[НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва кафедри)

«**До захисту допущено**»
В.о. Завідувачки кафедри
Біомедичної кібернетики

_____ Світлана АЛХІМОВА
(підпис) (ініціали, ПРІЗВИЩЕ

“___” __*червня*__2025р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп’ютерні технології в біології та медицині»
зі спеціальності *122 «Комп’ютерні науки»*

на тему: **Програмний застосунок сімейного лікаря для управління медичними записами в приватній клініці приміського типу**

Виконав: студент **IV** курсу, групи _БС-15__

Царик Олександр Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник: *Гладка Мирослава Вікторівна, старший викладач, к.т.н.* _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по ініціали) (підпис)

Консультант з розділів дипломної роботи: _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент: *Сичик Марина Михайлівна, доцент кафедри біомедичної Інженерії, к.т.н., доцент* _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»
(код і назва)

ЗАТВЕРДЖУЮ
В.о. Завідувачки кафедри БМК
_____ Світлана АЛХІМОВА
«___»__*квітень*___2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

Царику Олександру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Програмний застосунок сімейного лікаря для управління медичними записами в приватній клініці приміського типу»

керівник роботи
Гладка Мирослава Вікторівна, старший викладач, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26»_*травня*_ 2025 р. №_1758-с

2. Термін подання студентом роботи ***22-23 травня 2025 року***
3. Вихідні дані *реальні медичні записи отримано від приватної клініки.*
4. Зміст роботи Проаналізувати наявні на базі практики літературу та існуючі рішення про управління медичними записами. Проаналізувати та обрати засоби реалізації програмного застосунку, програмний продукт та засоби захисту інформації. Створити та протестувати програмний застосунок. Провести розрахунок економічного ефекту за темою ДР. Зробити загальні висновки та визначитись з остаточною темою ДР до наказу
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) *[Презентація на 15 слайдах, 14 рисунків і 2 таблиць]*
6. Консультанти розділів роботи*

□ □□□□□□□□□□□□ Якщо визначені консультанти. Консультантом не може бути зазначено наукового керівника магістерської дисертації.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата завдання	
		видав	прийняв

7. Дата видачі завдання ***17 березня 2025 р.***.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 17.03.2025	***виконано***
2	Переддипломна практика. Виконання практичної частини ДР та додаткових розділів	1-4 тиждень за графіком практики	***виконано***
3	Виконання основних розділів ДР (Аналіз аналогів програмного забезпечення, засоби реалізації програмного продукту, обґрунтування вибору програмного середовища розробки, програмна реалізація та методика використання програмного продукту, узагальнення результатів дослідження)	Протягом усієї практики	***виконано***
4	Перевірка ДР науковим керівником	до 20.05.2025	***виконано***
5	Подання в електронному вигляді ДР на перевірку нормоконтролера та плагіат (StrikePlagiarism.com). Доопрацювання ДР	Не пізніше 23.05.2025	***виконано***
6	Отримати відгук від керівника ДР	Не пізніше 05.06.2025	***виконано***
7	Надати пакету документів по ДР та супровідних до неї документів на засідання кафедри		***виконано***
8	Подання ДР рецензенту. Отримання рецензії.	09.06 – 13.06.2025	***виконано***
9	Захист ДР в ЕК	*16.06 - -21.06.2025*	

Студент _____ **Олександр ЦАРИК**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **МИРОСЛАВА ГЛАДКА**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **МАРИНА СИЧИК**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота за темою «Програмний застосунок сімейного лікаря для управління медичними записами в приватній клініці приміського типу» виконаний студентом кафедри біомедичної кібернетики ФБМІ *Цариком Олександром Миколайовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 5 розділів (аналіз аналогів програмного забезпечення, засоби реалізації програмного продукту, обґрунтування вибору програмного середовища розробки, програмна реалізація та методика використання програмного продукту, узагальнення результатів дослідження), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 31 джерела. Загальний обсяг роботи 75 сторінок.

Актуальність теми.

У сучасних умовах цифровізації особливого значення набуває впровадження електронних медичних систем. Електронна медична картка підвищує ефективність роботи закладів, якість послуг і безперервність лікування. Існуючі рішення, як-от Medtech, Epic Systems чи OpenMRS, мають недоліки: складну інтеграцію, потребу в постійному інтернеті, повільну обробку великих даних. Це ускладнює роботу, особливо у віддалених регіонах чи приватній практиці. Розробка автономного офлайн-застосунку для сімейного лікаря з простою інтеграцією та швидкою обробкою інформації є актуальною. Такий підхід підвищує стабільність, зменшує залежність від зовнішніх ресурсів і забезпечує безпечне зберігання персональних даних. Тема роботи є практично значущою й відповідає сучасним вимогам охорони здоров'я та інформаційної безпеки.

Мета і завдання роботи.

Підвищення ефективності ведення електронної медичної документації сімейними лікарями в умовах обмеженого доступу до інтернету шляхом

створення зручного, безпечного та автономного програмного застосунку для локального зберігання, обробки та управління медичними записами пацієнтів.

Досягнення поставленої мети передбачає вирішення наступних завдань:

- провести аналіз існуючих програмних рішень для роботи з електронними медичними картками, виявити їх переваги та недоліки.
- визначити функціональні та технічні вимоги до застосунку відповідно до потреб сімейного лікаря.
- обґрунтувати вибір мови програмування, архітектури, бази даних та інших інструментів реалізації.
- розробити інтуїтивно зрозумілий інтерфейс користувача з підтримкою світлої та темної тем.
- реалізувати основні функції програми: авторизацію користувача, додавання, редагування, пошук та фільтрацію даних пацієнтів.
- забезпечити локальне зберігання даних та їх захист відповідно до принципів інформаційної безпеки.
- провести тестування працездатності програмного продукту та оцінити його ефективність у реальних умовах використання.

