Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 «Інженерія програмного забезпечення»

На тему: *Розробка інформаційно-довідкової системи лікарських засобів*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-21-1
_____ *Веризуб М.І.*

Керівник кваліфікаційної роботи _____ */ Гриценко А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2025

Криворізький національний університет
Факультет: *Інформаційних технологій*
Кафедра: *Моделювання та програмного забезпечення*
Освітньо-кваліфікаційний рівень: *бакалавр*
Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу
студенту групи ІПЗ-21-1 Веризуб Максиму Ігоровичу

1. Тема: *Розробка інформаційно-довідкової системи лікарських засобів* затверджено наказом по КНУ №119 від «10» квітня 2025 р.
2. Термін подання студентом закінченого проекту «06» червня 2025 р.
3. Вихідні дані по роботі: *Пояснювальна записка: 57 сторінок, 23 рисунків, 1 додаток, 21 використаних у роботі джерел*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Огляд інформаційно-довідкових систем в галузі лікарських засобів. Проектування, розробка та тестування системи.
5. Перелік графічного демонстраційного матеріалу: Приклади існуючих програмних рішень. Архітектура системи. Структура бази даних. Блок-схеми основних алгоритмів. Приклади роботи розробленої системи (інтерфейс користувача програми).

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*13.02.2025-20.02.2025*
2	*Аналіз інформаційних джерел*	*21.02.2025-09.03.2025*
3	*Визначення вимог до програмного забезпечення*	*10.03.2025-18.03.2025*
4	*Розробка архітектури програмного продукту*	*19.03.2025-23.03.2025*
5	*Розробка алгоритмів*	*24.03.2025-31.03.2025*
6	*Розробка бази даних*	*01.04.2025-14.04.2025*
7	*Розробка програмного забезпечення*	*15.04.2025-13.05.2025*
8	*Тестування програмного забезпечення*	*14.05.2025-20.05.2025*
9	*Оформлення пояснювальної записки*	*21.05.2025-12.06.2025*
10	*Розробка демонстраційних матеріалів*	*13.06.2025-17.06.2025*

Дата видачі завдання: «_____»_____ 2025 р.

Студент _____ Веризуб М.І.

Керівник роботи _____ Гриценко А. М.

РЕФЕРАТ

ПІДБІР ЛІКАРСЬКИХ ЗАСОБІВ, ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА, СУМІСНІСТЬ ПРЕПАРАТІВ, НЕЙРОМЕРЕЖЕВІ РЕКОМЕНДАЦІЇ, ІНТЕГРАЦІЯ З AI, СКБД POSTGRESQL, REST API

Пояснювальна записка: 57 с., 6 табл., 20 рис., 1 дод., 20 джерел.

Метою кваліфікаційної роботи є розробка клієнт-серверного програмного забезпечення для інтелектуального підбору лікарських засобів з урахуванням діючої речовини, взаємної сумісності препаратів, регіональних особливостей та рекомендацій на основі штучного інтелекту.

У роботі проведено аналіз предметної області та існуючих інформаційних систем у фармацевтичній сфері. Визначено функціональні та нефункціональні вимоги до системи, розроблено архітектуру багаторівневого застосунку з поділом на клієнтську, серверну та інтелектуальну підсистеми.

Серверна частина реалізована з використанням PostgreSQL як реляційної бази даних і розгорнута на хмарному сервері з підтримкою багатокористувацького доступу. Застосовано REST API для взаємодії клієнта з сервером та зовнішнім AI-модулем. Клієнтський застосунок реалізовано на мові C# у середовищі Visual Studio з використанням WinForms.

Передбачено авторизацію користувачів за ролями (користувач, фармацевт, адміністратор), формування індивідуальних запитів до системи, перевірку сумісності препаратів, а також діалоговий інтерфейс AI-помічника, що збирає симптоми пацієнта та надає рекомендації. Особливу увагу приділено адаптивному дизайну інтерфейсу, зручності використання та масштабованості системи.

Основні результати підтверджено тестуванням функціональності та сумісності компонентів системи. Запропоноване рішення може бути використане в аптечних мережах, електронних медичних сервісах, а також у вигляді інтеграційного модуля до існуючих медичних інформаційних платформ.

ABSTRACT

SELECTION OF MEDICINES, GRAPHICAL USER INTERFACE, COMPATIBILITY OF MEDICINES, NEURAL NETWORK RECOMMENDATIONS, INTEGRATION WITH AI, POSTGRESQL DBMS, REST API

Explanatory note: 57 p., 6 tables, 20 figures, 1 appendices, 20 sources.

The purpose of the qualification work is to develop client-server software for intelligent selection of medicines taking into account the active substance, mutual compatibility of drugs, regional features and recommendations based on artificial intelligence.

The work analyzes the subject area and existing information systems in the pharmaceutical sector. Functional and non-functional requirements for the system are determined, and the architecture of a multi-level application is developed with a division into client, server and intelligent subsystems.

The server part is implemented using PostgreSQL as a relational database and deployed on a cloud server with multi-user access support. REST API is used for client interaction with the server and external AI module. The client application is implemented in C# in the Visual Studio environment using WinForms.

User authorization by roles (user, pharmacist, administrator), generation of individual requests to the system, checking drug compatibility, as well as a dialog interface for an AI assistant that collects patient symptoms and provides recommendations are provided. Special attention is paid to adaptive interface design, ease of use and scalability of the system.

The main results are confirmed by testing the functionality and compatibility of system components. The proposed solution can be used in pharmacy chains, electronic medical services, and also as an integration module for existing medical information platforms.

ЗМІСТ

ВСТУП

Сучасна система охорони здоров’я перебуває у стані динамічного розвитку, значною мірою завдяки впровадженню цифрових технологій, автоматизації та штучного інтелекту. Це стосується не лише великих медичних закладів, а й аптечних мереж, лікарів загальної практики, фармацевтів, а також самих пацієнтів. У зв’язку з цим особливої актуальності набуває створення спеціалізованих програмних рішень, які сприяють полегшенню прийняття рішень у медичній практиці, зокрема — при підборі лікарських засобів.

Підбір препаратів — критично важливий процес, який вимагає врахування великої кількості факторів: діюча речовина, форма випуску, протипоказання, побічні ефекти, можливі взаємодії з іншими ліками, виробник, регіональна доступність, вік та фізіологічні особливості пацієнта. Усі ці параметри мають бути проаналізовані за короткий проміжок часу, що в умовах великого потоку клієнтів, дефіциту персоналу або браку часу створює ризик для здоров’я пацієнтів. Крім того, лікарі та фармацевти часто використовують різні джерела інформації: довідники, інтернет-ресурси, внутрішні системи, що ускладнює доступ до єдиного, централізованого інструменту прийняття рішень.

Водночас, в умовах зростаючої кількості лікарських засобів на ринку та постійного оновлення нормативної бази, фармацевтам та лікарям необхідно мати під рукою гнучкий інструмент для пошуку аналогів, перевірки сумісності та формування рекомендацій для пацієнтів. У цьому контексті особливо перспективною є інтеграція інформаційної системи з модулем штучного інтелекту (AI), який здатен аналізувати введені користувачем симптоми, задавати уточнюючі запитання, враховувати побажання або індивідуальні характеристики пацієнта, що забезпечує більш точний, персоналізований підхід до лікування.

У більшості сучасних медичних або аптечних інформаційних систем відсутні інструменти комплексного підбору лікарських засобів, які поєднують як фільтрацію за технічними параметрами, так і експертні рекомендації на основі знань про дію речовин. Окремі системи орієнтовані лише на довідкову інформацію, або не мають відкритого доступу, або не підтримують масштабування на багато клієнтів одночасно. У зв'язку з цим постає реальна потреба у створенні власної системи з відкритою архітектурою, яка дозволяє легко розширювати функціонал, інтегрувати нові джерела даних та працювати у хмарному середовищі.

З урахуванням вищезазначеного, у межах даної кваліфікаційної роботи передбачено розробку інформаційної системи підбору лікарських засобів, що складається з графічного клієнтського застосунку, серверної частини з базою даних та зовнішнього модуля AI-помічника. Система орієнтована на багатокористувацьку архітектуру та підтримує ролі «користувач», «фармацевт» і «адміністратор», кожна з яких має відповідні права доступу та інтерфейс. Особливістю проєкту є інтеграція з модулем штучного інтелекту через API, що дозволяє зберегти модульність системи та забезпечити гнучке оновлення знань. Також передбачено налаштування віддаленого підключення до бази даних через захищене мережеве з'єднання, що відкриває перспективи масштабування системи в майбутньому.

Таким чином, розробка програмного забезпечення для підбору лікарських засобів з AI-помічником є актуальним інженерним завданням, що відповідає сучасним вимогам до медичних інформаційних систем. Очікуваний результат — створення повноцінного, функціонального та масштабованого продукту, який може бути використаний як у навчальних цілях, так і як прототип реального комерційного рішення.

1 ОГЛЯД ІНФОРМАЦІЙНО-ДОВІДКОВИХ СИСТЕМ В ГАЛУЗІ ЛІКАРСЬКИХ ЗАСОБІВ

1.1 Особливості предметної області в галузі підбору лікарських засобів

Підбір лікарських засобів є складним багатофакторним процесом, який передбачає врахування фармакологічних, клінічних та організаційних аспектів. Його мета — забезпечити максимально ефективне та безпечне медикаментозне лікування пацієнта з урахуванням його індивідуальних особливостей, супутніх захворювань, віку, стану здоров’я та наявності інших лікарських призначень. В основі цього процесу лежить правильне співвідношення між діючою речовиною, формою випуску препарату, дозуванням, способом введення, а також оцінкою ймовірних побічних ефектів та взаємодій із вже призначеними або вживаними ліками.

Одним із ключових факторів у виборі препарату є діюча речовина, що визначає фармакологічний ефект. Проте на ринку існує багато торгових назв, які можуть містити ту саму діючу речовину, але відрізнятися за допоміжними компонентами, які можуть викликати алергічні реакції або впливати на засвоєння препарату. Крім того, форма випуску (таблетки, капсули, сироп, мазь, ін'єкції) може бути критично важливою залежно від віку пацієнта або специфіки захворювання.

Ще одним важливим аспектом є взаємодія препаратів, особливо у випадках поліпрепаратної терапії. Неправильне поєднання ліків може призвести до зниження ефективності одного з них або, навпаки, до токсичних ефектів. У зв’язку з цим необхідно проводити перевірку сумісності лікарських засобів до їх призначення. Також враховується потреба в рецептурному контролі, адже частина препаратів може бути призначена виключно лікарем.

Регіональний аспект також відіграє важливу роль — доступність препаратів у конкретному регіоні, середня ринкова ціна, а також локальні

стандарти призначення можуть значно варіюватися. У деяких регіонах надається перевага препаратам локального виробника, або ж певні торгові марки взагалі не представлені на ринку.

У процесі підбору лікарських засобів беруть участь три основні групи користувачів: лікарі, які формують призначення відповідно до медичних протоколів; фармацевти, які можуть пропонувати альтернативи на основі наявності та вартості; та пацієнти, які прагнуть обрати безпечний, ефективний і доступний варіант. Невід'ємною частиною проблеми є й спроби самостійного підбору препаратів пацієнтами без фахових знань, що нерідко призводить до помилок, самолікування або нехтування протипоказаннями.

Таким чином, автоматизація процесу підбору лікарських засобів з використанням сучасних інформаційних технологій, баз знань і допомоги AI-помічника дозволяє не лише підвищити точність і безпечність призначень, але й адаптувати вибір до умов конкретного користувача — з урахуванням доступності препаратів, їх вартості, медичних обмежень і персональних потреб.

1.2 Критичний аналіз існуючих програмних рішень в галузі лікарських засобів

У процесі розробки інформаційної системи доцільним є вивчення наявних аналогів на ринку медичного та фармацевтичного програмного забезпечення. Це дозволяє виявити сильні сторони вже реалізованих проєктів, визначити їхні недоліки та сформулювати чіткі вимоги до функціональності та архітектури системи, що створюється в межах кваліфікаційної роботи.

Одним із найвідоміших сервісів в Україні є "DЛІКИ" (https://liki24.com) головна сторінка сайту наведена на рисунку 1.1, який є електронною платформою для пошуку та порівняння цін на лікарські засоби у конкретному регіоні. Сервіс надає доступ до бази препаратів, дозволяє фільтрувати їх за назвою або речовиною, але не має можливості перевірки взаємодії ліків або отримання рекомендацій. Відсутня підтримка ролей користувачів або

підключення до зовнішніх баз даних, що обмежує можливості інтеграції в комплексні медичні системи.

