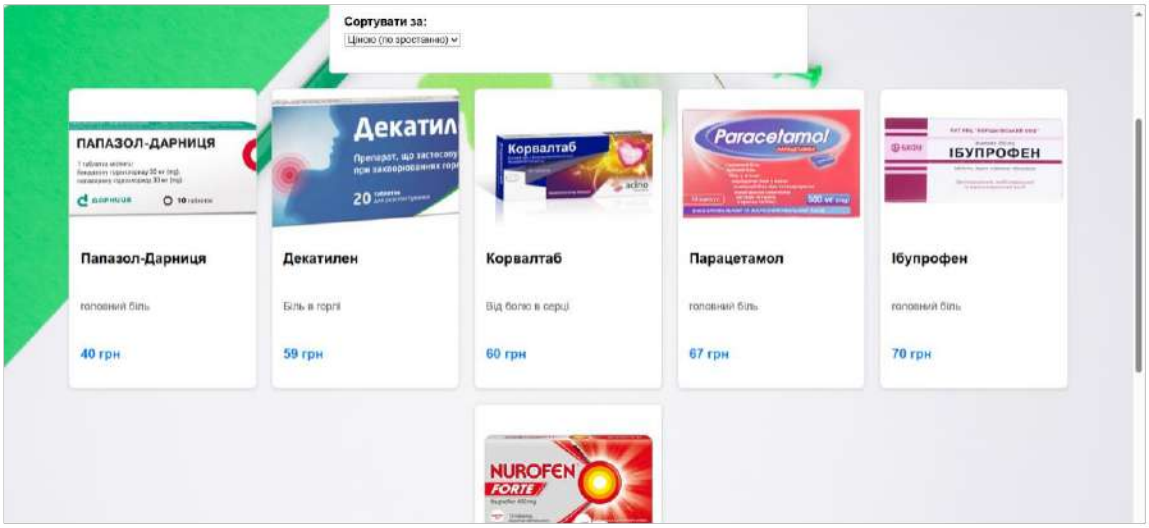

Рисунок 3.74 – Сортування (за ціною по зростанню)

Тепер в меню натиснемо «Звук», після чого спливе повідомлення, в якому буде запит на відтворення фонового звуку на сайті (рис. 3.75).

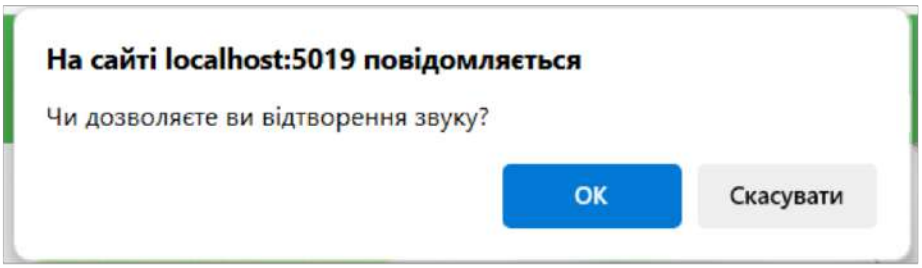

Рисунок 3.75 – Запит на відтворення фонового звуку

Після підтвердження буде програватися фоновий звук.

Тепер в меню натиснемо «Куточок адміністратора», після чого нас переносить на форму для підтвердження (рис. 3.76), чи ви дійсно адміністратор; вона прописана в представленні Login.cshtml. Там потрібно ввести пароль (admin) та логін (admin); якщо введено неправильно то вискочить помилка.

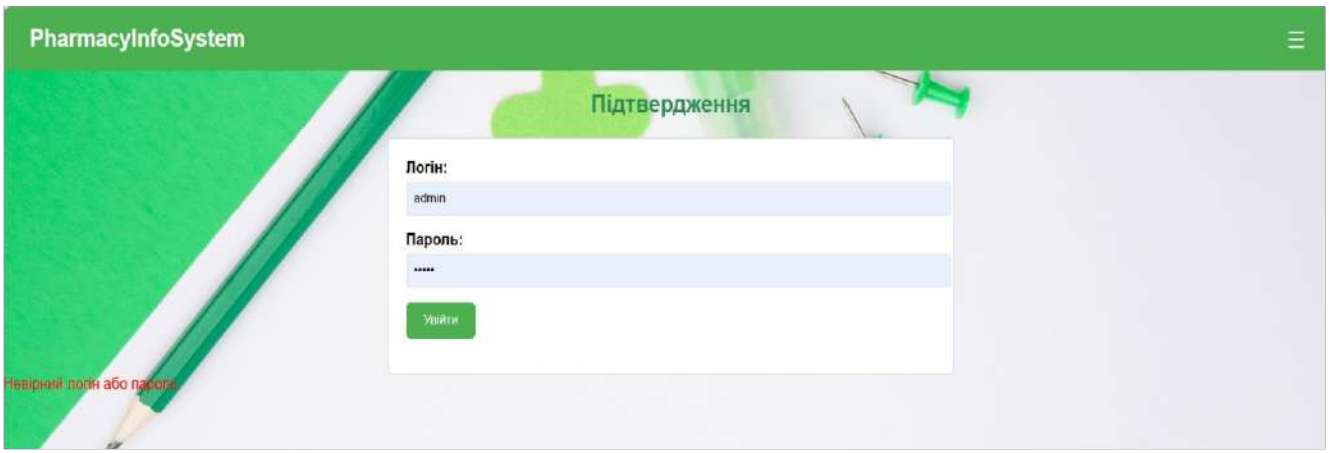

Рисунок 3.76 – Форма для авторизації адміністратора

Якщо все введено правильно, то після натискання «Увійти» нас має перенести на панель адміністратора (рис. 3.77), яка прописана в представленні Dashboard.cshtml. На панелі є кнопка для переходу до списку клієнтів та додавання ліків; також на панелі є список ліків з можливістю їх видалення та пошук ліків.