Використані методи.

У процесі розробки програмного забезпечення для електронного ведення медичних записів сімейного лікаря були використані такі методи:

- аналіз предметної області — вивчення існуючих медичних систем, виявлення їхніх переваг і недоліків, формування вимог до власного продукту.
- каскадна модель розробки — послідовне проходження етапів: аналіз, проєктування, реалізація, тестування, впровадження.
- структурне та об’єктно-орієнтоване проєктування — для модульності та розширюваності системи.
- модульне програмування — для спрощення тестування, супроводу та масштабування.
- інкапсуляція та абстрагування — для гнучкості та безпеки архітектури.

- валідація даних — для забезпечення точності введеної медичної інформації.
- тестування — модульне, інтеграційне та системне, для перевірки працездатності застосунку на всіх етапах.

Отримані результати.

Розроблений програмний застосунок забезпечує ефективне ведення електронних медичних записів у сімейній медичній практиці без необхідності підключення до Інтернету. Лікарі можуть швидко додавати, редагувати, переглядати та шукати інформацію про пацієнтів, включаючи діагнози, висновки та контактні дані.

Система гарантує безпеку та конфіденційність персональних даних завдяки локальному зберіганню та механізмам валідації введених даних. Інтерфейс користувача розроблено з акцентом на простоту та зручність — підтримується зміна теми (світла/темна), швидкий пошук і сортування записів.

Застосунок оптимізує щоденну роботу лікаря, зменшує адміністративне навантаження, прискорює доступ до клінічної інформації та підвищує точність ведення медичної документації.

Апробація результатів роботи не планується.

Ключові слова

- *електронна медична картка;*
- *програмне забезпечення;*
- *сімейний лікар;*
- *локальна база даних;*
- *JavaFX;*
- *інформаційна безпека;*
- *автономний застосунок.*

Бібліографічний опис ДР

Царик О.М. Програмний застосунок сімейного лікаря для управління медичними записами в приватній клініці приміського типу. : дипломна роб. бакалавра : 122 Комп’ютері науки / Царика Олександра Миколайовича. – Київ, 2025. – 75 с

ANNOTATION

The bachelor's thesis titled *"A Software Application for a Family Doctor to Manage Medical Records in a Suburban Private Clinic"* was completed by Oleksandr Mykolaiovych Tsaryk, a student of the Department of Biomedical Cybernetics at the Faculty of Biomedical Engineering. The work was carried out in the field of study 122 "Computer Science" under the educational and professional program "Computer Technologies in Biology and Medicine." The thesis consists of the following parts: introduction; five chapters (analysis of existing medical software, Implementation tools, Justification of development environment, Software implementation and usage methodology, Summary of research results); conclusions to each chapter; general conclusions; a list of 31 references. The total volume of the thesis is 75 pages.

Relevance of the topic

In the context of rapid digitalization and the growing demand for immediate access to medical data, the implementation of electronic medical systems has become increasingly critical. The Electronic Medical Record is a key tool for improving the efficiency of healthcare delivery, enhancing service quality, and ensuring continuity of care. Despite the existence of many systems such as Medtech, Epic Systems, and OpenMRS, most suffer from drawbacks like difficult integration, reliance on internet connectivity, and sluggish performance with large datasets. These issues pose challenges, especially in private practice or remote areas. The development of a standalone application for family doctors that offers offline access, ease of integration, and fast data processing is timely and relevant. This approach minimizes dependence on external resources, enhances software stability, and ensures secure storage of sensitive personal data. Thus, the thesis topic aligns with current healthcare and information security demands, holds practical value, and contributes to the digital transformation of medical services.

Purpose and objectives

The goal of this work is to improve the efficiency of managing electronic medical records by family doctors in offline settings through the development of a secure, user-friendly standalone software application that ensures fast access,

reliable editing, and local data storage, tailored to the practical needs of everyday medical practice.

To achieve this goal, the following tasks were set:

- analyze existing EMR software solutions and identify their strengths and weaknesses.
- define functional and technical requirements based on the needs of a family doctor.
- justify the choice of programming language, architecture, database, and other tools.
- develop a user-friendly interface with support for light and dark themes.
- implement core functionalities such as user authentication, adding, editing, searching, and filtering patient data.
- ensure local data storage and protection in line with information security principles.
- conduct software testing and evaluate its performance in real-life scenarios.

Methods used

During the development of the software for managing electronic medical records, the following methods were applied:

- domain analysis, including the study of existing EMR systems, their pros and cons, and formulation of requirements for the custom application.
- classical waterfall software development model involving sequential phases: requirements analysis, design, implementation, testing, and deployment — enabling thorough control and risk mitigation.
- structural and object-oriented design approaches to define system components, ensure modularity, and facilitate future expansion.
- modular programming to independently implement key functionalities, simplifying testing and maintenance.
- encapsulation and abstraction to enhance architectural flexibility and security.
- data validation methods to ensure input correctness, especially critical in medical documentation.
- testing techniques such as unit, integration, and system testing to verify application functionality at all development stages.

Results obtained

The developed software application enables efficient electronic medical record management in family medical practice without internet access. It allows physicians to quickly add, edit, view, and search for patient information, including diagnoses, notes, and contact details. The system ensures personal data confidentiality and security through local storage and data validation mechanisms. The user interface emphasizes ease of use, includes theme switching (light/dark), and supports fast search and record sorting.