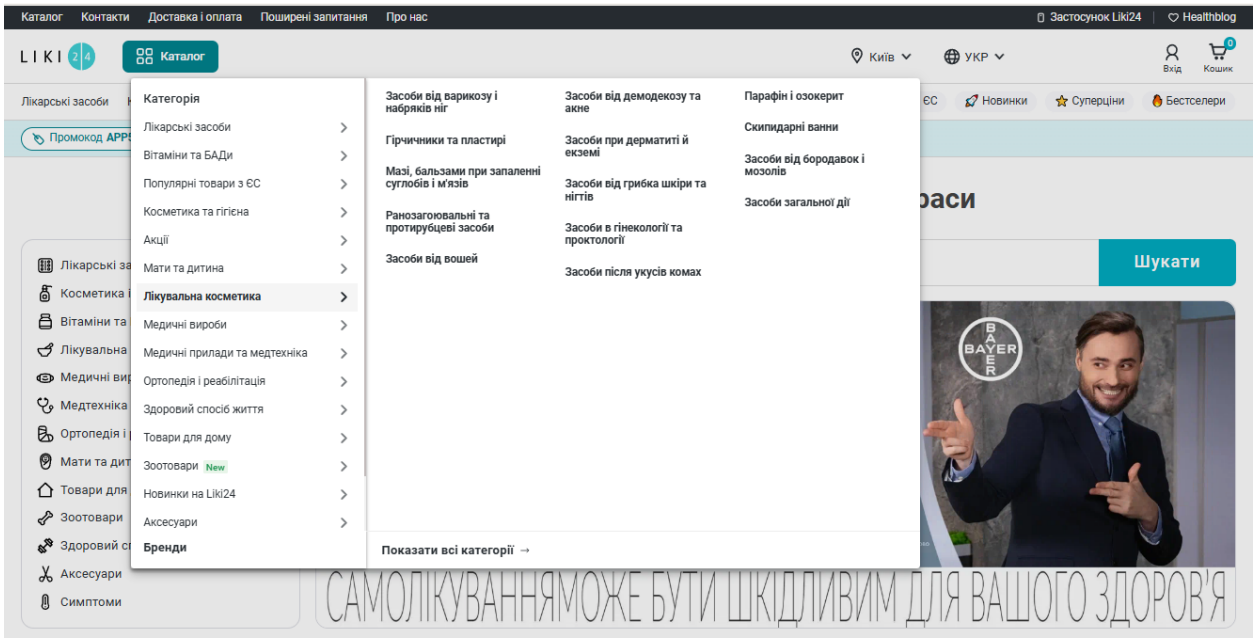

Рисунок 1.1 – Візуальне представлення головної сторінки сервісу «Ліки24»

Сайт RxList (https://rxlist.com) (див. рис. 1.2) є американським ресурсом, який надає довідкову інформацію про діючі речовини, препарати, побічні ефекти та дозування. Він популярний серед англомовної аудиторії, але не має української локалізації чи врахування регіональної доступності ліків. Частково реалізована перевірка взаємодій, проте відсутня інтелектуальна підтримка користувача.

Платформа Drugs.com (https://drugs.com) також англомовна, містить інструмент перевірки сумісності лікарських засобів (Drug Interaction Checker), зручну систему навігації та мобільні додатки, головна сторінка веб-сервісу наведено на рисунку 1.3. Незважаючи на функціональність, сайт не підтримує AI-помічники, не адаптований до фармацевтичного ринку України, не дозволяє налаштування під користувача чи інтеграцію з локальними базами.

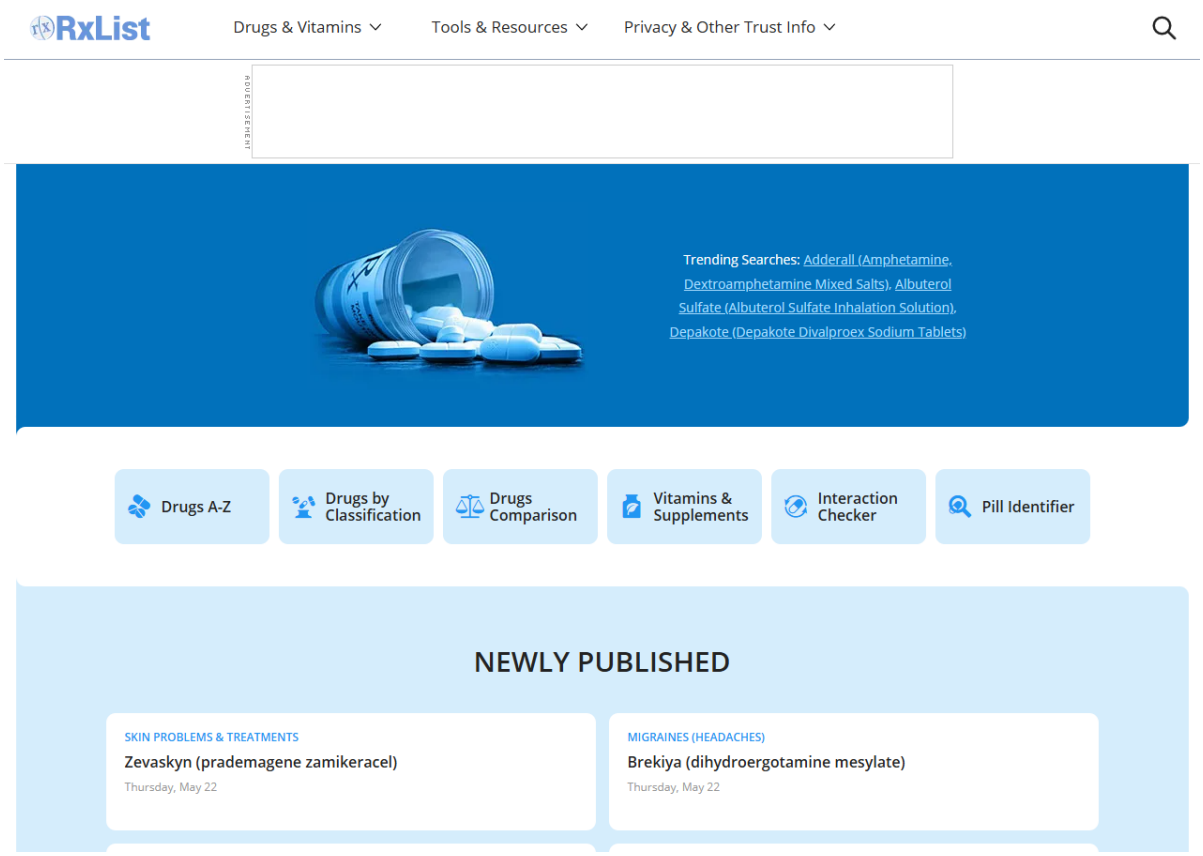

Рисунок 1.3 – Візуальне представлення головної сторінки сайту RxList.com

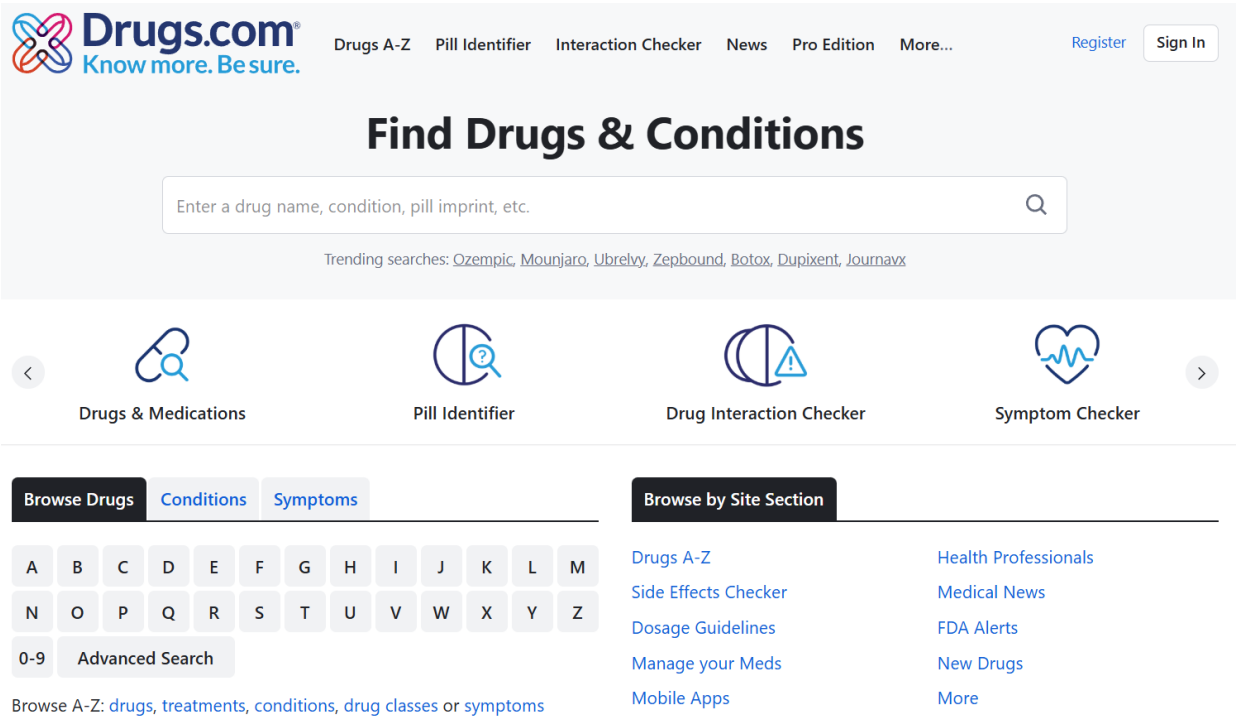

Рисунок 1.3 – Візуальне представлення головної сторінки платформи Drugs.com

Локальні рішення, створені на базі MS Excel або MS Access, досить поширені серед приватних аптек. Вони можуть включати внутрішні списки препаратів, дозволяють фільтрацію за певними параметрами. Проте такі системи є одноосібними, не масштабуються, не мають рівнів доступу і практично не підтримують багатокористувацький режим.

На основі аналізу було складено порівняльну таблицю (див. табл. 1) , в яку включено ключові параметри оцінювання.

Таблиця 1.1 - Порівняльна таблиця систем підбору лікарських засобів

Система	**Мова інтерфейсу**	**Пошук за назвою/речо**	**Перевірка взаємодії**	**AI-рекомендації**	**Регіональна доступність**	**Рольовий доступ**	**Інтеграція з БД**	**Масштабо-ваність**
DЛІКИ (liki24.com)	украї-нська	так	ні	ні	так	ні	част-ково	обме-жена
RxList (rxlist.com)	англій-ська	так	част-ково	ні	ні	ні	ні	обме-жена
Drugs.com (drugs.com)	англій-ська	так	так	ні	ні	ні	ні	обме-жена
Локальні рішення (Excel/Access)	украї-нська	так	ні	ні	част-ково	ні	част-ково	відсут-ня
Розроблювана система	украї-нська	так	так	так	так	так	так	висока

Проведено комплексне узагальнення переваг та недоліків відомих прогармних рішень. Серед переваг виділено основні:

- доступ до бази препаратів;
- фільтрація за назвою або речовиною;

- наявність мобільних версій (у деяких системах);
- простота використання;
- часткове відображення наявності ліків у регіонах.

Серед недоліків виділено основні:

- відсутність AI-рекомендацій
- відсутність підтримки ролей
- неможливість підключення до власних БД
- відсутність гнучкої архітектури та масштабованості
- орієнтація лише на довідкову функцію

1.3 Узагальнення функціональних вимог до системи та формування завдання для розробки системи

На основі аналізу предметної області підбору лікарських засобів та огляду існуючих програмних рішень сформовано перелік функціональних вимог, які визначають архітектуру та логіку розроблюваної інформаційної системи. Система має бути орієнтована на зручність використання як для пацієнтів, так і для фармацевтів та адміністраторів, забезпечуючи швидкий, надійний і безпечний доступ до фармакологічної інформації, з урахуванням особливостей кожного користувача.

Однією з основних вимог є реалізація розширеного пошуку лікарських засобів за декількома критеріями. Користувач повинен мати можливість здійснювати пошук за назвою препарату, діючою речовиною, формою випуску, країною виробника, наявністю в певному регіоні, середньою вартістю та іншими додатковими параметрами. Це дозволяє враховувати не лише медичні, а й економічні або логістичні аспекти при виборі препарату.

Ключовою функцією системи має стати перевірка сумісності між лікарськими засобами, що базується на аналізі взаємодій між діючими речовинами. Завдяки цьому механізму користувач або фармацевт може отримати попередження про можливі протипоказання чи небажані ефекти при

одночасному вживанні певних препаратів. Для цього використовується база знань, що зберігає правила взаємодії речовин, їх характер і рівень небезпеки.

Окремо слід виділити необхідність урахування регіональної специфіки, зокрема — доступності препаратів у певних географічних зонах, відмінностей у середніх ринкових цінах, а також регіональних обмежень або рекомендацій. Це дозволить зробити систему більш точною, адаптивною до потреб кінцевого користувача, і водночас — корисною для фармацевтичного бізнесу та локальних аптечних мереж.

Інноваційною складовою системи має бути впровадження AI-помічника, який за допомогою інтерактивного опитування користувача уточнює симптоми, побажання та індивідуальні особливості, і на основі цього формує персоналізовані рекомендації. AI повинен бути інтегрований через API до ядра системи та навчений на корпусі знань у сфері фармакології, що дозволить підвищити якість обслуговування користувачів без необхідності прямого втручання медичних працівників.

Функціональність системи передбачає реалізацію авторизації з розподілом ролей: адміністратор, фармацевт, користувач. Кожна роль має різні рівні доступу до функцій системи, можливість перегляду або редагування даних, ведення історії пошуків або змін у базі лікарських засобів. Така рольова модель дозволяє контролювати безпеку, забезпечити правильний обіг інформації та логічно розмежувати обов’язки.

Також передбачається підтримка багатокористувацького доступу до бази даних, що розміщена на сервері, з можливістю підключення великої кількості клієнтів (до 500 одночасно). Система має бути масштабованою, з урахуванням майбутнього зростання обсягів даних та кількості користувачів.

Отже, функціональні вимоги до системи охоплюють як основні механізми підбору та перевірки препаратів, так і сучасні засоби персоналізації, взаємодії користувача з інтерфейсом, безпеки та адміністрування. Ці вимоги визначають архітектуру та склад майбутнього програмного забезпечення, що розробляється в рамках цієї кваліфікаційної роботи.

Висновки за розділом

У ході аналізу предметної області підбору лікарських засобів було встановлено, що процес вибору медикаментів є багатофакторним та складним, оскільки вимагає врахування низки медичних, фармакологічних та адміністративних параметрів. До основних із них належать: діюча речовина, форма випуску, ймовірні побічні ефекти, наявність протипоказань, особливості взаємодії з іншими препаратами, необхідність рецепту, регіональна доступність, середня ринкова вартість, а також індивідуальні характеристики пацієнта. Належне врахування цих чинників дозволяє забезпечити ефективність і безпеку медикаментозної терапії, особливо при одночасному прийомі кількох лікарських засобів. Проте значна частина пацієнтів здійснює самостійний вибір препаратів, що часто призводить до помилок, небажаних ефектів або нераціонального витрачання коштів.