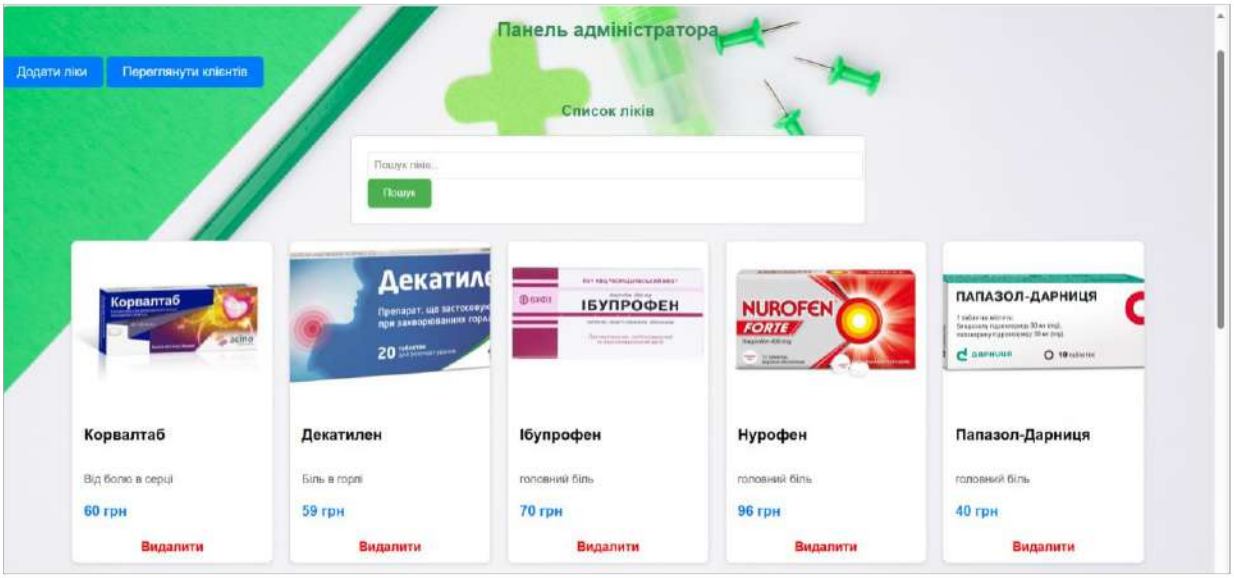

Рисунок 3.77 – Панель адміністратора

Спробуємо натиснути кнопку «Додати ліки», після чого нас має перенести на панель додавання ліків (рис. 3.78), яка прописана в представленні AddMedicine.cshtml.

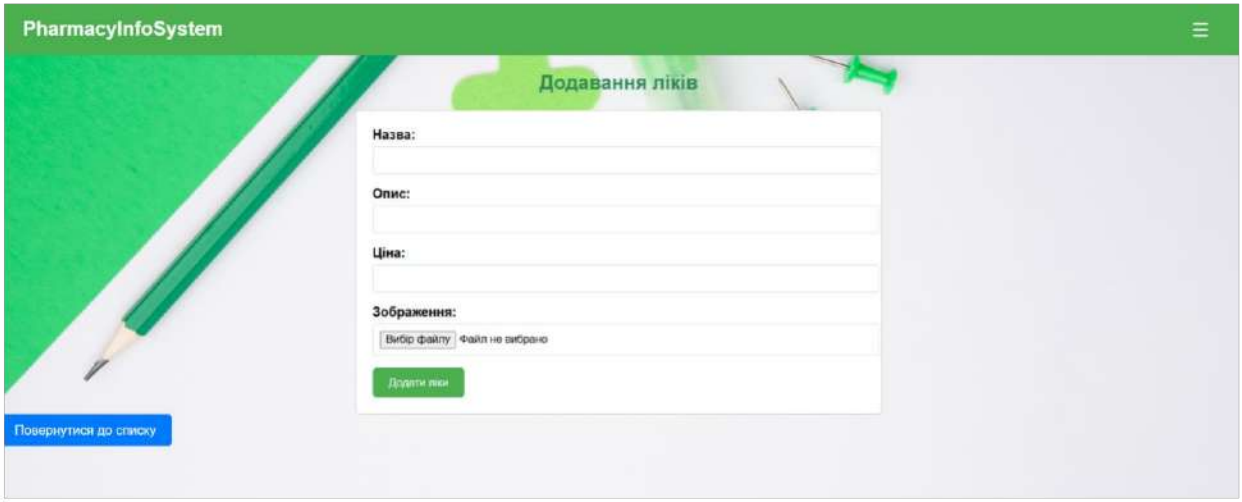

Рисунок 3.78 – Панель додавання ліків

На панелі додавання ліків потрібно обов'язково ввести назву, опис, ціну та додати зображення; якщо якесь з полів не заповнити, спливе повідомлення, що потрібно заповнити це поле або додати файл (рис. 3.79).

Для того щоб додати зображення до ліків, на панелі для додавання ліків потрібно натиснути на «Вибір файлу», після чого відкриється вікно, де можна

обрати зображення з комп’ютера (рис. 3.80).