The application optimizes daily workflow, reduces administrative burden, accelerates access to clinical information, and improves the accuracy of medical documentation.

Practical implementation of the results is not planned.

Keywords

- *electronic medical record;*
- *software application;*
- *family doctor;*
- *local database;*
- *JavaFX;*
- *data security;*
- *offline system.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (*Application Programming Interface*) — інтерфейс прикладного програмування, набір функцій для взаємодії програмних компонентів.

CSS (*Cascading Style Sheets*) — каскадні таблиці стилів, мова стилізації елементів інтерфейсу користувача.

ЦБД — централізована база даних, єдине сховище даних, доступне багатьом користувачам або системам.

ЕМК — електронна медична картка, цифрова форма зберігання історії хвороби пацієнта.

FXML — розмітка XML для опису інтерфейсу користувача в JavaFX.

ФВА — форма введення або редагування даних, представлена у вікні програми.

GDPR (*General Data Protection Regulation*) — загальний регламент захисту персональних даних, що діє в Європейському Союзі.

HIPAA (*Health Insurance Portability and Accountability Act*) — американський закон, що регулює конфіденційність медичних даних.

ID — унікальний ідентифікатор запису в базі даних.

IntelliJ IDEA — інтегроване середовище розробки програмного забезпечення на мові Java.

Java — мова програмування, що використовується для розробки додатку.

JavaFX — фреймворк для створення графічних інтерфейсів Java-застосунків.

JDBC (*Java Database Connectivity*) — інтерфейс для з’єднання Java-додатку з базами даних.

JSON (*JavaScript Object Notation*) — текстовий формат обміну даними, часто використовується для передачі структурованої інформації.

MIC — медична інформаційна система, програмний комплекс для обліку, обробки та зберігання медичних даних.

MVC (*Model-View-Controller*) — архітектурний шаблон, що розділяє дані, інтерфейс і логіку керування.

PDF (*Portable Document Format*) — формат збереження електронних документів, зокрема звітів та форм.

ПЗ — програмне забезпечення, сукупність програм, призначених для виконання певних функцій.

SQLite — легка реляційна система керування базами даних, яка не потребує встановлення серверу.

UI (*User Interface*) — інтерфейс користувача, засоби взаємодії з програмним продуктом.

UX (*User Experience*) — досвід користувача при взаємодії з програмою.

XML — мова розмітки для опису структури документів.

ВСТУП

У сучасних умовах цифровізації особливого значення набуває впровадження електронних медичних систем. Електронна медична картка підвищує ефективність роботи закладів, якість послуг і безперервність лікування. Існуючі рішення, як-от Medtech, Epic Systems чи OpenMRS, мають недоліки: складну інтеграцію, потребу в постійному інтернеті, повільну обробку великих даних. Це ускладнює роботу, особливо у віддалених регіонах чи приватній практиці. Розробка автономного офлайн-застосунку для сімейного лікаря з простою інтеграцією та швидкою обробкою інформації є актуальною. Такий підхід підвищує стабільність, зменшує залежність від зовнішніх ресурсів і забезпечує безпечне зберігання персональних даних. Тема роботи є практично значущою й відповідає сучасним вимогам охорони здоров'я та інформаційної безпеки.

Мета і завдання роботи.

Підвищення ефективності ведення електронної медичної документації сімейними лікарями в умовах обмеженого доступу до інтернету шляхом створення зручного, безпечного та автономного програмного застосунку для локального зберігання, обробки та управління медичними записами пацієнтів.

Досягнення поставленої мети передбачає вирішення наступних завдань:

- провести аналіз існуючих програмних рішень для роботи з електронними медичними картками, виявити їх переваги та недоліки.
- визначити функціональні та технічні вимоги до застосунку відповідно до потреб сімейного лікаря.
- обґрунтувати вибір мови програмування, архітектури, бази даних та інших інструментів реалізації.
- розробити інтуїтивно зрозумілий інтерфейс користувача з підтримкою світлої та темної тем.
- реалізувати основні функції програми: авторизацію користувача, додавання, редагування, пошук та фільтрацію даних пацієнтів.
- забезпечити локальне зберігання даних та їх захист відповідно до принципів інформаційної безпеки.

- провести тестування працездатності програмного продукту та оцінити його ефективність у реальних умовах використання.

Використані методи.

У процесі розробки програмного забезпечення для електронного ведення медичних записів сімейного лікаря були використані такі методи:

- аналіз предметної області — вивчення існуючих медичних систем, виявлення їхніх переваг і недоліків, формування вимог до власного продукту.
- каскадна модель розробки — послідовне проходження етапів: аналіз, проєктування, реалізація, тестування, впровадження.
- структурне та об’єктно-орієнтоване проєктування — для модульності та розширюваності системи.
- модульне програмування — для спрощення тестування, супроводу та масштабування.
- інкапсуляція та абстрагування — для гнучкості та безпеки архітектури.
- валідація даних — для забезпечення точності введеної медичної інформації.
- тестування — модульне, інтеграційне та системне, для перевірки працездатності застосунку на всіх етапах.

Отримані результати.

Розроблений програмний застосунок забезпечує ефективне ведення електронних медичних записів у сімейній медичній практиці без необхідності підключення до Інтернету. Лікарі можуть швидко додавати, редагувати, переглядати та шукати інформацію про пацієнтів, включаючи діагнози, висновки та контактні дані.