Також було визначено, що більшість сучасних інформаційно-довідкових систем, доступних як в онлайн-форматі, так і у вигляді мобільних додатків, виконують переважно довідкову або рекламну функцію. Вони орієнтовані на пошук препаратів за назвою, перегляд інструкцій, а в деяких випадках — на порівняння цін та наявності їх в певному регіоні. Лише поодинокі рішення передбачають врахування взаємодій діючих речовин, або надають гнучкий механізм підбору за розширеними критеріями. Більшість із них не мають українськомовного інтерфейсу, не враховують регіональної специфіки та не забезпечують персоналізованих рекомендацій, що обмежує їх ефективність для кінцевого користувача. Крім того, наявні сервіси рідко включають AI-компоненти, які могли б автоматизувати процес аналізу симптомів або індивідуальних характеристик пацієнта з метою формування більш точних рекомендацій.

На основі виконаного аналізу було узагальнено функціональні вимоги до нової системи, яка розробляється в межах цього проєкту. Вона має забезпечувати зручний та гнучкий пошук лікарських засобів за низкою

параметрів — назвою, діючою речовиною, формою випуску, побічними ефектами, ціною, регіональною доступністю. Обов’язковим компонентом є система перевірки взаємодії між препаратами, що базується на знаннях про діючі речовини та клінічно значущі протипоказання. Система повинна бути багатокористувацькою, з підтримкою розмежування ролей (адміністратор, фармацевт, звичайний користувач), реалізовувати захищену авторизацію та масштабуватись відповідно до кількості підключень. Окремим інноваційним компонентом стане AI-помічник, який на основі інтерактивного опитування користувача збиратиме дані про симптоми, особливості організму та інші побажання клієнта значно швидше надаватиме персоналізовані рекомендації щодо лікарських засобів.

Таким чином, обґрунтовано необхідність та доцільність створення нової, адаптивної та функціонально розширеної системи підбору лікарських засобів, яка дозволить ефективно автоматизувати ключові аспекти процесу прийняття рішень як для пацієнтів, так і для фармацевтів. Сформульовані вимоги лягли в основу технічного завдання на проєктування, реалізацію та тестування системи.

2 ПРОЕКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ

2.1 Розробка архітектури програмного продукту

Програмне забезпечення системи підбору лікарських засобів побудовано за принципами багаторівневої клієнт-серверної архітектури, що забезпечує гнучкість, масштабованість та надійність у використанні. Такий підхід дозволяє розділити функціональність між клієнтським інтерфейсом користувача, серверною логікою та базою даних, що суттєво спрощує супровід, тестування і подальший розвиток системи.

Система складається з кількох основних компонентів (див. рис. 2.1):

1) клієнтська частина, реалізована з використанням мови C# та технології WinForms, є графічним інтерфейсом користувача, через який здійснюється введення запитів, перегляд результатів, діалог з AI-помічником, а також керування обліковими даними. Вона має розділення доступу за ролями: користувач, фармацевт, адміністратор;
2) СКБД PostgreSQL виступає центральним сховищем усіх даних: лікарських засобів, діючих речовин, сумісності, середніх цін, налаштувань та іншої довідкової інформації. Вибір PostgreSQL обґрунтований високою продуктивністю, підтримкою транзакцій, надійністю та широкими можливостями розширення;
3) AI-помічник, реалізований як окремий мікросервіс або API, обробляє запити від клієнтської програми, задає уточнюючі запитання та повертає рекомендації, базуючись на знаннях про дію речовин, симптоми та побажання пацієнта. Зв’язок із клієнтом здійснюється через HTTP-протокол.

Така архітектура дозволяє гнучко масштабувати систему: додавати нові клієнтські програми, оновлювати серверну логіку або базу знань без переривання роботи користувачів. Застосування централізованої бази даних дає змогу підтримувати цілісність інформації, а винесення інтелектуального

модуля в окремий сервіс забезпечує модульність та можливість повторного використання в інших проектах.

Крім того, підтримка віддаленого підключення дозволяє організувати роботу користувачів незалежно від їхнього фізичного розташування — що є критично важливим для медичних установ із географічно розподіленою структурою.

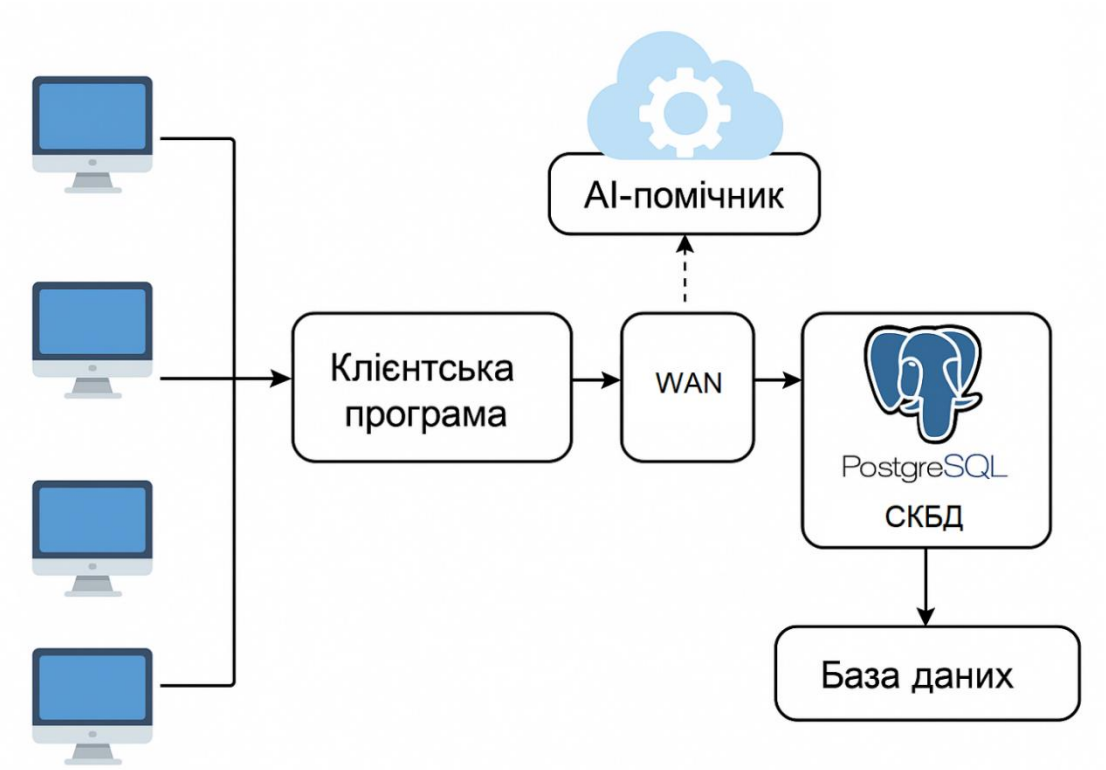

Рисунок 2.1 – Схема архітектури програмного забезпечення.

У процесі розробки програмного забезпечення системи підбору лікарських засобів було побудовано UML-діаграму взаємодії користувачів із формами інтерфейсу, яка відображає логіку переходів між основними компонентами програми, а також рольову модель доступу. Дана діаграма відповідає стандартам Unified Modeling Language (UML) типу «use case / навігаційна логіка» і дозволяє наочно продемонструвати архітектурну взаємодію між користувачами, формами та підсистемами (див. рис. 2.2).

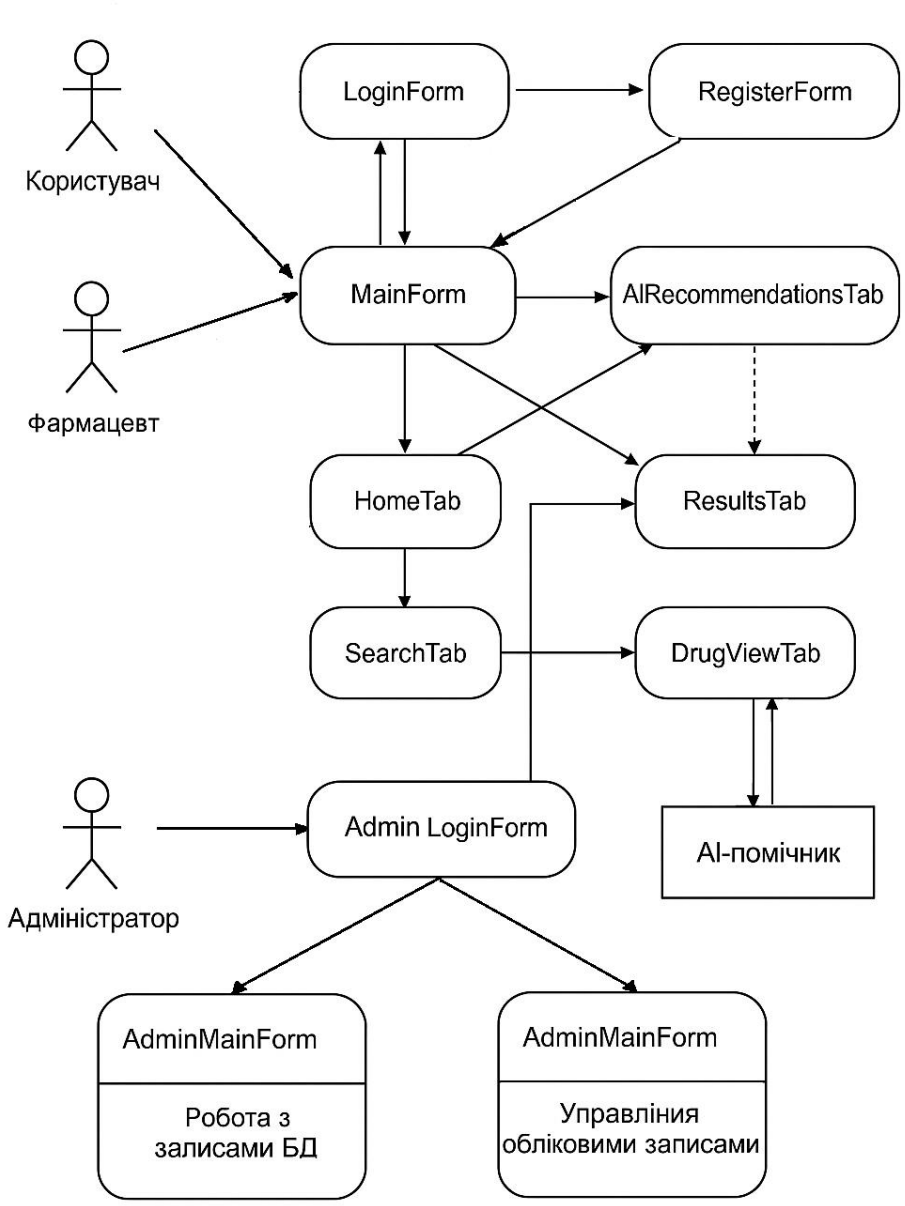

Рисунок 2.2 – UML-діаграма форм системи та переходів між ними.

У системі передбачено три основні ролі користувачів:

Користувач — базовий рівень доступу з можливістю здійснювати пошук лікарських засобів, перевіряти сумісність та отримувати рекомендації від AI-помічника.

Фармацевт — має доступ до додаткової інформації про препарати та можливість переглядати розширені AI-пояснення.

Адміністратор — керує базою даних лікарських засобів і обліковими записами користувачів.

Діаграма починається з етапу авторизації користувача, де визначається роль користувача (Користувач, Фармацевт, Адміністратор) взаємодіють із формами LoginForm та RegisterForm. Після успішного входу кожен користувач перенаправляється до головного вікна системи (MainForm), функціональність якого адаптована до його ролі.

В межах MainForm реалізовані вкладки (внутрішні підсистеми):

HomeTab — початкова панель з привітанням та короткою інформацією;

SearchTab — модуль пошуку лікарських засобів за назвою або діючою речовиною;

ResultsTab — відображення знайдених препаратів, включаючи назви, форму, ціну та країну виробника;

AIRecommendationsTab — взаємодія з AI-помічником, який задає уточнювальні запитання та повертає інтелектуальні рекомендації;

DrugViewTab — вікно перегляду детальної інформації про окремий лікарський засіб.

Особливістю діаграми є наявність зв’язку між вкладкою AIRecommendationsTab та зовнішнім модулем AI-помічника (як зразок AIHelper), що позначає інтеграцію з інтелектуальною підсистемою через API. Це рішення дозволяє гнучко змінювати або розширювати модель без впливу на клієнтську частину застосунку.

Для ролі адміністратора передбачено окреме вікно AdminMainForm, у якому реалізовані вкладки Управління обліковими записами користувачів (UserManagement) та Робота з записами БД (DBEditor). Ці функції дозволяють адміністратору переглядати, редагувати та видаляти користувачів, а також керувати структурою та вмістом бази даних системи.

UML-діаграма демонструє чітко організовану структуру переходів, яка не тільки відповідає вимогам зручності та логіки навігації в програмі, а й забезпечує підтримку масштабованості, модульності та можливості подальшого розширення системи. Такий підхід дозволяє реалізувати якісну

архітектуру користувацького інтерфейсу, що є важливим аспектом у процесі розробки сучасних інформаційних систем у медичній галузі.

2.2 Обґрунтування вибору СКБД PostgreSQL для інформаційної системи підбору лікарських засобів

У процесі проектування інформаційної системи підбору лікарських засобів одним із ключових рішень є вибір системи керування базами даних (СКБД), яка повинна забезпечити надійне зберігання, обробку, масштабування та доступність медичної інформації в багатокористувацькому середовищі. У рамках реалізації даної системи було обрано реляційну СКБД PostgreSQL, яка на сучасному етапі є однією з найпотужніших, стабільних і гнучких систем з відкритим кодом.

PostgreSQL підтримує повну реалізацію стандарту SQL, включаючи розширені можливості роботи з транзакціями, перевірку цілісності даних, зовнішні ключі, представлення (views), індексацію, тригери, функції та збережені процедури. Це дозволяє проєктувати надійні та строго структуровані бази даних, що особливо важливо у сфері охорони здоров’я, де мають значення точність, достовірність і відтворюваність даних.