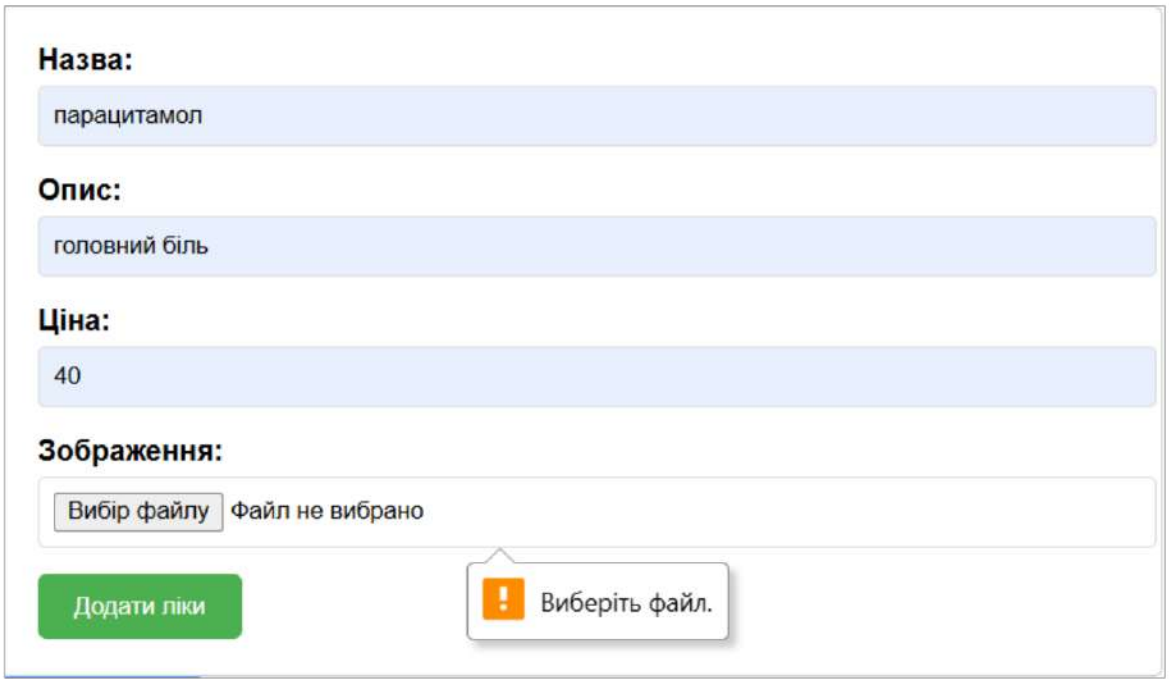

Рисунок 3.79 – Панель додавання ліків з спливаючим повідомленням

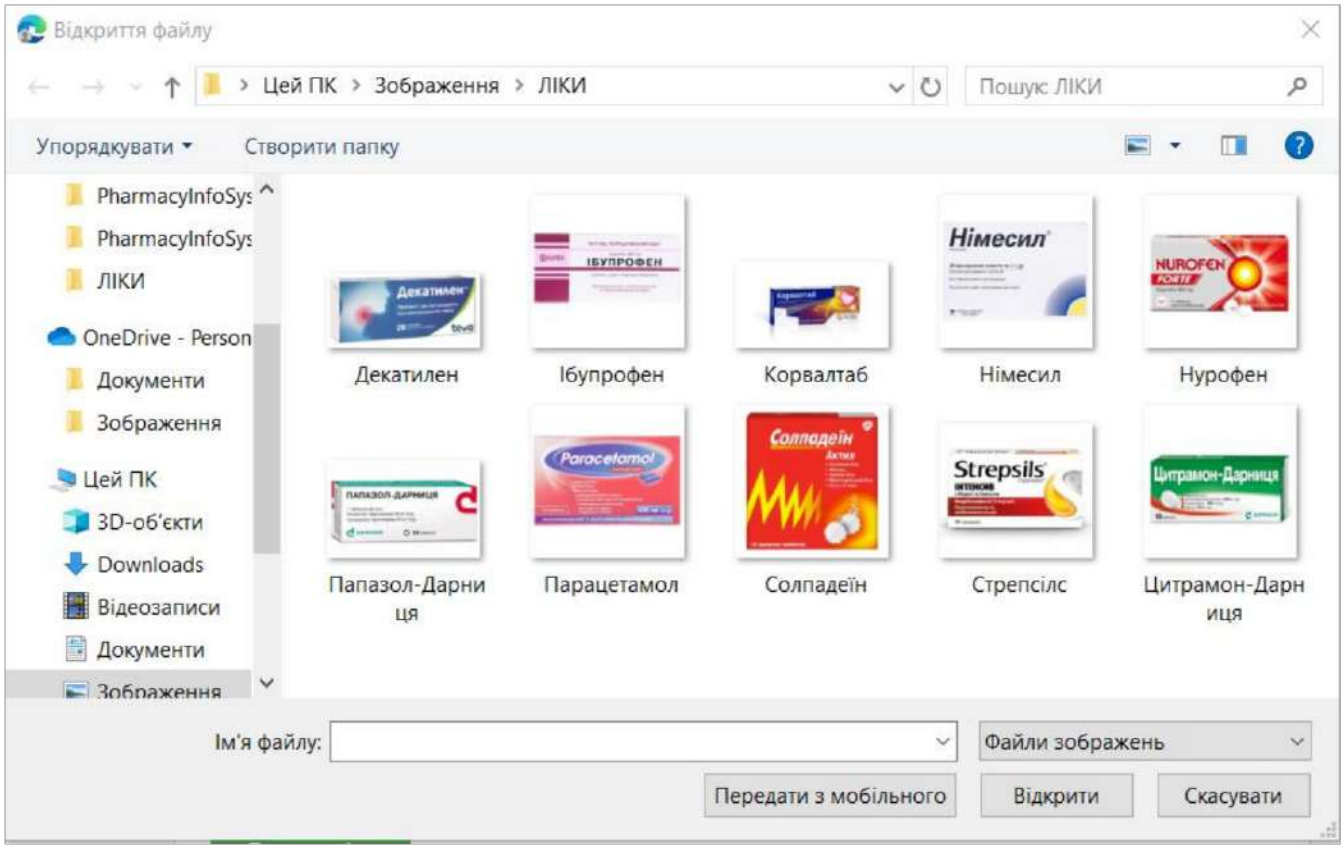

Рисунок 3.80 – Вікно де можна обрати зображення з комп’ютера

Обираємо зображення натискаємо «відкрити»; тепер на панелі видно, що забраження було додано (рис. 3.81).

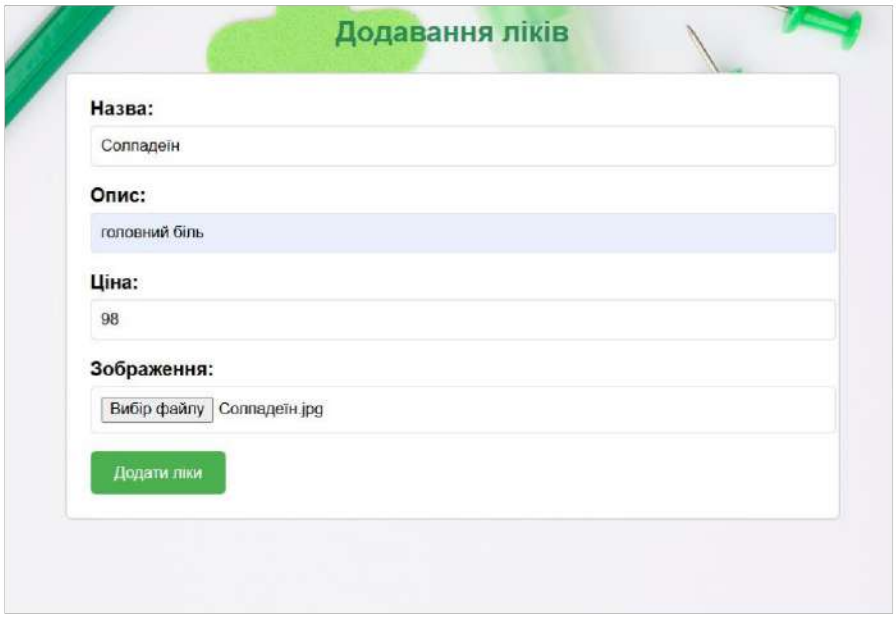

Рисунок 3.81 – Панель додавання ліків із заповненими полями та доданим зображенням

Після того, як заповнили всі поля і додали зображення, натискаємо кнопку «Додати ліки», після чого нас автоматично переносить на панель адміністратора, де в списку ліків зявилися додані ліки (рис. 3.82).