Система гарантує безпеку та конфіденційність персональних даних завдяки локальному зберіганню та механізмам валідації введених даних. Інтерфейс користувача розроблено з акцентом на простоту та зручність — підтримується зміна теми (світла/темна), швидкий пошук і сортування записів.

Застосунок оптимізує щоденну роботу лікаря, зменшує адміністративне навантаження, прискорює доступ до клінічної інформації та підвищує точність ведення медичної документації.

Апробація результатів роботи не планується.

Публікації. Не заплановано

Структура роботи

Дипломна робота за темою «Програмний застосунок сімейного лікаря для управління медичними записами в приватній клініці приміського типу» виконаний студентом кафедри біомедичної кібернетики ФБМІ *Цариком Олександром Миколайовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 5 розділів (аналіз аналогів програмного забезпечення, засоби реалізації програмного продукту, обґрунтування вибору програмного середовища розробки, програмна реалізація та методика використання програмного продукту, узагальнення результатів дослідження), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 31 джерел, та додатків. Загальний обсяг роботи 75 сторінок. В роботі представлено 19 рисунків і 2 таблиці.

РОЗДІЛ 1
АНАЛОГИ ПРОГРАМНОГО ПРОДУКТУ

У розділі наведено огляд сучасних систем для управління електронними медичними картками, таких як Medtech, Epic Systems та OpenMRS. Розглянуто їх функціональні можливості, переваги та недоліки. Також описано, як у власному програмному забезпеченні були враховані недоліки існуючих рішень — додано офлайн-доступ, спрощено інтеграцію з іншими системами та покращено швидкодію, що підвищує ефективність роботи лікарів у реальних умовах[1].

1.1 Аналіз сучасних систем управління електронними медичними картками та їх вплив на розробку програмного забезпечення

Огляд існуючих систем для управління електронними медичними картками.

Системи для управління електронними медичними картками (ЕМК) стали важливим інструментом в охороні здоров'я, надаючи лікарям та медичним працівникам можливість ефективно зберігати, обробляти та використовувати медичні дані пацієнтів. Існують численні програмні рішення, які активно використовуються в медичних установах по всьому світу. Ось деякі з них:

Medtech — це сучасна система для ведення медичних записів у реальному часі, яка дозволяє лікарям ефективно управляти даними про своїх пацієнтів. Вона дає можливість зберігати історію хвороби, включаючи діагнози, лікування, рецепти, лабораторні результати та інші важливі медичні дані. Крім того, система інтегрується з апаратами та медичними засобами, що дозволяє лікарям отримувати актуальну інформацію про стан пацієнта під час прийому, покращуючи процес діагностики та лікування. Medtech забезпечує надійну платформу для організації робочих процесів у медичних установах, що робить доступ до важливої інформації швидким та зручним[2].

Epic Systems є однією з найбільших та найпоширеніших платформ для управління медичними картками, активно використовуваною в лікарнях і клініках по всьому світу. Вона підтримує електронні медичні картки, лабораторні дані, рецепти та виписки. Epic Systems не тільки дозволяє лікарям і медичним працівникам ефективно управляти медичною інформацією, але й інтегрується з іншими системами охорони здоров'я, що дозволяє створювати єдину базу даних на рівні національних та міжнародних мереж. Ця інтеграція забезпечує обмін інформацією між медичними установами, що важливо для надання високоякісної медичної допомоги, особливо у випадках, коли пацієнт отримує лікування в кількох закладах[3].

OpenMRS — це відкрите програмне забезпечення для управління медичними даними, яке знаходить широке застосування, особливо в країнах, що розвиваються. Завдяки своїй доступності та гнучкості, OpenMRS дозволяє медичним установам адаптувати систему під свої конкретні потреби, що робить її дуже корисною для організацій з обмеженими ресурсами. Вона підтримує зберігання та обробку медичної інформації, а також дає можливість кастомізації для задоволення специфічних вимог. Однак варто зазначити, що її використання вимагає додаткових ресурсів для налаштування та постійної підтримки системи, що може бути складним для деяких медичних установ, особливо в умовах обмежених бюджетів або технічних можливостей[4].

1.2Порівняння функціональних можливостей, інтерфейсів та переваг/недоліків цих систем.

У таблиці 1.1 наведено порівняльний аналіз трьох популярних медичних інформаційних систем: Medtech, Epic та OpenMRS. Розгляд здійснено за такими критеріями: основні функції, переваги та недоліки кожної системи.

Аналіз показує, що кожна з розглянутих систем має свої унікальні переваги та обмеження. Medtech вирізняється зручністю використання та швидким доступом до інформації, проте вимагає значних фінансових і людських ресурсів для впровадження. Epic є потужною та масштабованою системою, яка підходить для великих медичних установ, однак її складність

може бути бар'єром на початковому етапі. OpenMRS, у свою чергу, є оптимальним вибором для організацій з обмеженим бюджетом, завдяки відкритому коду та можливості адаптації до конкретних потреб, але потребує технічної підтримки[32].

Таблиця 1.1

Порівняльна характеристика медичних інформаційних систем

Система	Основні функції	Переваги	Недоліки
Medtech	Ведення медичних записів, управління апаратами, історія хвороби пацієнта	Реальний час доступу до інформації, зручний інтерфейс	Висока вартість впровадження, потребує спеціалізованих навичок
Epic	Інтеграція з лабораторіями, ведення електронних карток, управління фінансами	Висока масштабованість, інтеграція з іншими системами	Складність впровадження, велика кількість функцій
OpenMRS	Зберігання медичних записів, кастомізація інтерфейсу, доступність через інтернет	Відкрите джерело, гнучкість налаштувань, доступність для країн з обмеженими ресурсами	Потребує додаткових ресурсів для налаштування та обслуговування

Таким чином, вибір конкретної системи залежить від масштабів медичного закладу, доступного бюджету та технічної готовності персоналу.