Серед головних переваг PostgreSQL варто відзначити високу масштабованість — як вертикальну (за рахунок оптимізації запитів, розподіленої обробки), так і горизонтальну (через можливість шардінгу та реплікації). Це означає, що система може ефективно працювати як у середовищі з невеликою кількістю користувачів, так і в умовах великої інфраструктури, яка охоплює сотні клієнтів, підключених до одного центрального сервера бази даних.

Іншою важливою перевагою PostgreSQL є надійність і стабільність роботи, підтверджена роками використання в промислових, наукових, державних і медичних системах у всьому світі. PostgreSQL має потужний механізм відновлення після збоїв, журналювання транзакцій (write-ahead

logging), що забезпечує збереження та цілісність даних навіть у разі аварійного вимкнення або втрати з’єднання.

Ще одним аргументом на користь цієї СКБД є її відкритість і безкоштовна ліцензія (PostgreSQL License, схожа на MIT), що дозволяє інтегрувати її в будь-які типи програмного забезпечення — як комерційні, так і відкриті. Це особливо важливо у випадку створення кваліфікаційної роботи бакалавра, яка може бути в майбутньому розгорнута як прототип або навіть дослідницький продукт без додаткових витрат на ліцензування.

Також варто відзначити широкі можливості розширення PostgreSQL: підтримка користувацьких типів даних, власних функцій, підключення мов програмування (PL/pgSQL, Python, SQL, C), модулів (наприклад, PostGIS — для геопросторової інформації), а також інтеграції з API на стороні клієнта (наприклад, через pgAdmin, DBeaver або з клієнтських програм на C#, Java, Python тощо).

Інформаційна система підбору лікарських засобів побудована за клієнт-серверною архітектурою, що передбачає підключення великої кількості клієнтів (до 500 одночасно) до центрального сервера PostgreSQL, розгорнутого у хмарному середовищі (наприклад, DigitalOcean, AWS, Heroku або інший хостинг).

Підключення клієнтської програми, написаної мовою C# з використанням WinForms, до віддаленої бази PostgreSQL відбувається за допомогою драйвера Npgsql — це офіційний .NET-драйвер для PostgreSQL. Драйвер реалізує всі необхідні протоколи підключення та дозволяє легко інтегрувати запити SQL безпосередньо у код програми або через ORM (наприклад, Entity Framework).

Стандартний рядок підключення до PostgreSQL виглядає наступним чином:

```
string connectionString =
"Host=your_server_address;Port=5432;
Username=your_user;Password=your_password;
Database=your_database;";
```

Усі ці параметри можуть зберігатися у файлі конфігурації або передбачені як налаштування у відповідній вкладці «Налаштування» графічного інтерфейсу користувача. Під час першого запуску або при зміні сервера, користувач має змогу ввести:

```
Адресу сервера (hostname/IP);
Порт (стандартний — 5432);
Ім’я користувача бази даних;
Пароль доступу;
```

З технічного боку підключення виконується через TCP/IP, і необхідно дозволити вхідні з’єднання на сервері PostgreSQL з відповідної IP-адреси клієнтів (через редагування файлів pg_hba.conf та postgresql.conf).

Для гарантії захищеного з'єднання PostgreSQL підтримує SSL/TLS, шифрування з’єднання, а також автентифікацію через сертифікати. Це дозволяє використовувати систему навіть у медичних закладах з підвищеними вимогами до безпеки даних.

Таким чином, вибір PostgreSQL як основної СКБД для даної системи обумовлений її функціональною повнотою, високою продуктивністю, безкоштовною ліцензією, підтримкою масштабування, стабільністю роботи та активною спільнотою розробників. Завдяки відкритим стандартам та підтримці віддаленого підключення, PostgreSQL ідеально підходить для реалізації розподілених інформаційних систем з великою кількістю одночасних користувачів, що в повній мірі відповідає вимогам до системи підбору лікарських засобів.

2.3 Проектування структури бази даних

На етапі розробки структури бази даних для системи підбору лікарських засобів було враховано функціональні вимоги, сформульовані під час аналізу предметної області, та необхідність підтримки гнучких запитів до фармацевтичної інформації. База даних є центральним компонентом програмного продукту, оскільки зберігає повну інформацію про лікарські

засоби, діючі речовини, регіональні особливості, користувачів, історію пошуків та взаємодії між препаратами.

Розробка структури бази даних здійснювалась із застосуванням принципів нормалізації, що дозволило уникнути надмірного дублювання даних, забезпечити логічну цілісність і підвищити продуктивність запитів. У якості системи керування базами даних було обрано PostgreSQL, що підтримує транзакційність, реляційні зв'язки, високий рівень безпеки та масштабованість — ключові характеристики для багатокористувацької медичної системи.

Логічна модель бази даних представлена у вигляді ER-діаграми (див. рис. 2.3), що ілюструє основні сутності системи, зв'язки між ними та кардинальності. Це дозволяє візуально відобразити структуру та взаємозалежності даних, що стане основою для подальшої реалізації SQL-структур та програмної логіки системи.

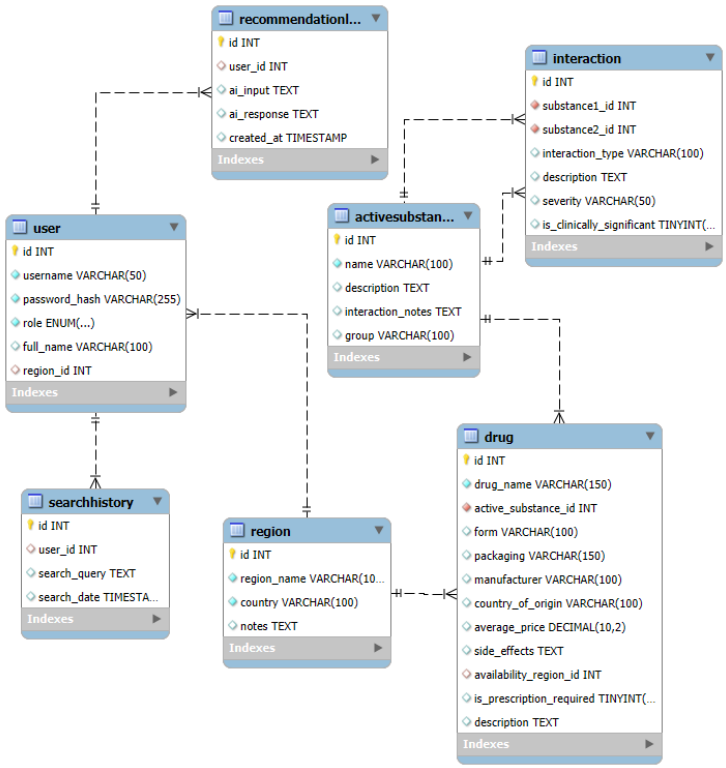

Рисунок 2.3 – ER-діаграма розробленої бази даних

У таблиці User зберігається інформація про зареєстрованих користувачів в системі. Структура таблиці бази даних наведена в таблиці 2.1.

Таблиця 2.1 – Структура таблиці User

Поле	Тип	Опис
user_id	PK, int	Унікальний ідентифікатор користувача
username	string	Ім’я користувача
password_hash	string	Хеш пароля користувача
role	enum (user, pharmacist, admin)	Роль користувача у системі
region_id	FK → Region.region_id	Посилання на регіон користувача

Таблиця Region виконує функцію зберігання довідникової інформації про регіони, до яких належать користувачі або до яких можуть бути прив’язані специфічні властивості лікарських засобів — наприклад, наявність на локальному ринку, середня вартість, особливості сертифікації, локальні обмеження або пріоритетні бренди.

Таким чином, таблиця Region є допоміжною, але критично важливою для адаптації системи до реальних умов, зберігаючи географічний контекст у процесі підбору лікарських засобів і дозволяючи враховувати місцеву специфіку при формуванні рекомендацій.

Таблиця 2.2 – Структура таблиці Region

Поле	Тип	Опис
region_id	PK, int	Унікальний ідентифікатор регіону
name	string	Назва регіону

Таблиця Drug є однією з ключових у структурі бази даних системи підбору лікарських засобів. Вона містить повну інформацію про кожен зареєстрований лікарський засіб, що використовується для пошуку, аналізу сумісності та формування рекомендацій для користувачів або фармацевтів. Ця

таблиця забезпечує основу для зв’язку з іншими сутностями, такими як діюча речовина, форма випуску, виробник, побічні ефекти, країна виробництва та цінова категорія.

Таблиця 2.3 – Структура таблиці Drug

Поле	Тип	Опис
drug_id	PK, int	Унікальний ідентифікатор препарату
name	string	Комерційна назва
active_substance_id	FK → ActiveSubstance. active_substance_id	Посилання на діючу речовину
form	string	Форма випуску
manufacturer_country	string	Країна-виробник
side_effects	text	Побічні ефекти
average_price	decimal	Середня ціна на ринку

Таблиця ActiveSubstance у базі даних системи підбору лікарських засобів виконує роль довідника діючих речовин, які лежать в основі медичних препаратів. Її головне призначення — забезпечити нормалізацію даних і дозволити групування лікарських засобів за основною активною фармакологічною складовою, що є критично важливим для медичної логіки системи.

Замість дублювання інформації про діючі речовини в кожному записі препарату, система використовує зв’язок між таблицею Drug і ActiveSubstance, що дозволяє:

- швидко знаходити всі лікарські засоби з однаковою діючою речовиною;
- аналізувати взаємозаміни при непереносимості певного бренду;

- застосовувати рекомендації штучного інтелекту на рівні речовини, а не конкретного препарату.

Таблиця 2.4 – Структура таблиці ActiveSubstance

Поле	Тип	Опис
active_substance_id	PK, int	Унікальний ідентифікатор речовини
name	string	Назва діючої речовини

Таблиця Interaction у базі даних системи підбору лікарських засобів призначена для зберігання інформації про взаємодію між діючими речовинами, що дозволяє системі перевіряти сумісність лікарських засобів під час формування призначень або рекомендацій.

Її наявність є критично важливою для забезпечення безпечного використання кількох препаратів одночасно, особливо при супутньому лікуванні кількох захворювань або призначенні комплексної терапії. Завдяки таблиці Interaction, система може автоматично виявляти потенційно небажані або небезпечні поєднання ліків, про що буде повідомлено користувачу чи фармацевту.

Таблиця 2.5 – Структура таблиці Interaction

Поле	Тип	Опис
interaction_id	PK, int	Унікальний ідентифікатор взаємодії
drug1_id	FK → Drug.drug_id	Перший препарат
drug2_id	FK → Drug.drug_id	Другий препарат
compatibility	enum (compatible, not_recommended, dangerous)	Рівень сумісності
comment	text	Коментар до взаємодії

2.4 Алгоритм пошуку лікарських засобів за критеріями

Розроблений алгоритм реалізує процес багатокритеріального пошуку лікарських засобів із залученням штучного інтелекту (AI), що виконує функцію динамічного уточнення запиту користувача. Процес побудований у вигляді послідовної блок-схеми, яка включає циклічний етап комунікації з AI-помічником.

Алгоритм (див. рис. 2.4) починається з блоку «Початок», після чого користувач вводить початкові критерії пошуку. До них можуть входити назва препарату, діюча речовина, форма випуску, країна виробник, побічні ефекти та інші параметри. Ці дані передаються до блоку формування запиту до бази даних, який готує SQL-запит на основі введених даних.

Далі система виконує фільтрацію записів у СКБД PostgreSQL, де відбираються лікарські засоби, що відповідають заданим критеріям. Якщо результатів багато, система передає їх у блок «AI-помічник задає уточнюючі питання». Це можуть бути запитання щодо віку пацієнта, тяжкості симптомів, бажаної форми випуску, супутніх захворювань, тощо.

Наступним етапом є формування списку препаратів, які аналізуються системою. AI проводить оцінку отриманих даних та звужує перелік засобів, формуючи більш релевантну вибірку.

Після цього спрацьовує умовний блок — перевірка, чи потрібно ще уточнення. Якщо відповідь «так», алгоритм повертається на етап діалогу з AI-помічником, що забезпечує циклічність ітерацій уточнення. Такий підхід дає змогу наблизити результат до потреб користувача без необхідності вручну переглядати велику кількість позицій.

Якщо уточнення більше не потрібне, система переходить до формування остаточного списку результатів, який включає назви препаратів, їхню діючу речовину, форму випуску, виробника та додаткову інформацію. На етапі виведення результатів користувач бачить перелік із можливістю перегляду повної інструкції та переходу до детального опису. Алгоритм завершується блоком «Кінець».

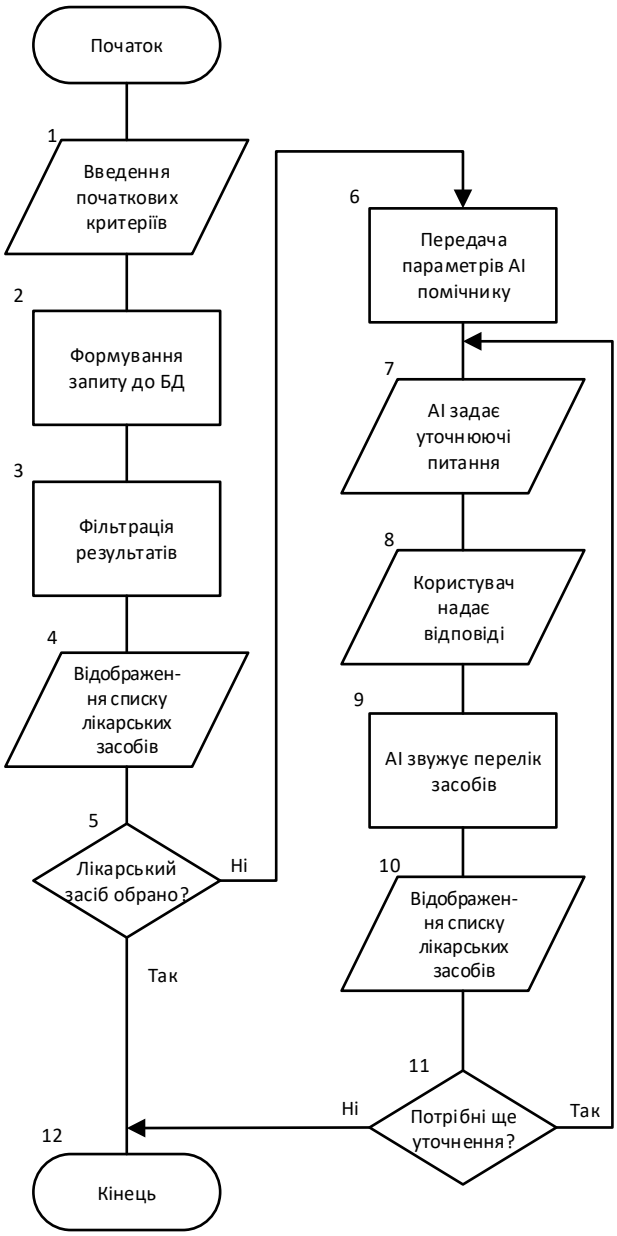

Рис. 2.4 – Блок-схема алгоритму багатокритеріального пошуку лікарських засобів

Такий підхід забезпечує:

- високу точність підбору засобів завдяки інтерактивній взаємодії;
- гнучкість алгоритму, що дозволяє враховувати індивідуальні особливості користувача;
- зменшення когнітивного навантаження на користувача за рахунок автоматичного фільтрування.