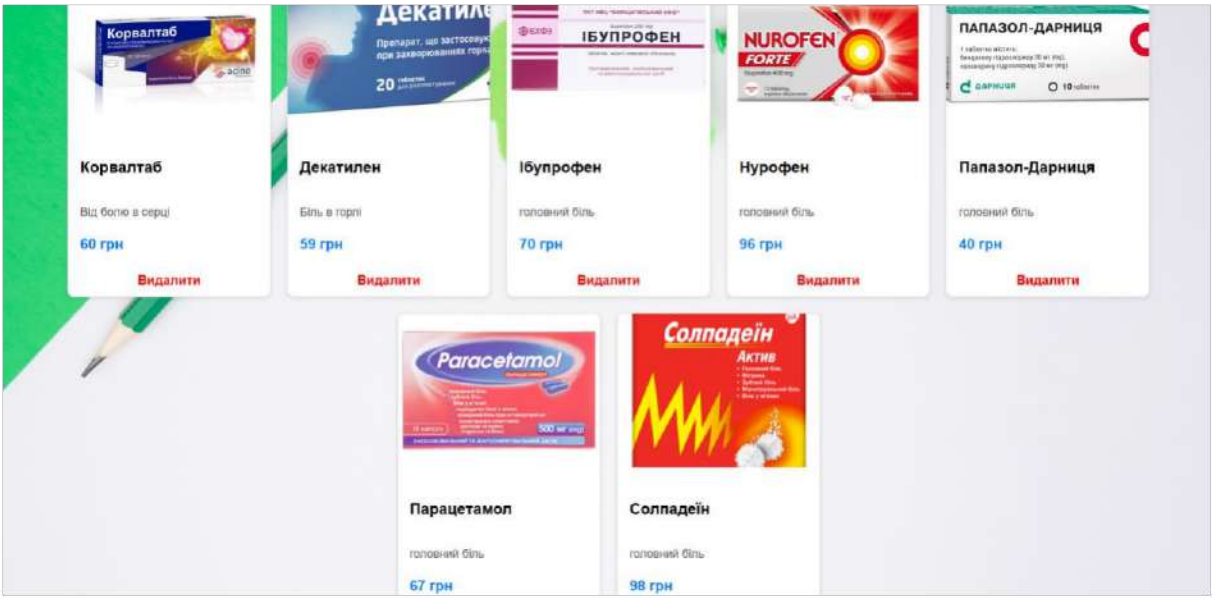

Рисунок 3.82 – Панель адміністратора з доданими ліками (Солпадеїн)

Тепер спробуємо видалити якийсь із препаратів. Натискаємо видалити на «Корвалтаб»; спливає вікно для підтвердження видалення (рис. 3.83); натискаємо «ОК», і спливло повідомлення, що ліки видалено (рис. 3.84); натискаємо «ОК», і тепер видно, що «Корвалтаб» зник зі списку (рис. 3.85).