1.3 Врахування проблем та обмежень існуючих медичних систем при розробці програмного забезпечення для електронних карток

При розробці програмного забезпечення для електронних медичних карток я врахував кілька важливих проблем та обмежень, що є у вже існуючих

рішеннях. Це дозволило мені створити продукт, який є зручнішим і ефективнішим у використанні в реальних умовах.

Однією з основних проблем сучасних медичних систем є потреба в постійному підключенні до інтернету для доступу до даних. Це створює труднощі для медичних установ, які працюють у віддалених районах, де стабільне інтернет-з'єднання може бути відсутнє. Щоб вирішити цю проблему, я додав можливість офлайн роботи в нашому програмному забезпеченні. Лікарі можуть працювати з медичними картками пацієнтів навіть без інтернету, а всі зміни автоматично синхронізуються, як тільки зв'язок буде відновлений[5].

Ще одна проблема багатьох медичних систем — складність інтеграції з іншими платформами. Більшість великих систем, таких як Еріс, мають складні механізми інтеграції, які займають багато часу та потребують значних ресурсів. Я зробив інтеграцію простішою, розробивши інтерфейси, які дозволяють швидко підключати нові модулі або об'єднувати нашу систему з іншими платформами без великих зусиль[5].

Також існуючі рішення часто мають проблеми зі швидкістю роботи, особливо коли потрібно обробляти великі обсяги даних. Це може призвести до затримок і уповільнити роботу медичних працівників. Я оптимізував структуру бази даних і архітектуру програмного забезпечення, що дозволяє ефективно працювати з великими обсягами інформації без значних затримок[6].

Завдяки вирішенню цих проблем моє програмне забезпечення стало більш зручним і ефективним для використання в реальних умовах, де інші системи можуть виявитися менш зручними або занадто складними для впровадження.

Висновок з розділу 1

Огляд існуючих систем управління електронними медичними картками в Україні показує значний прогрес у цифровізації медичної сфери. Основні переваги ЕМК - це швидкий доступ лікарів до повної історії пацієнта, можливість обміну інформацією між різними медичними установами,

зниження часу на паперову документацію та підвищення якості медичної допомоги.

Водночас існуючі системи мають обмеження, зокрема:

- потребу в постійному підключенні до Інтернету для роботи з централізованою базою даних,
- складність інтеграції різних медичних інформаційних систем,
- проблеми з оптимізацією швидкості обробки даних через навантаження на сервери.

Розроблене програмне забезпечення, яке забезпечує офлайн-доступ, спрощену інтеграцію та високу швидкість обробки даних, ефективно усуває низку ключових проблем, характерних для сучасних електронних медичних систем. Завдяки автономній роботі застосунок не залежить від стабільності інтернет-з’єднання, що значно підвищує його надійність і дозволяє лікарям користуватися всіма функціями навіть у польових умовах, мобільних медичних пунктах, сільських амбулаторіях чи під час виїздів на виклики.

Крім того, спрощена архітектура системи зручна для впровадження у закладах з обмеженими технічними або фінансовими ресурсами. Завдяки зрозумілому інтерфейсу та мінімальним вимогам до обладнання, програмний продукт може бути використаний як у великих державних лікарнях, так і у приватних медичних кабінетах. Така гнучкість і масштабованість робить систему привабливою для широкого кола користувачів і сприяє її поширенню в національній системі охорони здоров’я.

Можливість швидкого доступу до медичних записів у поєднанні з функціями фільтрації, пошуку та редагування дає змогу лікарю оперативно приймати клінічні рішення, не витрачаючи зайвий час на бюрократичні процедури. Це, у свою чергу, покращує якість надання медичної допомоги, знижує ризик медичних помилок і підвищує задоволеність пацієнтів.

Таким чином, впровадження таких удосконалених систем електронної медичної картки сприяє формуванню більш ефективного, безпечного та доступного медичного середовища. Вони дозволяють лікарям більше уваги зосереджувати на клінічному процесі та безпосередньому спілкуванні з

пацієнтами, скорочуючи навантаження, пов’язане з адміністративними завданнями, документацією та технічними бар’єрами. У результаті покращується не лише індивідуальний рівень медичного обслуговування, а й загальна ефективність роботи медичних установ.

РОЗДІЛ 2
ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

У розділі описано методи, інструменти, процедури та процеси розробки програмного забезпечення для сімейного лікаря. Розробка здійснювалася відповідно до класичної каскадної моделі життєвого циклу ПЗ, що включала етапи від збору вимог до впровадження готового продукту в умовах реального використання. Під час реалізації було застосовано структурне та об'єктно-орієнтоване проєктування, модульну організацію компонентів, перевірені інженерні практики й сучасні засоби, зокрема IntelliJ IDEA, JavaFX, SQLite, JDBC та CSS.

Проєкт орієнтований на практичні потреби лікаря, що працює в умовах приватної амбулаторної практики або невеликого медичного закладу. Його реалізація враховує вимоги до простоти, автономності, стабільності, безпеки й зручності використання. Для забезпечення якості програмного продукту було реалізовано механізми валідації введених даних, ручного та автоматизованого тестування, документування та підтримки коду. Наведені нижче підрозділи деталізують кожен із напрямів реалізації.