Можливість легко розширити логіку діалогу із застосуванням NLP-моделей AI.

У результаті реалізовано інтуїтивно зрозумілий, адаптивний алгоритм, який поєднує технології реляційних баз даних та штучного інтелекту в єдину систему підтримки прийняття рішень для фармацевтів та пацієнтів.

2.4 Розробка системи

У процесі реалізації програмного забезпечення системи підбору лікарських засобів основна частина клієнтської логіки була розроблена мовою програмування C# із використанням середовища розробки Visual Studio. Це дозволило ефективно поєднати надійність інструментів розробки для Windows-платформи із широким функціональним набором мови для побудови структурованих, об’єктно-орієнтованих рішень. Оскільки програмне забезпечення передбачає інтерактивну взаємодію з користувачем та зв’язок із зовнішніми сервісами, особливу увагу було приділено модульності коду, повторному використанню методів і відповідності принципам SOLID.

Одним із ключових елементів системи стала реалізація модуля взаємодії з AI-помічником, який відповідає за формування персоналізованих рекомендацій щодо вибору лікарських засобів. Для цього було розроблено окремий клас AIHelper, який інкапсулює всі необхідні функції для встановлення з’єднання з зовнішнім AI-сервером через REST API, обробки повідомлень користувача та отримання відповідей. Клас побудовано з використанням асинхронного доступу (async/await) для забезпечення стабільної та швидкої взаємодії без блокування інтерфейсу користувача(див. рис. 2.5).

```
public class AIHelper
{
    private readonly HttpClient _httpClient;
    private readonly string _apiEndpoint;
    private readonly string _apiKey;
    Ссылок: 0
    public AIHelper(string apiEndpoint, string apiKey)...

    // Метод: Надсилання повідомлення та отримання відповіді від AI
    Ссылок: 1
    public async Task<string> GetAIResponseAsync(string userMessage)
    {
        var requestBody = new
        {
            prompt = userMessage,
            max_tokens = 150
        };
        string json = JsonConvert.SerializeObject(requestBody);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        try
        {
            var response = await _httpClient.PostAsync(_apiEndpoint, content);
            response.EnsureSuccessStatusCode();

            string result = await response.Content.ReadAsStringAsync();

            // В залежності від API структури, адаптуй нижчий рядок:
            dynamic data = JsonConvert.DeserializeObject(result);
            string aiReply = data?.choices?[0]?.text ?? "Немає відповіді";

            return aiReply.Trim();
        }
        catch (Exception ex)
        {
            return $"Помилка підключення до AI: {ex.Message}";
        }
    }
```

Рис. 2.5 – Програмний код реалізації методів класу AIHelperdd

У класі реалізовано метод GetAIResponseAsync, який отримує текст запиту від користувача, формує структуру JSON-запиту до API (наприклад, OpenAI або власного сервера), відправляє його через HTTP-протокол та обробляє відповідь сервера у вигляді рядка. Отриманий текст аналізується і виводиться в інтерфейс у вигляді відповіді AI-помічника. Для симуляції діалогу в консолі або тестовому режимі було додатково реалізовано метод StartConversationAsync, що дає змогу будувати прості цикли запитів і відповідей у текстовому форматі. Код виклику методу наведено на рисунку 2.6

```
var ai = new AIHelper("https://api.openai.com/v1/completions", "api_key");
Ссылок: 0
await ai.StartConversationAsync();
```

Рис. 2.6 – Програмний виклик методу класу взаємодії з віддаленим AI сервісом

Завдяки інкапсуляції логіки роботи з AI в окремому класі було досягнуто високої гнучкості розробки — цей модуль можна легко адаптувати до будь-якої зовнішньої AI-платформи, змінивши лише параметри підключення та формат запиту. Це забезпечує розширюваність та відкриває перспективи для інтеграції з більш складними моделями рекомендацій у майбутньому.

Головна форма (див. рис. 2.7) програмного забезпечення "Система підбору лікарських засобів" є ключовим елементом взаємодії користувача з системою та виконує роль основного навігаційного вікна для доступу до всіх функціональних можливостей. Вона відкривається після авторизації користувача та адаптується до його ролі (звичайний користувач, фармацевт або адміністратор), відображаючи лише ті розділи, що відповідають рівню доступу.

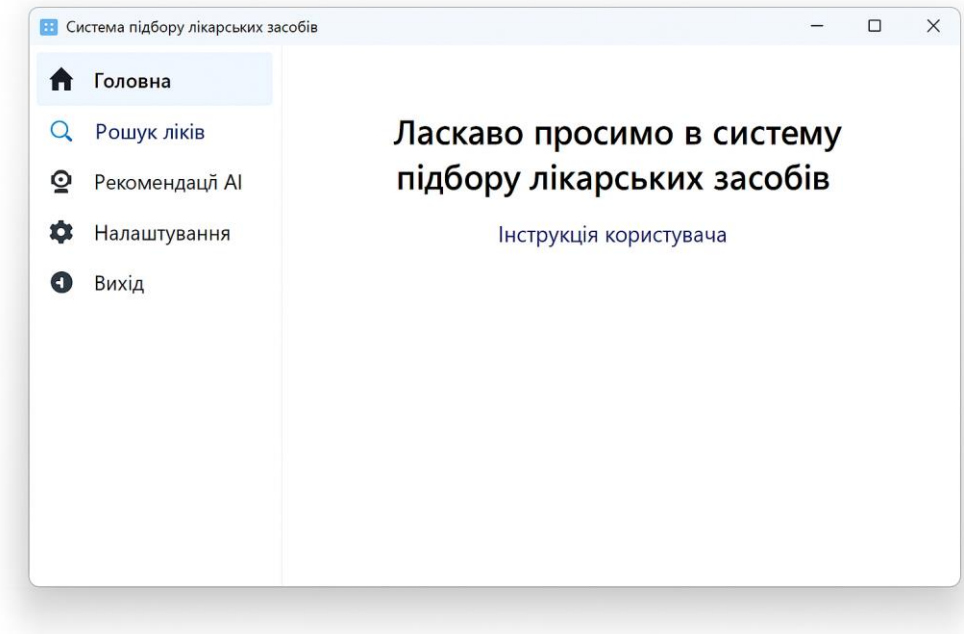

Рисунок 2.7 – Головна форма програми

Інтерфейс форми побудований за принципом класичного Windows-дизайну із використанням бокового меню (панелі навігації) зліва та динамічної області контенту праворуч. Ліва панель містить кнопки з графічними піктограмами та підписами українською мовою: "Головна", "Пошук ліків"(див. рис. 2.8), "AI-помічник"(див. рис. 2.9+), "Налаштування" (див. рис. 2.10) та "Вихід". Натискання на відповідну кнопку автоматично завантажує відповідну вкладку у центральну частину форми без відкриття нових вікон, що забезпечує швидку навігацію та зручність у користуванні.

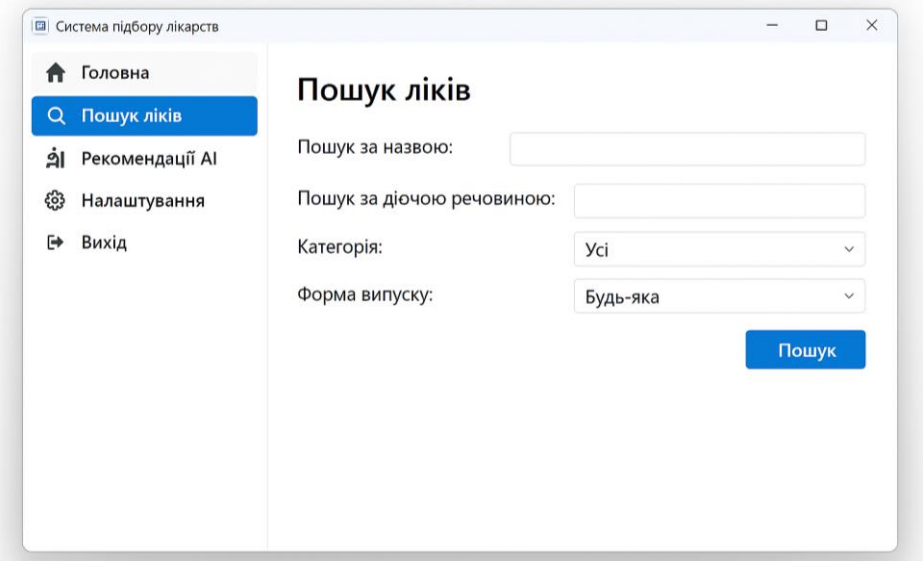

Рисунок 2.8 – Головна форма програми (вкладка «Пошук ліків»)

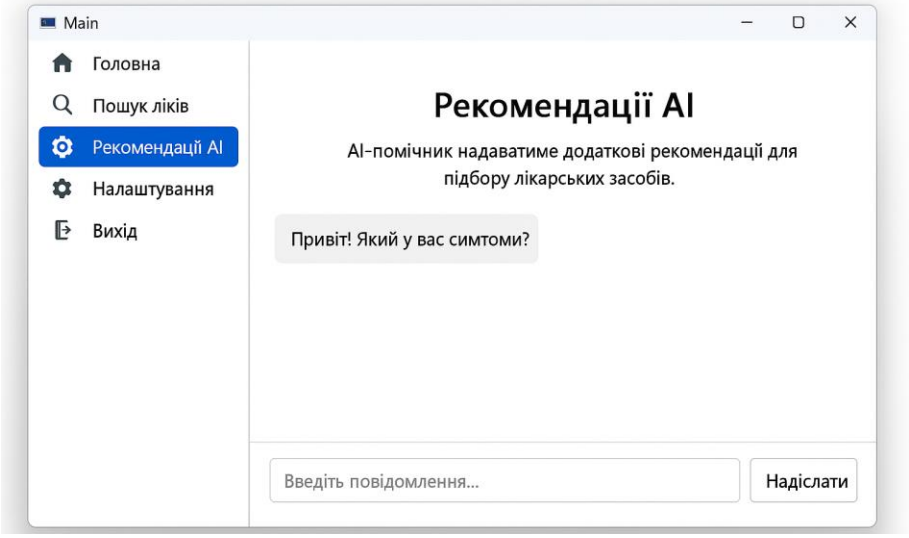

Рисунок 2.9 – Головна форма програми (вкладка «Рекомендації AI»)

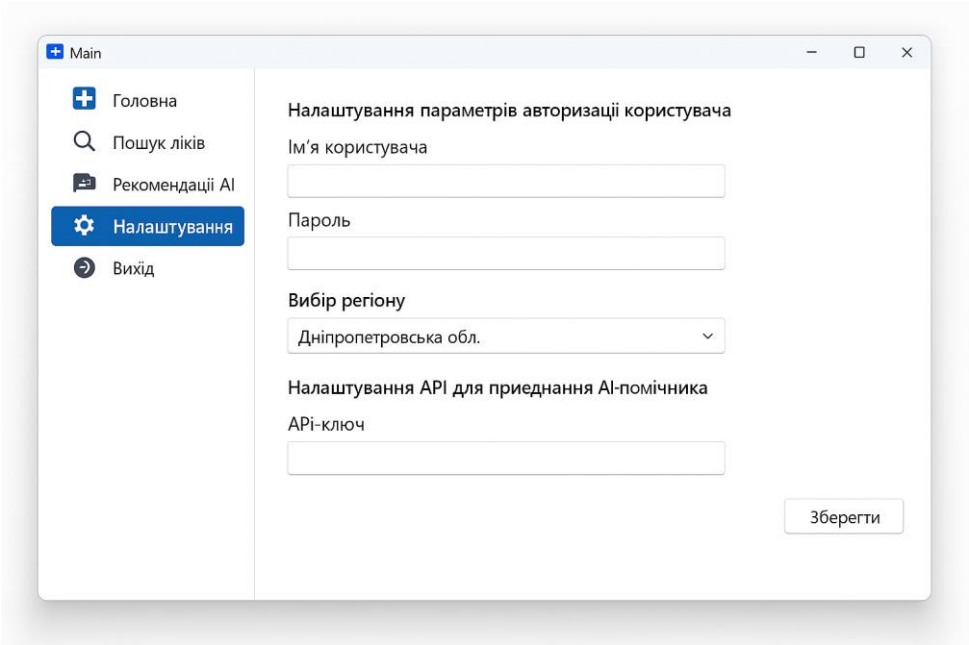

Рисунок 2.10 – Головна форма програми (вкладка «Налаштування»)

У верхній частині вікна розміщено заголовок із назвою системи, а також логотип, який надає інтерфейсу впізнаваності та підкреслює його офіційний характер. Основна область контенту, за замовчуванням, відкривається вкладкою "Головна", на якій розміщується привітальне повідомлення, а також коротка статистика про кількість зареєстрованих пацієнтів, виконаних підборів та виявлених попереджень щодо сумісності лікарських засобів. Тут також розташована кнопка "Пошук ліків", яка слугує швидким переходом до відповідної функції системи.