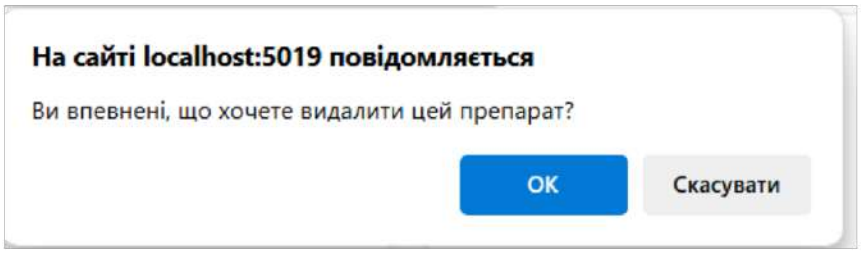

Рисунок 3.83 – Підтвердження видалення

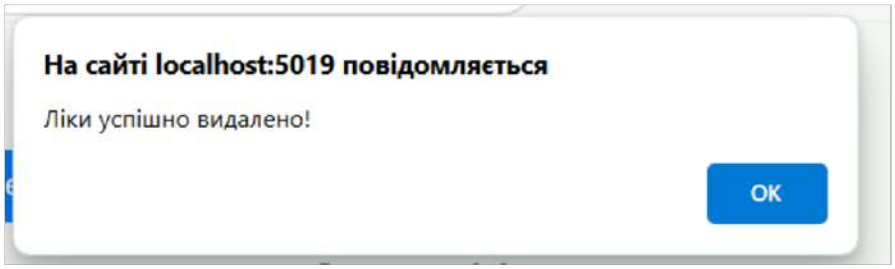

Рисунок 3.84 – Повідомлення що ліки видалено

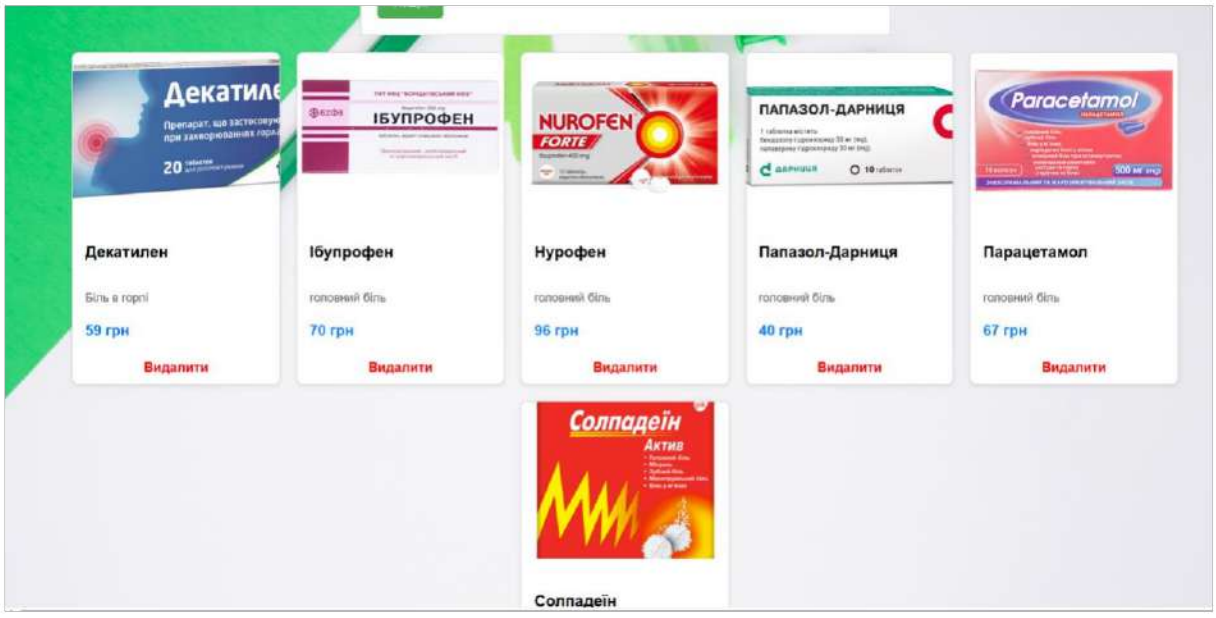

Рисунок 3.85 – Список ліків з видаленим препаратом (Корвалтаб)

Тепер, коли ми переконалися, що ліки видаляються із списку, ми можемо перевірити список клієнтів (рис. 3.86); він прописаний в CustomerList.cshtml. Для того, щоб переглянути список клієнтів, нам потрібно на панелі адміністратора натиснути вкладку «Переглянути клієнтів».

Рисунок 3.86 – Список клієнтів

Фонове зображення було завантажене з сайту Freepik [18], звук зевантажений із сайту Freesound [19], а забраження для препаратів з сайту TABLETKI.UA [20].

ВИСНОВКИ

Всі завдання, поставлені у кваліфікаційній роботі бакалавра, виконані у повному обсязі.

Зокрема, у кваліфікаційній роботі:

1) проведено огляд програмних засобів розробки програмно-апаратної системи інформування клієнтів аптеки засобами C# ASP.NET Core MVC;

2) створено проєкт, що складається з двох частин: адміністративної та користувацької;

3) додано стилі та фонові зображення для покращення візуального оформлення;

4) реалізовано аудіосупровід у вигляді фонової мелодії;

5) створено шапку з логотипом та спливаючим меню для зручності навігації;

6) додано футер з динамічними елементами;

7) для користувачів реалізовані: можливість реєстрації та авторизації; перегляд списку ліків із можливістю пошуку та сортування;

8) для адміністратора реалізовані: можливість авторизації; перегляд списку ліків; додавання нових ліків; видалення ліків; перегляд списку клієнтів;

9) використані JSON-файли як альтернативу базі даних для зберігання інформації про ліки та клієнтів.

Програмний проєкт був відлагоджений та протестований, що підтвердило його коректну роботу. Розроблена програмно-апаратна система може бути впроваджена в реальних аптеках для оптимізації роботи та підвищення рівня обслуговування клієнтів. Крім того, вона може бути використана у навчальному процесі для вивчення технологій веб-розробки та інформаційних систем у фармацевтичній сфері.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/cs/cs_intro.php (дата звернення: 15.03.2025).

2. Учасники проєктів Вікімедіа. C Sharp Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 26.03.2025).

3. What is JavaScript? - Learn web development | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/ Learn_web_development/Core /Scripting/ What_is_JavaScript (дата звернення: 15.03.2025).

4. Учасники проєктів Вікімедіа. JavaScript Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 15.03.2025).

5. Учасники проєктів Вікімедіа. HTML Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення: 17.03.2025).

6. HTML | MDN. MDN Web Docs. URL: https://developer.mozilla.org/ https://developer.mozilla.org/en-US /docs/Web/HTML (дата звернення: 17.03.2025).

7. Учасники проєктів Вікімедіа. CSS Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення: 17.03.2025).

8. ITSTEP Академія Київ. URL: https://kiev.itstep.org/blog/what-is-css-and-why-does-a-web-developer-need-it (дата звернення: 17.03.2025).

9. Учасники проєктів Вікімедіа. ASP.NET MVC Framework – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/ASP.NET_MVC_Framework (дата звернення: 20.03.2025).

10. Учасники проєктів Вікімедіа. Модель-вид-контролер Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki /Модель-вид-контролер (дата звернення: 23.03.2025).

11. Configuration in ASP.NET Core. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core /fundamentals/configuration/?view=aspnetcore-9.0 (дата звернення: 23.03.2025).

12. Controllers in ASP.NET Core MVC. Dot Net Tutorials. URL: https://dotnettutorials.net /lesson /controllers-asp-net-core-mvc/ (дата звернення: 23.03.2025).

13. Part 4, add a model to an ASP.NET Core MVC app. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-9.0&tabs=visual-studio (дата звернення: 23.03.2025).

14. Views in ASP.NET Core MVC. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core /mvc/views/overview?view=aspnetcore-9.0 (дата звернення: 24.03.2025).

15. ASP.NET Core MVC wwwroot Folder. TutorialsTeacher - Learn, Practice and Master Technologies. URL: https://www.tutorialsteacher.com /core/aspnet-core-wwwroot (дата звернення: 24.03.2025).

16. Provide localized resources for languages and cultures in an ASP.NET Core app. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/localization/provide-resources?view=aspnetcore-8.0 (дата звернення: 24.03.2025).

17. Static files in ASP.NET Core. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core /fundamentals/static-files?view=aspnetcore-9.0 (дата звернення: 24.03.2025).

18. Freepik | Create great designs, faster. Freepik. URL: https://www.freepik.com/ (дата звернення: 26.03.2025).

19. Freesound. Freesound. URL: https://freesound.org/ (дата звернення: 26.03.2025).

20. TABLETKI.UA. Сервіс пошуку та порівняння цін на лікарські засоби. URL: https://tabletki.ua/ (дата звернення: 26.03.2025).