2.1 Методи

Для розробки було обрано класичну каскадну модель, яка передбачає чітко визначену послідовність етапів. Цей підхід виявився доцільним для проєкту з наперед визначеними вимогами та відносно стабільною структурою функціональності. Він дозволив ретельно пропрацювати кожен етап, зосередившись на якісному виконанні завдань без потреби повертатися до попередніх кроків. На першому етапі аналізу визначалися функціональні вимоги до системи — зокрема, можливість створення та збереження медичних записів, швидкий пошук пацієнтів, генерація звітів у PDF-форматі, а також дотримання вимог безпеки персональних даних. На основі цих вимог формувалася структура бази даних і графічний інтерфейс, що згодом визначило архітектуру всієї програми[7].

Під час проєктування враховувалися два ключові підходи: структурне та об’єктно-орієнтоване проєктування. Структурний підхід дав змогу чітко визначити логіку взаємозв’язків між компонентами бази даних, а також забезпечити оптимальну організацію зберігання інформації[8].

Натомість об’єктно-орієнтований підхід дозволив гнучко описати взаємодію між різними частинами програми, чітко відокремити логіку від представлення й дотримуватись принципів чистої архітектури. Особливу увагу приділяли модульності — кожен логічний блок програми реалізовано як окремий модуль, що значно спростило тестування, обслуговування та розширення функціональності. Наприклад, функціонал фільтрації, експорту або додавання записів реалізовано незалежно, що дало змогу перевіряти й оновлювати їх без ризику порушити роботу інших частин системи[9].

Код писався з дотриманням принципів інкапсуляції, абстрагування та мінімізації зв’язності між модулями. Це дозволило приховати технічні деталі реалізації та зробити систему більш гнучкою до змін і масштабування.[10]

Завдяки повторному використанню коду було скорочено обсяг дублювання: наприклад, створені універсальні класи для підключення до бази даних, обробки введених даних або генерації PDF-файлів повторно застосовувалися в різних компонентах програми[10].

Також активно використовувалися шаблони проєктування — зокрема, шаблон «одинак» (singleton) для контролю доступу до єдиної точки підключення до бази[11].

Тестування проводилося на всіх етапах розробки. На рівні окремих модулів застосовувалося модульне тестування, що дозволяло перевіряти правильність логіки окремих функцій без участі всієї системи. Після інтеграції компонентів проводилися інтеграційні тести, які виявляли помилки у взаємодії між модулями. Крім того, здійснювалося системне тестування, під час якого перевірялися всі основні сценарії користувача — від додавання нового пацієнта до експорту всієї бази даних у PDF-файл. Тестування охоплювало як позитивні сценарії, коли введені дані коректні, так і негативні, наприклад, спробу залишити обов’язкові поля порожніми або ввести текст замість чисел[12].

Усі помилки, виявлені під час тестування, фіксувалися та одразу виправлялися. Завдяки цьому вдалося досягти стабільної та надійної роботи програми навіть у випадку активної взаємодії з великою кількістю записів. На завершальному етапі програма адаптувалась під реальні потреби медичного закладу, що включало фінальне тестування, уточнення функцій згідно з відгуками користувачів та підготовку зрозумілої користувацької інструкції. Це дозволило створити програмний продукт, який не лише повністю відповідає технічному завданню, а й зручний для щоденного використання медичним персоналом[12].

2.2Інструменти

Розробка виконувалась у середовищі IntelliJ IDEA, яке надало всі необхідні інструменти для зручної та продуктивної роботи з кодом. Підсвітка синтаксису, система автозавершення, відлагодження коду та вбудована інтеграція з системами контролю версій, зокрема Git, значно прискорили процес написання й налагодження програми. Крім того, середовище забезпечило гнучку конфігурацію проєкту та підтримку роботи з бібліотеками й залежностями, що було особливо корисним на етапі інтеграції зовнішніх модулів[13].

Для створення графічного інтерфейсу обрано JavaFX — сучасну бібліотеку для побудови настільних інтерфейсів, яка поєднує гнучкість налаштувань із хорошою продуктивністю. Вона дозволила реалізувати зрозуміле, візуально привабливе й водночас функціональне середовище взаємодії з користувачем. Структура інтерфейсу була продумана таким чином, щоб усі основні дії — додавання пацієнта, перегляд картки, фільтрація та експорт даних — виконувалися інтуїтивно, без потреби в додаткових інструкціях. Завдяки використанню CSS вдалося точно налаштувати вигляд елементів, уніфікувати стиль і зробити застосунок візуально цілісним. Це позитивно вплинуло на зручність роботи, особливо в умовах щоденного використання лікарями[14].

У ролі системи керування базами даних обрана SQLite — вбудована СКБД, яка не вимагає окремого сервера або складного налаштування. Вона зберігає всі дані в одному файлі, що спрощує супровід і перенесення даних у разі потреби. Такий підхід виявився ідеальним для настільного застосунку, який має працювати офлайн у приватній клініці без залежності від зовнішньої інфраструктури. Підключення до бази даних реалізовано через технологію JDBC — універсальний інтерфейс доступу до СКБД, який забезпечив стабільну і швидку роботу з даними, включаючи додавання, пошук, оновлення та видалення записів[15].

Усі інструменти й технології були підібрані з урахуванням простоти розгортання, невисоких вимог до апаратного забезпечення та легкості подальшого супроводу. Завдяки цьому програму можна швидко встановити на будь-якому комп’ютері з Java, не потребуючи тривалого налаштування чи спеціальних знань. Це особливо важливо для медичного персоналу, який цінує простоту та надійність у роботі з електронними інструментами.