Форма реалізована з використанням мови програмування C# у середовищі Visual Studio з технологією Windows Forms. Вона є адаптивною — масштабування елементів інтерфейсу здійснюється автоматично відповідно до розміру вікна або роздільної здатності монітора. Кожен розділ, до якого здійснюється перехід, реалізований у вигляді окремого UserControl, що забезпечує модульність коду та спрощує супровід системи.

Таким чином, головна форма не лише виконує функцію навігаційного вузла між модулями, але й забезпечує зручність, стабільність та логічну організацію роботи користувача в межах всієї системи підбору лікарських засобів. Її дизайн відповідає сучасним вимогам до програм інженерії

програмного забезпечення, враховує принципи UI/UX, а реалізація відповідає умовам безперервної розширюваності й підтримки.

Форма «Авторизації» (LoginForm/Вхід) є однією з ключових складових користувацького інтерфейсу розробленої інформаційної системи підбору лікарських засобів. Її головним призначенням є забезпечення авторизованого доступу користувачів до функціональних можливостей системи з урахуванням їхньої ролі (звичайний користувач, фармацевт або адміністратор). За допомогою цієї форми (див. рис. 2.11) реалізується механізм аутентифікації, що дозволяє перевірити особу користувача та забезпечити належний рівень безпеки при роботі з даними системи.

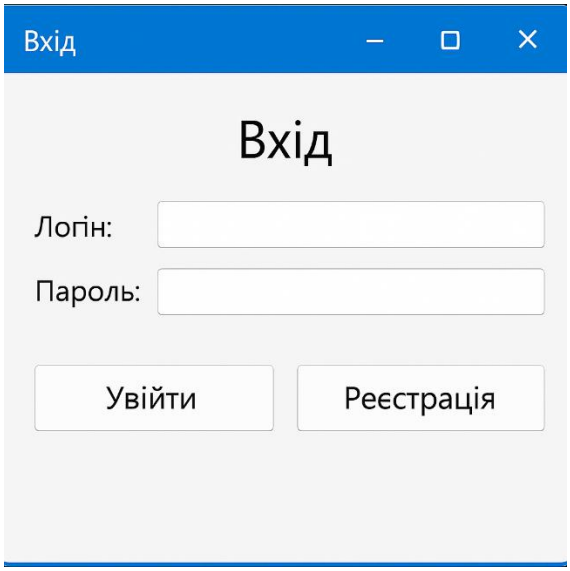

Рисунок 2.11 – Форма авторизації користувачів в системі

Форма має простий та інтуїтивно зрозумілий інтерфейс, виконаний у стилістиці Windows Forms. У верхній частині вікна розміщено заголовок «Вхід», який чітко вказує на призначення форми. Основна частина інтерфейсу містить два текстових поля — «Логін» та «Пароль», у які користувач вводить відповідні облікові дані. Для взаємодії передбачено дві кнопки: «Увійти», що ініціює процес перевірки введеної інформації та відкриття головного вікна, а також «Реєстрація», яка дозволяє новому користувачу перейти до форми створення облікового запису.

Ця форма є обов’язковим етапом входу до системи, оскільки дозволяє встановити індивідуальний профіль користувача, включно з регіональними налаштуваннями, які в подальшому впливають на відбір лікарських засобів. Залежно від успішності авторизації та ролі, користувач буде перенаправлений до відповідної головної форми (користувача, фармацевта або адміністратора), що реалізує відповідні можливості в межах системи.

2.5 Тестування розробленої системи

Після завершення етапів проектування та реалізації програмного забезпечення необхідним кроком є тестування системи. Цей процес забезпечує впевненість у її працездатності, функціональній повноті, стабільності та відповідності вимогам поставлених завдань. Тестування є критичним етапом у життєвому циклі програмного продукту, оскільки дозволяє виявити дефекти на ранніх стадіях та гарантує якість, надійність і зручність користування програмою.

Мета тестування — верифікація і валідація всіх функціональних модулів системи, перевірка відповідності вимогам, виявлення потенційних збоїв, помилок і логічних недоліків. Зокрема, важливо було протестувати не лише стандартні сценарії взаємодії, а й граничні умови, помилки введення, реакцію на некоректні дані та стійкість програми до навантаження.

Тестування відбувалося поетапно — за ключовими функціональними модулями: авторизація, навігація, пошук лікарських засобів, взаємодія з AI-помічником, перегляд результатів, робота адміністратора. Кожен із цих модулів мав окремі сценарії тестування (тест-кейси), які виконувалися в середовищі Microsoft Visual Studio з використанням інтерактивного інтерфейсу WinForms. Також перевірялася взаємодія з базою даних СКБД PostgreSQL та мережевими API.

Першим кроком стало тестування модуля входу в систему. Авторизація є критично важливою для контролю доступу та визначення ролі користувача. Основна логіка полягала у введенні логіна й пароля (див. рис. 2.12), перевірці

наявності такого користувача в базі та завантаженні відповідної головної форми.

Було перевірено кілька сценаріїв:

- успішний вхід для користувача з роллю «Користувач»;
- вхід під обліковим записом «Адміністратор»;
- помилковий логін (неіснуючий користувач);
- неправильний пароль;
- порожні поля.

Усі сценарії відпрацювали коректно: система відображала відповідні повідомлення про помилки, блокувала вхід за хибними даними та виконувала перенаправлення при успішній авторизації. Перевірялося також відображення повідомлень українською мовою та збереження сесії після входу.

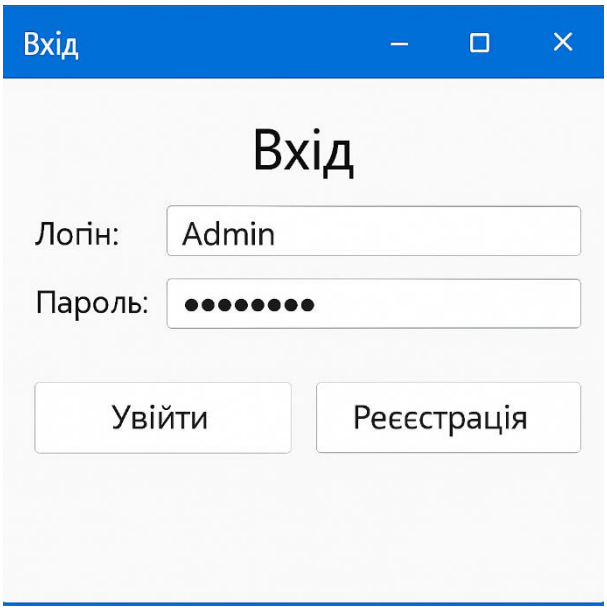

Рисунок 2.12 – Графічний інтерфейс форми авторизації під час тестування

Наступним етапом стало тестування головної форми, яка є навігаційною точкою системи. Основна мета — перевірити доступність і зміну вкладок, динамічне відображення елементів залежно від ролі користувача та правильність стилізації інтерфейсу.

Користувачам різних ролей відображаються різні кнопки меню (наприклад, адміністратору доступна вкладка "Адміністрування"). При натисканні на кожну кнопку — динамічно оновлювався вміст панелі, без відкриття нових вікон. Було перевірено:

- відображення активної вкладки синім кольором;
- збереження розмітки при зміні розміру вікна;
- робота навігаційних кнопок «Пошук ліків», «Помічник AI», «Налаштування» та «Вихід»;
- Перехід до авторизації користувача з кнопки у верхньому правомі куті.

Центральною функцією системи є можливість здійснення пошуку лікарських засобів за вказаними критеріями підбору. Було перевірено:

- роботу полів пошуку (див. рис. 2.13);
- фільтрацію за країною виробника, формою випуску, ціною;
- обробку запитів без результату;
- коректне формування таблиці результатів ;
- перехід до перегляду детальної інформації (див. рис. 2.14).

Тестування показало, що система правильно відображає дані, формує запроси та інтерпретує інформацію з бази даних, працює швидко та стабільно. Також було перевірено реакцію системи на великі обсяги запитів.

Тестування модуля AI-помічника, що є унікальною особливістю цієї системи. Він реалізує базову експертну логіку, яка дозволяє уточнювати симптоми користувача, формувати рекомендації, задавати уточнюючі запитання (див. рис. 2.15). Було протестовано:

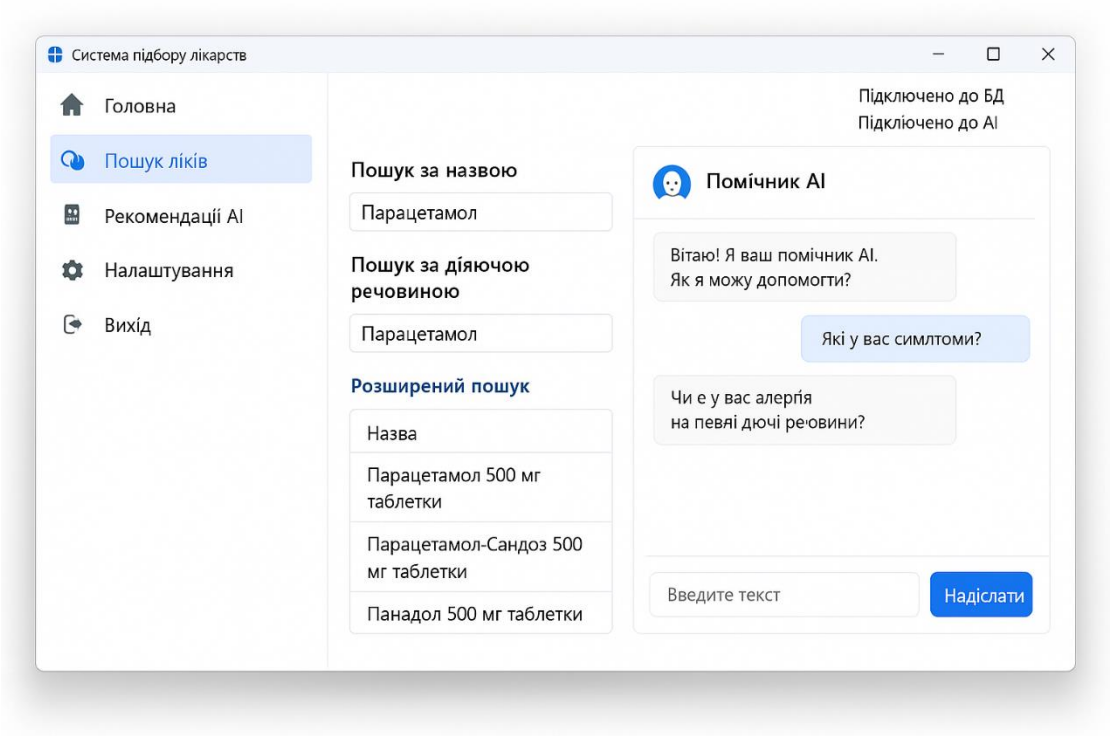

Рисунок 2.13 – Графічний інтерфейс головної форми вкладка «пошук ліків»

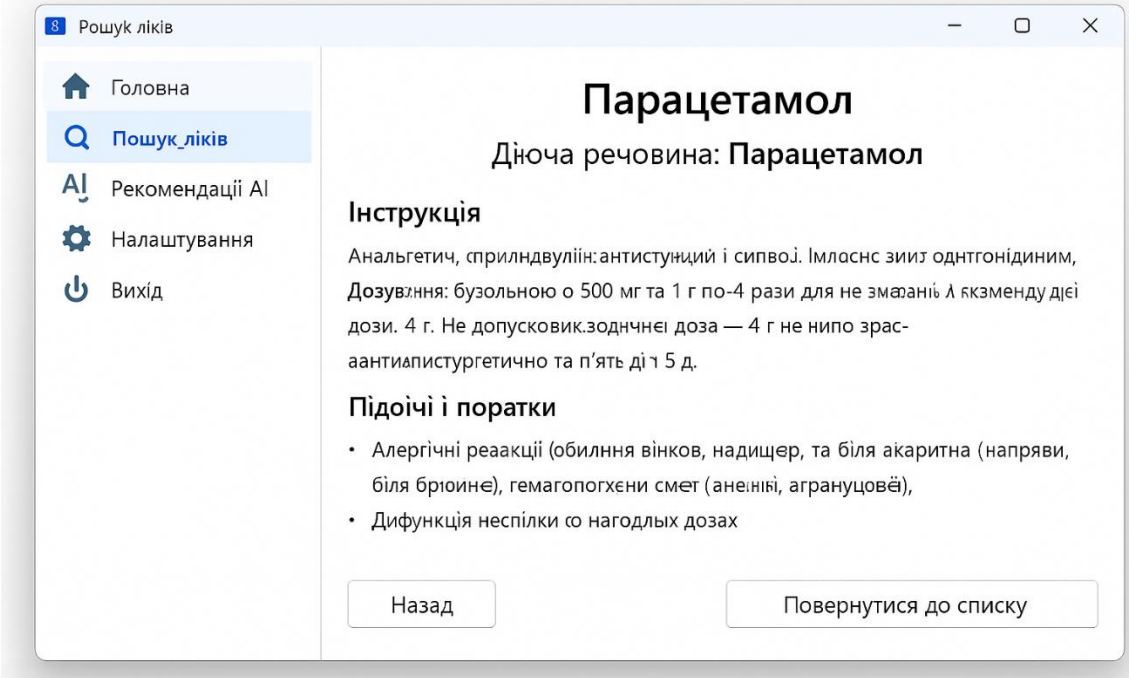

Рисунок 2.14 – Графічний інтерфейс головної форми при перегляді детальної інформації

- відправлення повідомлення до AI;
- отримання відповіді;
- логіка уточнення;
- поведінка у разі невалідного введення;
- робота в умовах повільного інтернет-з'єднання.