ДОДАТКИ

Додаток А

Повний код HomeController

```
using Microsoft.AspNetCore.Mvc;
using PharmacyInfoSystem.Models;
using PharmacyInfoSystem.Services;
namespace PharmacyInfoSystem.Controllers
{
    public class HomeController : Controller
    {
        private readonly PharmacyService _pharmacyService;

        public HomeController(PharmacyService pharmacyService)
        {
            _pharmacyService = pharmacyService;
        }

        public IActionResult Index()
        {
            var medicines = _pharmacyService.Medicines;
            return View(medicines);
        }


        // Дія для реєстрації клієнта
        [HttpGet]
        public IActionResult Register()
        {
            return View();
        }

        // Дія для обробки реєстрації
        [HttpPost]
        public IActionResult Register(Customer customer)
        {
            if (ModelState.IsValid)
            {
                _pharmacyService.AddCustomer(customer);
                HttpContext.Session.SetString("Phone", customer.Phone);
                return RedirectToAction("Medicines");
            }
```

```
            return View(customer);
        }
        // Дія для авторизації
        [HttpGet]
        public IActionResult LoginC()
        {
            return View();
        }

        // Дія для обробки авторизації
        [HttpPost]
        public IActionResult LoginC(string name, string phoneNumber)
        {

            var customer = _pharmacyService.Customers
            .FirstOrDefault(c => c.Name == name && c.Phone ==
phoneNumber);

            if (customer != null)
            {

                HttpContext.Session.SetString("Phone", phoneNumber);
                return RedirectToAction("Medicines");
            }
            else
            {

                ViewBag.ErrorMessage = "Невірне ім'я або номер
телефону.";
                return View();
            }
        }
        // Дія для відображення ліків
        public IActionResult Medicines(string searchTerm, string
sortOrder)
        {

            var phone = HttpContext.Session.GetString("Phone");
            if (string.IsNullOrEmpty(phone) ||
!_pharmacyService.IsCustomerRegistered(phone))
            {
```

```
                ViewBag.ErrorMessage = "Для перегляду ліків потрібно
авторизуватися.";
                return View();
            }

            var medicines = _pharmacyService.Medicines;

            //Пошук
            if (!string.IsNullOrEmpty(searchTerm))
            {
                medicines = medicines.Where(m =>
m.Name.Contains(searchTerm, StringComparison.OrdinalIgnoreCase) ||
                m.Description.Contains(searchTerm,
StringComparison.OrdinalIgnoreCase)).ToList();
            }

            // Перевіряємо, чи є ліки після пошуку
            if (medicines == null || !medicines.Any())
            {
                ViewBag.ErrorMessage = "Ліки не знайдені.";
            }
            // Сортування
            switch (sortOrder)
            {
                case "name_asc":
                    medicines = medicines.OrderBy(m => m.Name).ToList();
                    break;
                case "name_desc":
                    medicines = medicines.OrderByDescending(m =>
m.Name).ToList();
                    break;
                case "price_asc":
                    medicines = medicines.OrderBy(m => m.Price).ToList();
                    break;
                case "price_desc":
                    medicines = medicines.OrderByDescending(m =>
m.Price).ToList();
                    break;
                default:
                    break;
            }
            ViewData["SortOrder"] = sortOrder;
            ViewData["SearchTerm"] = searchTerm;
```

```
            return View(medicines);
        }
    }
}
```

Додаток Б

Повний код AdminController

```
using Microsoft.AspNetCore.Mvc;
using PharmacyInfoSystem.Models;
using PharmacyInfoSystem.Services;
using System;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;

namespace PharmacyInfoSystem.Controllers
{
    public class AdminController : Controller
    {
        private readonly PharmacyService _pharmacyService;

        public AdminController(PharmacyService pharmacyService)
        {
            _pharmacyService = pharmacyService ?? throw new
ArgumentNullException(nameof(pharmacyService));
        }

        [HttpGet]
        public IActionResult Dashboard(string searchTerm)
        {

            if (HttpContext.Session.GetString("IsAdmin") != "true")
            {
                return RedirectToAction("Login");
            }
            var medicines = _pharmacyService.GetMedicines();
            if (!string.IsNullOrEmpty(searchTerm))
            {
                medicines = medicines.Where(m =>
m.Name.Contains(searchTerm, StringComparison.OrdinalIgnoreCase) ||
                m.Description.Contains(searchTerm,
StringComparison.OrdinalIgnoreCase)).ToList();
            }
            foreach (var medicine in medicines)
```

```
                {
                    Console.WriteLine($"Medicine in dashboard:
{medicine.Name}, {medicine.Description}");
                }
                ViewData["SearchTerm"] = searchTerm;
                return View(medicines);
            }

            public IActionResult CustomerList()
            {
                var customers = _pharmacyService.Customers;
                return View(customers);
            }

            [HttpGet]
            public IActionResult AddMedicine()
            {
                if (HttpContext.Session.GetString("IsAdmin") != "true")
                {
                    return RedirectToAction("Login");
                }
                return View();
            }
            [HttpPost]
            public async Task<IActionResult> AddMedicine(Medicine medicine,
IFormFile image)
            {
                Console.WriteLine("Метод AddMedicine викликано");

                if (HttpContext.Session.GetString("IsAdmin") != "true")
                {
                    return RedirectToAction("Login");
                }

                Console.WriteLine($"Received medicine: {medicine.Name},
{medicine.Description}, {medicine.Price}");

                if (image == null || image.Length == 0)
                {
                    Console.WriteLine("Зображення не вибрано або файл
пустий.");
                    ModelState.AddModelError("ImageUrl", "Зображення є
обов'язковим.");
```

```
                    }
                    else if (image.ContentType.StartsWith("image"))
                    {
                        Console.WriteLine($"Зображення отримано:
{image.FileName}, Тип: {image.ContentType}");

                        string fileName = Guid.NewGuid().ToString() +
Path.GetExtension(image.FileName);
                        var filePath =
Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images", fileName);

                        Console.WriteLine($"Унікальне ім'я для зображення
згенеровано: {fileName}");

                        var imageDirectory =
Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images");
                        if (!Directory.Exists(imageDirectory))
                        {
                            Console.WriteLine($"Папка {imageDirectory} не існує,
створюємо її...");
                            Directory.CreateDirectory(imageDirectory);
                        }
                        else
                        {
                            Console.WriteLine($"Папка для зображень існує:
{imageDirectory}");
                        }

                        try
                        {
                            using (var stream = new FileStream(filePath,
FileMode.Create))
                            {
                                await image.CopyToAsync(stream);
                            }
                            Console.WriteLine($"Зображення збережено за шляхом:
{filePath}");

                            medicine.ImageUrl = $"/images/{fileName}";
                            Console.WriteLine($"Присвоєно шлях зображення до
моделі: {medicine.ImageUrl}");
                            ModelState.Remove("ImageUrl");
                        }
```

```
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Помилка при збереженні
зображення: {ex.Message}");
                        ModelState.AddModelError("ImageUrl", "Помилка при
збереженні зображення.");
                    }
                }
                else
                {
                    Console.WriteLine($"Невірний формат файлу:
{image.ContentType}");
                    ModelState.AddModelError("ImageUrl", "Файл не є
зображенням.");
                }

                foreach (var modelError in ModelState.Values.SelectMany(v =>
v.Errors))
                {
                    Console.WriteLine($"Помилка: {modelError.ErrorMessage}");
                }

                if (!ModelState.IsValid)
                {
                    Console.WriteLine("Модель некоректна, повертаємо на
форму.");
                    return View(medicine);
                }

                _pharmacyService.AddMedicine(medicine);

                var medicines = _pharmacyService.GetMedicines();
                foreach (var med in medicines)
                {
                    Console.WriteLine($"Medicine added to memory: {med.Name},
{med.Description}, {med.ImageUrl}");
                }

                return RedirectToAction("Dashboard");
            }

            [HttpGet]
            public IActionResult Login()
```

```
            {
                return View();
            }
            [HttpPost]
            public IActionResult Login(string username, string password)
            {
                if (username == "admin" && password == "admin")
                {
                    HttpContext.Session.SetString("IsAdmin", "true");
                    return RedirectToAction("Dashboard");

                }

                ViewBag.ErrorMessage = "Невірний логін або пароль";
                return View();
            }
            [HttpPost]
            public JsonResult Delete(int id)
            {
                var medicineToRemove =
_pharmacyService.GetMedicines().FirstOrDefault(m => m.Id == id);
                if (medicineToRemove != null)
                {
                    _pharmacyService.DeleteMedicine(id);
                    return Json(new { success = true });
                }
                return Json(new { success = false });
            }

        }
    }
```