2.3Процедури

Розробка проходила за заздалегідь визначеним планом, який охоплював усі ключові етапи життєвого циклу програмного забезпечення — від збору вимог до тестування та впровадження. Першочерговим завданням стало вивчення потреб майбутніх користувачів. Для цього проводилися консультації з лікарями, які безпосередньо працюватимуть із програмою. Їхні коментарі та побажання стали основою для формування функціональних вимог, що дозволило сфокусуватися на дійсно потрібних функціях і уникнути перевантаження інтерфейсу другорядними опціями[16].

Особливу увагу приділено валідації даних. Усі дані, які вводяться користувачем, проходять перевірку на правильність і повноту: обов’язкові поля не можуть залишатися порожніми, дати перевіряються на реалістичність, а текстові поля — на відповідність очікуваному формату. Це дозволило мінімізувати ризик збереження помилкової або неповної інформації, що є

критично важливим у медичному контексті, де кожна деталь може мати значення[17].

У процесі розробки активно залучалися потенційні користувачі, з якими регулярно узгоджувалися макети інтерфейсу, функціональні сценарії та логіка роботи програми. Такий підхід дав змогу оперативно виявляти й усувати незручності, адаптуючи застосунок до звичних для лікарів робочих процесів. Завдяки цьому остаточна версія програми виявилася інтуїтивно зрозумілою та придатною до щоденного використання без потреби в тривалому навчанні.

Кодова база підтримувалась у чистому та структурованому стані. Усі класи й методи супроводжувались зрозумілими коментарями відповідно до загальноприйнятих стандартів. Це значно полегшує майбутнє обслуговування системи, внесення змін і передачу проєкту іншим розробникам, якщо така потреба виникне.

2.4Процеси розробки

Розробка програмного забезпечення розпочалася з глибокого вивчення потреб майбутніх користувачів — сімейних лікарів, які щодня працюють із великою кількістю пацієнтів. Спілкування з медичним персоналом дозволило чітко окреслити основні сценарії використання програми та визначити ключові вимоги до її функціональності. Це дало змогу сформувати концепцію системи, яка відповідає реальним умовам роботи в приватній клініці[18].

Після узагальнення зібраної інформації розпочався етап технічного проєктування. Було розроблено структуру бази даних, орієнтовану на ефективне зберігання та швидкий доступ до інформації про пацієнтів і їхні медичні записи. Конструкція сховища передбачає надійне збереження чутливих даних і спрощує фільтрацію та пошук потрібної інформації[18].

На основі створеної архітектури почалась розробка графічного інтерфейсу. Його основна мета — забезпечити інтуїтивну взаємодію користувача з програмою, навіть за мінімального рівня технічної підготовки. Інтерфейс вийшов зрозумілим, функціональним і придатним для щоденного використання в медичній практиці[18].

Далі була реалізована бізнес-логіка — функціональні механізми, що забезпечують обробку введених даних, їх валідацію, фільтрацію, збереження та представлення у вигляді звітів. З'єднання з базою даних здійснювалося через JDBC, що забезпечило стабільну й ефективну взаємодію між користувацьким інтерфейсом і внутрішнім сховищем[18].

Завершальним етапом стало всебічне тестування, яке включало перевірку окремих функцій, перевірку взаємодії компонентів системи та тестування повної програми в умовах, наближених до реального використання. Після виправлення виявлених помилок була підготовлена супровідна документація, що містить інструкції з встановлення та користування програмою. Готове рішення передано до клініки для практичного використання[18].

Висновок до розділу 2

Розробка програмного забезпечення для сімейного лікаря здійснювалася з урахуванням перевірених підходів інженерії програмного забезпечення, що забезпечило структурованість та контроль на кожному етапі. Основою проєкту стала каскадна модель, яка дозволила послідовно перейти від формування вимог до тестування та впровадження, мінімізуючи ризики і непередбачувані зміни.

Застосування сучасних інструментів значно спростило процес реалізації. Середовище IntelliJ IDEA забезпечило комфортну роботу з кодом завдяки зручним інструментам автодоповнення, навігації та відлагодження. JavaFX дозволив створити інтуїтивно зрозумілий і візуально приємний інтерфейс, адаптований під потреби медичних працівників. Застосування CSS допомогло налаштувати зовнішній вигляд елементів і підвищити зручність взаємодії з програмою. Як база даних була обрана SQLite — легка, вбудована СКБД, що не потребує окремого серверного середовища. Для з’єднання з нею використано JDBC, який забезпечив надійну інтеграцію з додатком.

Програмна архітектура будувалась на принципах модульності та повторного використання коду. Кожен логічний компонент мав чітко визначене

призначення, що полегшувало супровід, тестування та потенційне розширення функціоналу. Значну увагу приділено інкапсуляції даних та спрощенню доступу до основних функцій програми, що зробило її зручною у користуванні навіть для фахівців без глибоких технічних знань.

Ключовою метою було створити не просто працюючий інструмент, а програму, яка дійсно полегшує щоденну роботу лікаря. Завдяки ретельному тестуванню, залученню кінцевих користувачів на етапах перевірки та адаптації, система вийшла стабільною, зрозумілою і повністю готовою до використання в реальних умовах приватної практики.