Для тестування модуля інтерактивного AI-помічника було використано емуляційний API, який моделює поведінку мовної моделі, натренованої на корпусі знань з лікарських засобів. Це дозволило перевірити логіку формування запитів, коректність обробки введених симптомів користувачем, а також відображення діалогового вікна з відповідями системи. У подальшому передбачено підключення до реального AI-сервісу, наприклад OpenAI, через REST API з передачею текстового контексту.

Результати тестування показали, що система коректно працює з API: передає вхідні дані, отримує логічні відповіді, забезпечує діалоговий режим спілкування. Повідомлення AI формуються на основі знань про лікарські засоби.

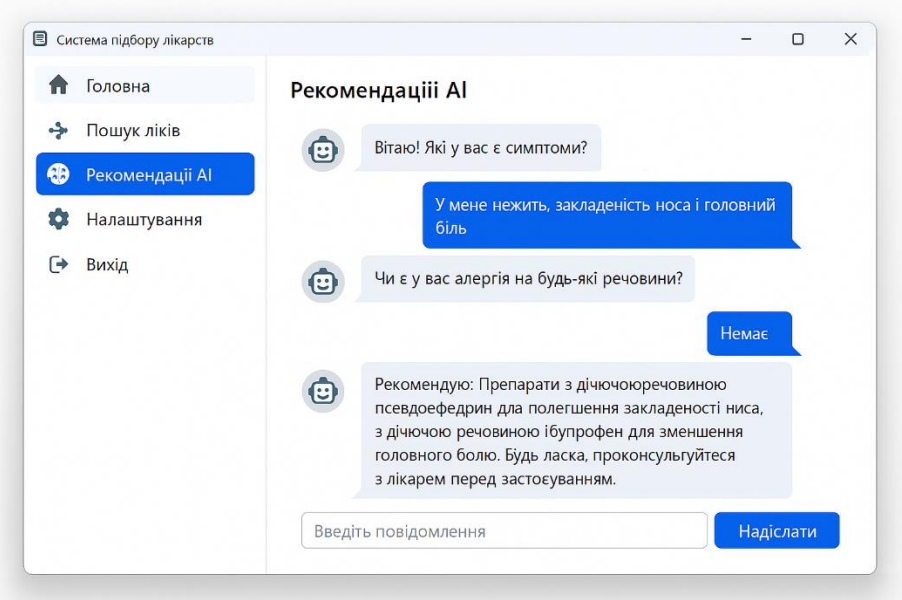

Рисунок 2.15 – Графічний інтерфейс головної форми при перегляді чату з AI помычником

Останнім було протестовано адміністративний інтерфейс. Цей модуль дозволяє адміністратору:

- переглядати список користувачів (див. рис. 2.16);
- додавати нові облікові записи;
- призначати або змінювати ролі;
- видаляти користувачів;
- працювати з базою лікарських засобів (рис. 2.17).

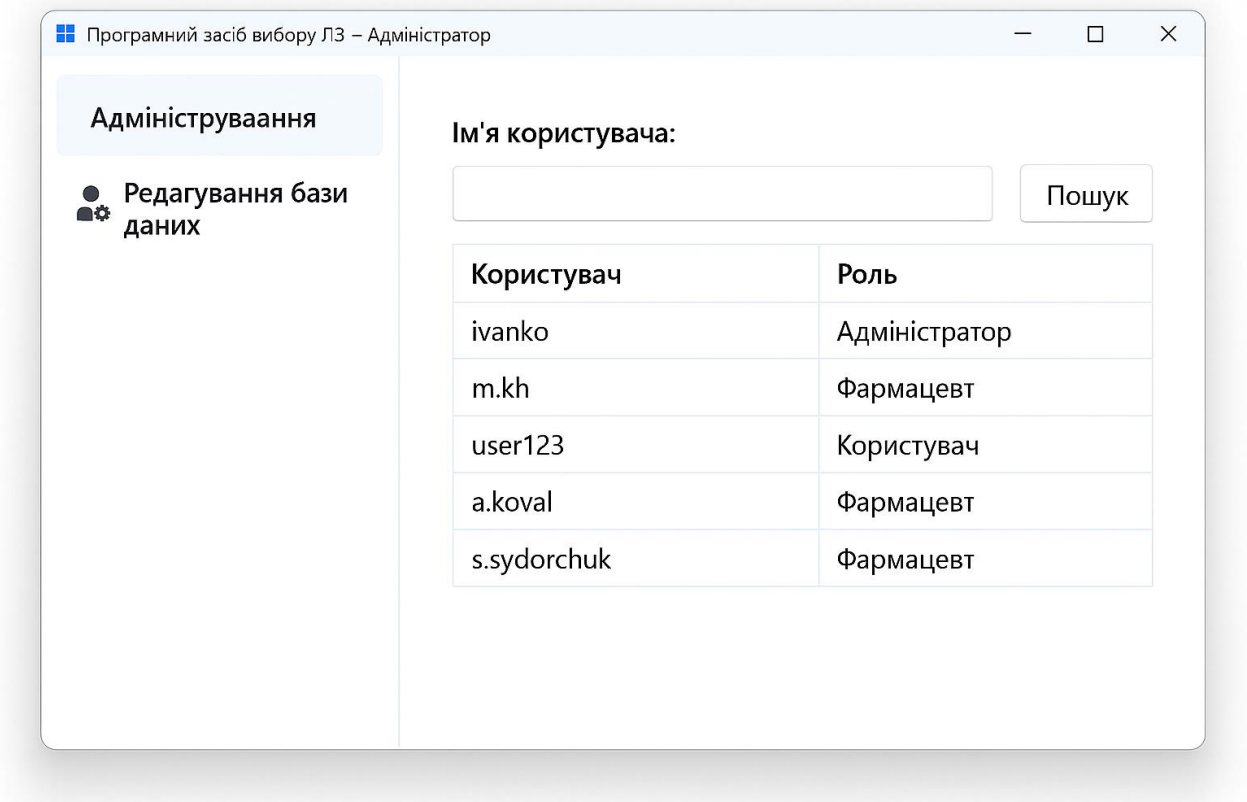

Рисунок 2.16 – Графічний інтерфейс головної форми в режимі перегляду списку користувачів.

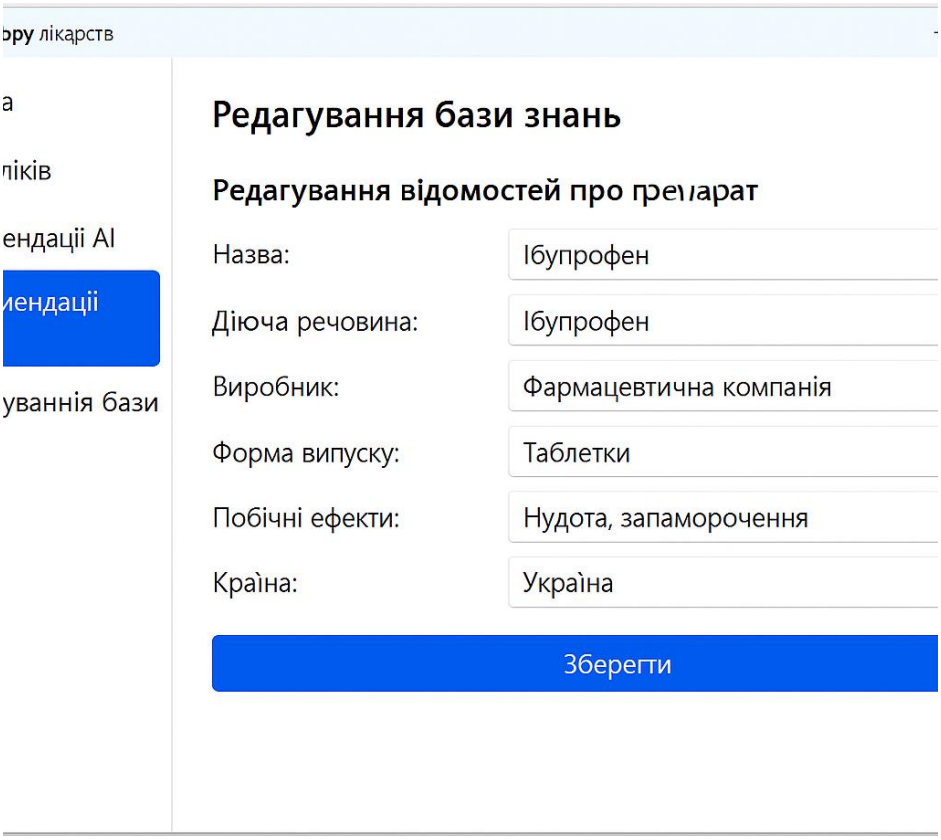

Рисунок 2.17 – Фрагмент інтерфейсу системи в режимі редагування інформації про лікарські засоби у БД.

Особливу увагу приділено стабільності обробки запитів, що надходять з клієнтського застосунку до серверної частини через REST API. Було протестовано обробку ситуацій з невалідними даними, помилками автентифікації, відсутністю підключення до сервера бази даних, а також випадки взаємної несумісності ліків, які мають бути оброблені системою з відповідним повідомленням користувачу. Усі запити до бази PostgreSQL виконувалися без помітних затримок, з максимальною швидкістю реакції на дії користувача, що свідчить про ефективність побудованої структури БД та оптимізацію SQL-запитів.

Під час перевірки багатокористувацької взаємодії було змодельовано одночасний доступ до бази даних з кількох клієнтських форм, при цьому не зафіксовано конфліктів, втрати даних або зависання застосунку. Результати змін у базі відображались у режимі реального часу, що свідчить про правильну реалізацію транзакційної моделі та конкурентного доступу.

Таким чином, за результатами тестування підтверджено стабільність роботи системи в різних режимах, її відповідність функціональним вимогам, зручність використання та ефективну обробку запитів до бази даних. Отримані результати свідчать про готовність програмного продукту до впровадження та подальшого масштабування.

Висновки за розділом

Розділ присвячено детальному огляду повного циклу створення інформаційної системи підбору лікарських засобів — від початкового проектування архітектури до фінального тестування працездатності. Цей етап роботи відіграв ключову роль у практичному втіленні задуманої функціональності та реалізації вимог, визначених на попередньому етапі аналізу предметної області.

Першочергово було здійснено проектування архітектури програмного продукту, де обґрунтовано вибір клієнт-серверної моделі з розподілом системи на клієнтський інтерфейс, серверну логіку та інтеграцію штучного інтелекту як елементу системи через API. Такий підхід забезпечує масштабованість, модульність та можливість інтеграції з іншими медичними інформаційними системами. Особливу увагу приділено багаторівневій організації системи, яка дозволяє одночасне підключення багатьох клієнтів через мережу Інтернет.

У підрозділі, присвяченому обґрунтуванню вибору системи керування базами даних, було розглянуто переваги використання PostgreSQL як основної СКБД для даної системи. PostgreSQL було обрано завдяки її надійності, відповідності вимогам до реляційної структури даних, підтримці складних запитів, безпеки, а також можливості роботи у хмарному середовищі для забезпечення багатокористувацького доступу.

Подальшим кроком стало проектування логічної структури бази даних, у межах якого сформовано сутності, атрибути та зв’язки, що охоплюють інформацію про препарати, діючі речовини, регіони, користувачів, взаємодії

між ліками тощо. Створена модель бази забезпечує оптимальне зберігання даних і дозволяє здійснювати гнучкі запити для реалізації функцій пошуку та фільтрації.

У процесі розробки алгоритмів пошуку лікарських засобів були враховані такі критерії, як назва препарату, діюча речовина, форма випуску, країна виробник, середня ціна та побічні ефекти. Реалізовано механізм перевірки сумісності декількох лікарських засобів, що базується на попередньо заданих правилах у таблиці взаємодій. Алгоритм забезпечує точний і ефективний відбір медичних засобів у відповідності до запиту користувача.

У межах розробки системи створено графічний інтерфейс користувача на мові C# з використанням технології Windows Forms. Реалізовано розмежування доступу за ролями (користувач, фармацевт, адміністратор), діалогове вікно AI-помічника, динамічну взаємодію з серверною частиною через REST API. Візуальний інтерфейс спроєктовано таким чином, щоб забезпечити інтуїтивну навігацію, адаптивність до різних розмірів екрана та відповідність сучасним вимогам до UI/UX дизайну.

Під час тестування системи перевірено коректність функціонування ключових модулів: авторизації, пошуку, перевірки сумісності, інтеграції з модулем рекомендацій, а також стабільність під час одночасного доступу з кількох клієнтських додатків. Виявлені під час тестування недоліки були виправлені на рівні бізнес-логіки та структури запитів до бази даних. У результаті тестування підтверджено відповідність розробленого програмного забезпечення технічному завданню та поставленим функціональним вимогам.

Таким чином, результати, отримані на етапі проєктування, розробки та тестування, засвідчують, що створена система є функціонально повноцінною, надійною та готовою до практичного впровадження в умовах реального використання в фармацевтичній або медичній галузі.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи була розроблена інформаційна система для підбору лікарських засобів, що поєднує сучасні підходи до медичного програмного забезпечення, бази даних та технології штучного інтелекту. Проєкт орієнтований на вирішення актуальної проблеми — автоматизації процесу пошуку ефективних і сумісних лікарських засобів із урахуванням побажань користувача, клінічних параметрів, побічних ефектів, діючих речовин і регіональної доступності.

Розроблене програмне забезпечення реалізовано у вигляді настільного клієнтського застосунку на мові програмування C# із використанням технологій Windows Forms, що забезпечує простоту інтерфейсу та високу швидкість розгортання. Інтерфейс системи побудований з дотриманням принципів юзабіліті: всі функціональні можливості згруповані у вкладки, навігація здійснюється через бокове меню, що адаптується залежно від ролі користувача.

Особливістю системи є модуль інтеграції з багатокористувацькою реляційною базою даних, реалізованою на основі СКБД PostgreSQL. Це дозволяє забезпечити збереження, фільтрацію та оновлення великого обсягу інформації про лікарські засоби, облікові записи користувачів та їх взаємодію з системою. Створена структура бази охоплює всі ключові сутності: користувачі, препарати, діючі речовини, форми випуску, країну виробника та історію підборів.