Додаток В

Повний код Services

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.Json;
using PharmacyInfoSystem.Models;


namespace PharmacyInfoSystem.Services
{
    public class PharmacyService
    {
        private List<Customer> _customers = new List<Customer>();
        private List<Medicine> _medicines = new List<Medicine>();

        private static readonly string DataDirectory =
Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "data");

        private static readonly string CustomersFile =
Path.Combine(DataDirectory, "customers.json");
        private static readonly string MedicinesFile =
Path.Combine(DataDirectory, "medicines.json");
        private int _nextId = 1;

        // Властивість для отримання списку клієнтів
        public IEnumerable<Customer> Customers => _customers;
        // Властивість для отримання списку ліків
        public IEnumerable<Medicine> Medicines => _medicines;

        public PharmacyService()
        {
            LoadCustomers();
            LoadMedicines();
        }
        //Завантаження клієнтів з JSON
        private void LoadCustomers()
        {
            if (File.Exists(CustomersFile))
```

```
                {
                    string json = File.ReadAllText(CustomersFile);
                    _customers =
JsonSerializer.Deserialize<List<Customer>>(json) ?? new List<Customer>();
                }
            }
            //Збереження клієнтів у JSON
            private void SaveCustomers()
            {
                string json = JsonSerializer.Serialize(_customers, new
JsonSerializerOptions { WriteIndented = true });
                File.WriteAllText(CustomersFile, json);
            }
            // Метод для перевірки наявності клієнта за номером телефону
            public bool IsCustomerRegistered(string phone)
            {
                return _customers.Any(c => c.Phone == phone);
            }
            //Додавання нового клієнта
            public void AddCustomer(Customer customer)
            {
                _customers.Add(customer);
                SaveCustomers();
            }

            // Метод для додавання ліків
            public void AddMedicine(Medicine medicine)
            {
                Console.WriteLine($"Adding medicine: {medicine.Name}, " +
                                  $"{medicine.Description}, " +
                                  $"{medicine.Price}");
                medicine.Id = _nextId++;
                _medicines.Add(medicine);
                SaveMedicines();
                Console.WriteLine($"Ліки додано: {medicine.Name}");
            }

            // Метод для отримання списку ліків
            public List<Medicine> GetMedicines()
            {
                return _medicines;
            }
```

```
        // Метод для авторизації клієнта
        public string AuthorizeCustomer(string name, string phone)
        {
            var customer = _customers.FirstOrDefault
                (c => c.Name == name && c.Phone == phone);
            if (customer != null)
            {
                return "Авторизація успішна";
            }
            else
            {
                return "Клієнта з таким ім'ям та номером не знайдено.";
            }
        }
        //Завантаження ліків з JSON
        private void LoadMedicines()
        {
            if (File.Exists(MedicinesFile))
            {
                string json = File.ReadAllText(MedicinesFile);
                _medicines =
JsonSerializer.Deserialize<List<Medicine>>(json) ?? new List<Medicine>();

                if (_medicines.Any())
                {
                    _nextId = _medicines.Max(m => m.Id) + 1;
                }
            }
        }
        //Збереження ліків у JSON
        private void SaveMedicines()
        {
            string json = JsonSerializer.Serialize(_medicines, new
JsonSerializerOptions { WriteIndented = true });
            File.WriteAllText(MedicinesFile, json);
        }

        // Метод для видалення ліків за ID
        public void DeleteMedicine(int id)
        {
            var medicine = _medicines.FirstOrDefault(m => m.Id == id);
            if (medicine != null)
            {
```

```
                    _medicines.Remove(medicine);
                    SaveMedicines();
                }
                else
                {
                    // Логіка для обробки ситуації, коли ліки з таким ID не
знайдено
                    Console.WriteLine($"Medicine with ID {id} not found.");
                }
            }

        }
    }
```

Додаток Г

Стилі сайту

```
/* Основний фон сайту */
body {
    background-color: #f4f9f4;
    background-image: url('/images/fon.jpg');
    background-size: cover;
    background-position: center;
    background-attachment: fixed;
    background-repeat: no-repeat;
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}

/* Заголовки */
h1, h2, h3 {
    color: #2e8b57;
    text-align: center;
}
/* Основний контейнер, щоб врахувати футер */
.container {
    padding-bottom: 80px;
    min-height: 100vh;
}

/* Кнопки */
.btn {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    text-decoration: none;
    display: inline-block;
    transition: background-color 0.3s ease, box-shadow 0.3s ease;
```

```
}

    .btn:hover {
        background-color: #45a049;
        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
    }

/* Лінки */
a {
    color: #4CAF50;
    text-decoration: none;
    transition: color 0.3s ease;
}

    a:hover {
        color: #45a049;
        text-decoration: underline;
    }

/* Стилізація форми */
form {
    width: 100%;
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #fff;
}

    form div {
        margin-bottom: 15px;
    }

    form label {
        display: block;
        font-weight: bold;
        margin-bottom: 5px;
    }

    form input {
        width: 100%;
        padding: 8px;
```

```
        border: 1px solid #ddd;
        border-radius: 4px;
    }

    form button {
        padding: 10px 20px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

        form button:hover {
            background-color: #45a049;
        }

/* Кнопка повернення */
.btn-primary {
    background-color: #007bff;
}

    .btn-primary:hover {
        background-color: #0056b3;
    }

/*****************************/

/* Ліки в горизонтальному списку */
.medicine-list {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    padding: 20px;
    justify-content: center;
}

.medicine-item {
    display: flex;
    flex-direction: column;
    width: 250px;
    height: 400px;
    border: 1px solid #ddd;
```

```
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    background-color: #fff;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
    cursor: pointer;
}

    .medicine-item:hover {
        transform: translateY(-5px);
        box-shadow: 0 6px 15px rgba(0, 0, 0, 0.1);
    }

.medicine-img {
    width: 100%;
    height: 200px;
    overflow: hidden;
    position: relative;
}

    .medicine-img img {
        width: 100%;
        height: 100%;
        object-fit: cover;
        transition: transform 0.3s ease;
    }

    .medicine-img:hover img {
        transform: scale(1.1);
    }

.medicine-details {
    padding: 15px;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    height: 150px;
}

.medicine-name {
    font-size: 1.2em;
    margin-bottom: 10px;
    font-weight: bold;
```

```
}

.medicine-description {
    font-size: 0.9em;
    margin-bottom: 10px;
    color: #555;
}

.medicine-price {
    font-size: 1.1em;
    font-weight: bold;
    color: #007bff;
}

.medicine-actions {
    padding: 10px;
    text-align: center;
}

.delete-btn {
    color: red;
    text-decoration: none;
    font-weight: bold;
}

    .delete-btn:hover {
        text-decoration: underline;
    }

/* Стилі для маленьких екранів */
@media (max-width: 768px) {
    .medicine-item {
        width: 100%;
        height: auto;
    }
}
/************************************************/
/* Загальний стиль для таблиці */
.table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
```

```
}

    .table th, .table td {
        padding: 10px;
        text-align: left;
        border: 1px solid #ddd;
    }

    .table th {
        background-color: #f4f9f4;
        font-weight: bold;
    }

    .table tr:nth-child(even) {
        background-color: #f9f9f9;
    }

    .table tr:hover {
        background-color: #f1f1f1;
    }

/* Стиль для кнопки повернення */
.btn-primary {
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 5px;
    font-size: 16px;
}

    .btn-primary:hover {
        background-color: #0056b3;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
    }

/* Стилізація заголовків */
h2 {
    color: #2e8b57;
    text-align: center;
    margin-bottom: 20px;
}
/* Оформлення шапки */
```

```
.header {
    background-color: #4CAF50;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px 30px;
    color: white;
    position: relative;
}

/* Логотип */
.logo {
    font-size: 24px;
    font-weight: bold;
}

/* Кнопка меню */
.menu-btn {
    background: none;
    border: none;
    color: white;
    font-size: 24px;
    cursor: pointer;
}

/* Стилізація випадаючого меню */
.menu {
    position: absolute;
    top: 60px;
    right: 20px;
    background-color: #4CAF50;
    padding: 10px;
    border-radius: 5px;
    display: none;
}

    .menu ul {
        list-style: none;
        margin: 0;
        padding: 0;
    }

        .menu ul li {
```

```
    padding: 10px;
}

    .menu ul li a {
        color: white;
        text-decoration: none;
    }

        .menu ul li a:hover {
            text-decoration: underline;
        }
```

Додаток Д

Повний код Program.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using PharmacyInfoSystem.Services;
using System.Net;

namespace PharmacyInfoSystem
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Додаємо логування в консоль
            builder.Logging.AddConsole();

            // Додаємо служби для підтримки локалізації та MVC
            builder.Services.AddLocalization(options =>
options.ResourcesPath = "Resources");
            builder.Services.AddControllersWithViews()
                .AddDataAnnotationsLocalization()
                .AddViewLocalization();

            // Додаємо підтримку сесій
            builder.Services.AddDistributedMemoryCache();
            builder.Services.AddSession(options =>
            {
                options.IdleTimeout = TimeSpan.FromMinutes(30);
                options.Cookie.HttpOnly = true;
                options.Cookie.IsEssential = true;
            });

            // Додаємо службу PharmacyService
            builder.Services.AddSingleton<PharmacyService>();

            var app = builder.Build();
```

```
                    // Використання сесії
                    app.UseSession();

                    // Використання статичних файлів (CSS, JS)
                    app.UseStaticFiles(new StaticFileOptions
                    {
                        ServeUnknownFileTypes = true,
                        DefaultContentType = "image/png"
                    });

                    // Додаємо підтримку маршрутизації
                    app.UseRouting();

                    // Додаємо маршрути
                    app.MapControllerRoute(
                    name: "admin",
                    pattern:
"admin/{controller=Admin}/{action=Dashboard}/{id?}");

                    app.MapControllerRoute(
                        name: "default",
                        pattern: "{controller=Home}/{action=Index}/{id?}");

                    // Запуск додатку
                    app.Run();
                }
            }
        }
```