Ключовим нововведенням проєкту стало впровадження інтерактивного AI-помічника, який, на основі навченої моделі, здатен уточнювати запити користувача, інтерпретувати симптоми та надавати рекомендації. Цей компонент реалізується через зовнішній API, що дозволяє в подальшому масштабувати інтелектуальні функції системи без втручання в клієнтську частину.

У межах реалізації було також створено адміністративний інтерфейс, що дозволяє керувати обліковими записами, контролювати права доступу, редагувати базу лікарських засобів та здійснювати моніторинг дій користувачів. Це дає змогу ефективно підтримувати та супроводжувати систему в умовах реального використання.

У процесі тестування всі компоненти програмного забезпечення пройшли перевірку на функціональність, зручність, стійкість до помилкових введень та відповідність поставленим завданням. Жодних критичних дефектів виявлено не було. Система успішно виконує пошук, фільтрацію, верифікацію взаємодій між ліками та надає інтелектуальні поради.

Результати кваліфікаційної роботи засвідчили, що створена система повністю відповідає поставленим вимогам, є технічно та логічно завершеною, а також має потенціал до масштабування та інтеграції з іншими медичними інформаційними сервісами. Проєкт демонструє приклад поєднання класичних методів розробки програмного забезпечення з сучасними інтелектуальними рішеннями у сфері охорони здоров’я.

Таким чином, виконання цієї кваліфікаційної роботи дозволило на практиці реалізувати повноцінний життєвий цикл розробки програмного забезпечення — від формування вимог і проєктування бази даних до програмування, тестування та візуалізації інтерфейсу. Система підбору лікарських засобів може бути застосована як у навчальних цілях, так і слугувати основою для впровадження у мережах фармацевтичних та медичних закладів.

СПИСОК ПОСИЛАНЬ

1. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE).

2. Коваленко В. В., Сидоренко О. В. Інженерія програмного забезпечення: навч. посібник. — Київ: Ліра-К, 2021.

3. Барабаш О. І., Кушніренко С. В. Проектування інформаційних систем: UML та аналіз вимог. — Харків: ХНУРЕ, 2020.

4. Sommerville I. Software Engineering. 10th ed. — Pearson Education, 2015.

5. Weller D. Beginning C# Programming with .NET. — Wiley, 2018.

6. RxNorm API. National Library of Medicine. URL: https://www.nlm.nih.gov/research/umls/rxnorm/

7. Drugs.com Professional. URL: https://www.drugs.com/pro/

8. DailyMed. U.S. National Library of Medicine. URL: https://dailymed.nlm.nih.gov/dailymed/

9. PubChem Database. National Center for Biotechnology Information. URL: https://pubchem.ncbi.nlm.nih.gov/

10. OpenFDA API. U.S. Food & Drug Administration. URL: https://open.fda.gov/

11. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

12. Kreibich J. Using PostgreSQL. — O'Reilly Media, 2010.

13. Obe R. O., Hsu L. S. PostgreSQL: Up and Running. 3rd Edition. — O'Reilly, 2017.

14. Microsoft Docs. C# Windows Forms Overview. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/

15. Troelsen A., Japikse P. Pro C# 8 with .NET Core 3. 9th ed. — Apress, 2020.

16. OpenAI API Reference. URL: https://platform.openai.com/docs

17. Chollet F. Deep Learning with Python. — Manning Publications, 2018.

18. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. — Pearson, 2020.

19. Goodfellow I., Bengio Y., Courville A. Deep Learning. — MIT Press, 2016.

20. Національний перелік основних лікарських засобів України. МОЗ України. URL: https://moz.gov.ua

Додаток А – Код програми

```
// Program.cs
using System;
using System.Windows.Forms;

namespace MedSelectionSystem
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}

// MainForm.cs
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

namespace MedSelectionSystem
{
    public partial class MainForm : Form
    {
        private Panel navigationPanel;
        private Panel contentPanel;
        private Button btnHome, btnSearch, btnCompatibility,
btnAI, btnHistory, btnSettings, btnExit;
        private PictureBox logoBox;
        private Label lblTitle;
        private Dictionary<string, UserControl> tabs;

        public MainForm()
        {
            InitializeComponent();
            InitializeUI();
            InitializeTabs();
        }

        private void InitializeUI()
        {
            // Ініціалізація навігаційної панелі та кнопок
        }

        private void InitializeTabs()
        {
```

```
                tabs = new Dictionary<string, UserControl>
                {
                    { "Home", new Controls.HomeControl() },
                    { "Search", new Controls.SearchControl() },
                    {            "Compatibility",            new
Controls.InteractionControl() },
                    { "AI", new Controls.AIControl() },
                    { "History", new Controls.HistoryControl() },
                    { "Settings", new Controls.SettingsControl()
}
                };
                LoadTab("Home");
            }

            private void LoadTab(string key)
            {
                contentPanel.Controls.Clear();
                if (tabs.ContainsKey(key))
                    contentPanel.Controls.Add(tabs[key]);
            }

            private  Button  CreateNavButton(string  text,  int
index)
            {
                var btn = new Button
                {
                    Text = text,
                    Height = 50,
                    Width = 200,
                    FlatStyle = FlatStyle.Flat,
                    ForeColor = Color.White,
                    Location = new Point(0, 50 * index)
                };
                btn.Click += (s, e) => LoadTab(text);
                return btn;
            }
        }
    }

    // AIHelper.cs
    using System;
    using System.Net.Http;
    using System.Net.Http.Headers;
    using System.Text;
    using System.Threading.Tasks;
    using Newtonsoft.Json;

    namespace MedSelectionSystem
    {
        public class AIHelper
        {
            private readonly HttpClient httpClient;
            private readonly string apiEndpoint;
```

```
            public AIHelper(string endpoint, string apiKey)
            {
                apiEndpoint = endpoint;
                httpClient = new HttpClient();
                httpClient.DefaultRequestHeaders.Authorization =
new AuthenticationHeaderValue("Bearer", apiKey);
            }

            public async Task<string> GetAIResponseAsync(string
userMessage)
            {
                var requestBody = new { prompt = userMessage,
max_tokens = 150, temperature = 0.7 };
                var content = new
StringContent(JsonConvert.SerializeObject(requestBody),
Encoding.UTF8, "application/json");
                var response = await
httpClient.PostAsync(apiEndpoint, content);
                response.EnsureSuccessStatusCode();
                dynamic data =
JsonConvert.DeserializeObject(await
response.Content.ReadAsStringAsync());
                return (string)(data?.choices?[0]?.text ?? "");
            }
        }
    }

    // Controls/HomeControl.cs
    using System.Windows.Forms;

    namespace MedSelectionSystem.Controls
    {
        public partial class HomeControl : UserControl
        {
            private Label lblWelcome;
            private LinkLabel linkInstructions;

            public HomeControl()
            {
                InitializeComponent();
                // Ініціалізація вітального блоку
            }
        }
    }

    // Controls/SearchControl.cs
    using System;
    using System.Windows.Forms;

    namespace MedSelectionSystem.Controls
    {
        public partial class SearchControl : UserControl
```

```
        {
            private TextBox txtName;
            private TextBox txtSubstance;
            private ComboBox cmbForm;
            private ComboBox cmbRegion;
            private NumericUpDown nudPriceMin;
            private NumericUpDown nudPriceMax;
            private Button btnFind;

            public SearchControl()
            {
                InitializeComponent();
                // Ініціалізація елементів пошуку
            }

            private void btnFind_Click(object sender, EventArgs
e)
            {
                // Обробка пошукового запиту
            }
        }
    }

    // Controls/InteractionControl.cs
    using System;
    using System.Windows.Forms;

    namespace MedSelectionSystem.Controls
    {
        public partial class InteractionControl : UserControl
        {
            private ComboBox cmbDrug1;
            private ComboBox cmbDrug2;
            private Button btnCheck;
            private Label lblResult;

            public InteractionControl()
            {
                InitializeComponent();
                // Ініціалізація перевірки сумісності
            }

            private void btnCheck_Click(object sender, EventArgs
e)
            {
                // Логіка перевірки взаємодії
            }
        }
    }

    // Controls/AIControl.cs
    using System;
    using System.Windows.Forms;
```

```
using System.Threading.Tasks;

namespace MedSelectionSystem.Controls
{
    public partial class AIControl : UserControl
    {
        private TextBox txtChat;
        private TextBox txtInput;
        private Button btnSend;
        private AIHelper aiHelper;

        public AIControl()
        {
            InitializeComponent();
            aiHelper = new AIHelper("<endpoint>", "<key>");
            // Ініціалізація інтерфейсу чат-бота
        }

        private async void btnSend_Click(object sender,
EventArgs e)
        {
            string userMsg = txtInput.Text;
            string response = await
aiHelper.GetAIResponseAsync(userMsg);
            txtChat.AppendText("Користувач: " + userMsg +
"\nAI: " + response + "\n");
        }
    }

    // Controls/HistoryControl.cs
    using System.Windows.Forms;

    namespace MedSelectionSystem.Controls
    {
        public partial class HistoryControl : UserControl
        {
            private DataGridView dgvHistory;

            public HistoryControl()
            {
                InitializeComponent();
                // Завантаження історії пошуків
            }
        }
    }

    // Controls/SettingsControl.cs
    using System;
    using System.Windows.Forms;

    namespace MedSelectionSystem.Controls
    {
```

```
        public partial class SettingsControl : UserControl
        {
            private ComboBox cmbRegion;
            private Button btnSave;

            public SettingsControl()
            {
                InitializeComponent();
                // Ініціалізація налаштувань
            }

            private void btnSave_Click(object sender, EventArgs
e)
            {
                // Збереження налаштувань користувача
            }
        }
    }


    using System;
    using System.Collections.Generic;
    using System.Net.Http;
    using System.Net.Http.Headers;
    using System.Text;
    using System.Text.Json;
    using System.Threading;
    using System.Threading.Tasks;

    namespace MedSelectionSystem.AI
    {
        /// <summary>
        /// Клас для взаємодії з зовнішнім AI-помічником через
REST API.
        ///  Інкапсулює підключення, відправку повідомлень та
отримання відповідей.
        /// </summary>
        public class AIClient : IDisposable
        {
            // HTTP-клієнт для відправки запитів
            private readonly HttpClient _httpClient;

            // URL кінцевої точки AI-сервісу
            private readonly Uri _endpoint;

            // Ключ доступу до API
            private readonly string _apiKey;

            // Ідентифікатор сеансу (можна використовувати для
ведення діалогу)
            private string _sessionId;

            // Історія повідомлень у поточному сеансі
```

```
        private readonly List<ChatMessage> _history = new
List<ChatMessage>();

        /// <summary>
        /// Ініціалізує новий екземпляр AIClient.
        /// </summary>
        /// <param name="endpoint">URL API-сервісу</param>
        /// <param name="apiKey">Ключ доступу</param>
        public AIClient(string endpoint, string apiKey)
        {
            _endpoint = new Uri(endpoint);
            _apiKey    =    apiKey    ??    throw    new
ArgumentNullException(nameof(apiKey));

            _httpClient = new HttpClient();
            _httpClient.DefaultRequestHeaders.Authorization
=
                new      AuthenticationHeaderValue("Bearer",
_apiKey);
            _httpClient.Timeout = TimeSpan.FromSeconds(30);

            // Згенеруємо унікальний sessionId
            _sessionId = Guid.NewGuid().ToString();
        }

        /// <summary>
        /// Відправляє повідомлення користувача і повертає
відповідь AI.
        /// </summary>
        ///   <param   name="userText">Текст   запиту   від
користувача</param>
        ///      <param      name="cancellationToken">Токен
скасування</param>
        public  async  Task<string>  SendMessageAsync(string
userText, CancellationToken cancellationToken = default)
        {
            if (string.IsNullOrWhiteSpace(userText))
                throw new ArgumentException("Повідомлення не
може бути пустим.", nameof(userText));

            // Додати повідомлення користувача до історії
            _history.Add(new  ChatMessage  {  Role  =  "user",
Content = userText });

            // Побудова тіла запиту
            var payload = new
            {
                session_id = _sessionId,
                messages = _history,
                user = new { id = _sessionId }
            };
            string               jsonPayload              =
JsonSerializer.Serialize(payload);
```

```
                using var content = new
StringContent(jsonPayload, Encoding.UTF8, "application/json");
                using var response = await
_httpClient.PostAsync(_endpoint, content, cancellationToken);
                response.EnsureSuccessStatusCode();

                string jsonResponse = await
response.Content.ReadAsStringAsync(cancellationToken);
                var aiReply = ParseAIReply(jsonResponse);

                // Додати відповідь AI до історії
                _history.Add(new ChatMessage { Role =
"assistant", Content = aiReply });

                return aiReply;
            }

            /// <summary>
            /// Парсить JSON-відповідь від сервісу і повертає
текст відповіді AI.
            /// </summary>
            private string ParseAIReply(string jsonResponse)
            {
                try
                {
                    using var doc =
JsonDocument.Parse(jsonResponse);
                    // Приклад шляху до тексту:
data.choice[0].message.content
                    var content = doc.RootElement
                                     .GetProperty("choices")[0]
                                     .GetProperty("message")
                                     .GetProperty("content")
                                     .GetString();
                    return content ?? string.Empty;
                }
                catch (Exception ex)
                {
                    // У разі помилки парсингу повертаємо текст
помилки
                    return $"[Помилка розбору відповіді AI:
{ex.Message}]";
                }
            }

            /// <summary>
            /// Очищує історію поточного сеансу.
            /// </summary>
            public void ResetSession()
            {
                _history.Clear();
                _sessionId = Guid.NewGuid().ToString();
```

```
        }

        /// <summary>
        /// Звільняє ресурси.
        /// </summary>
        public void Dispose()
        {
            _httpClient.Dispose();
        }

        /// <summary>
        /// Внутрішня модель повідомлення для обміну з AI.
        /// </summary>
        private class ChatMessage
        {
            public string Role { get; set; }      // "user" або "assistant"
            public string Content { get; set; }   // Текст повідомлення
        }
    }
}
```