РОЗДІЛ 3
ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНОГО СЕРЕДОВИЩА

У цьому розділі наведено аргументацію щодо вибору технологій, архітектурного підходу та середовищ розробки, які були використані під час створення програмного забезпечення для підтримки роботи сімейного лікаря. Враховуючи специфіку проєкту — створення зручного у користуванні, автономного застосунку для ведення електронних медичних записів — особливу увагу було приділено стабільності роботи, простоті розгортання, зрозумілості інтерфейсу та можливості масштабування у майбутньому. Кожне обране рішення базується на реальних потребах медичної практики, з якими автор мав змогу ознайомитися під час аналізу вимог користувачів[33].

Окрему увагу було звернено на захист персональних даних, адже програмне забезпечення працює з конфіденційною інформацією пацієнтів. У цьому контексті важливим було не лише запобігти технічним уразливостям, а й створити архітектуру, яка мінімізує ризики людських помилок. Розглядалися як загальні підходи до інформаційної безпеки в медичних IT-системах, так і практичні методи їх реалізації в умовах локального застосунку без підключення до Інтернету. Таким чином, розділ охоплює як технічні, так і організаційні аспекти вибору програмного середовища[34].

3.1Обґрунтування вибору технологій і архітектури для вирішення поставленої задачі

Створення автономного десктопного застосунку для приватної медичної практики вимагало ретельного вибору технологічного стеку, орієнтованого на простоту використання, надійність і довговічність рішення. У цьому контексті було прийнято рішення використати мову програмування Java, яка на сьогодні є однією з найпопулярніших і найбільш універсальних платформ для розробки багатофункціональних додатків. Java забезпечує кросплатформеність, що дає змогу запускати програму як на Windows, так і на Linux або macOS без значних

модифікацій коду. Крім того, Java відзначається високим рівнем безпеки та широкою підтримкою багатьох бібліотек і фреймворків, що значно прискорює розробку і підвищує якість кінцевого продукту[19].

Для створення графічного інтерфейсу користувача був обраний фреймворк JavaFX — сучасний інструмент, який дозволяє створювати красиві, функціональні та інтуїтивно зрозумілі інтерфейси. JavaFX підтримує багатий набір елементів керування, а також дає можливість легко реалізовувати анімації, переходи між вікнами та інші візуальні ефекти, що покращують взаємодію користувача із застосунком. Окрім цього, використання CSS для стилізації інтерфейсу зробило можливим реалізувати гнучкий і привабливий дизайн, який одночасно відповідає принципам юзабіліті та адаптований під різні типи користувачів, зокрема тих, хто не має спеціальної технічної підготовки[14].

Для зберігання даних було обрано базу даних SQLite, що є одним із найпопулярніших варіантів для локальних застосунків. SQLite вирізняється своєю легкістю, відсутністю потреби у встановленні окремого серверного програмного забезпечення та мінімальними вимогами до ресурсів. Така база ідеально підходить для автономних застосунків, які мають зберігати і обробляти дані локально, без необхідності постійного підключення до мережі Інтернет. Для забезпечення комунікації між Java-застосунком та базою даних використано JDBC (Java Database Connectivity) — стандартний інтерфейс доступу до реляційних баз даних у Java. Це дозволило реалізувати стабільний та ефективний обмін інформацією, а також зручно управляти транзакціями, виконувати запити і обробляти результати[15].

З архітектурної точки зору застосунок побудовано за модульним принципом, що передбачає чітке розділення на три основні рівні: графічний інтерфейс користувача, бізнес-логіку та рівень взаємодії з базою даних. Такий підхід забезпечує низку переваг, серед яких — висока гнучкість, простота тестування окремих компонентів, зручність у підтримці та подальшому масштабуванні системи. Модульність сприяє кращій інкапсуляції, тобто приховуванню внутрішніх деталей реалізації кожного рівня, що допомагає уникати несподіваних помилок і робить код більш зрозумілим та

структурованим. Завдяки застосуванню принципів об’єктно-орієнтованого програмування — таких як абстрагування, наслідування і повторне використання коду — вдалося створити стабільний і легко розширюваний застосунок, який можна адаптувати під нові вимоги та інтегрувати з іншими системами в майбутньому[20].

Розробка здійснювалася у професійному середовищі розробки IntelliJ IDEA, яке забезпечує комфортну та продуктивну роботу програміста. IntelliJ IDEA підтримує функції автозавершення коду, що значно прискорює написання програмних модулів і зменшує кількість синтаксичних помилок. Вбудовані інструменти для налагодження дозволяють відслідковувати роботу програми на різних етапах виконання, швидко знаходити і виправляти помилки. Крім того, середовище підтримує інтеграцію з системами контролю версій, такими як Git, що є важливим для організації командної роботи, збереження історії змін і забезпечення безпеки коду. Завдяки широкому набору плагінів і зручному інтерфейсу, IntelliJ IDEA значно підвищує ефективність розробки, що позитивно вплинуло на якість кінцевого продукту.

3.2Захист персональних даних

Особливу увагу під час розробки було приділено питанням інформаційної безпеки, адже робота з медичною інформацією передбачає обробку чутливих персональних даних пацієнтів, що вимагає дотримання високих стандартів конфіденційності та захисту. Навіть за відсутності підключення до мережі Інтернет, забезпечення безпеки на рівні локального зберігання і доступу до даних є пріоритетним завданням, оскільки будь-яке порушення може призвести до витоку або псування інформації.

Одним із базових заходів стало обмеження доступу до бази даних. Застосунок має ексклюзивні права на читання та зміну даних, що зберігаються локально, виключаючи будь-які сторонні програми або процеси від доступу до інформації. Для цього база даних розміщується у спеціально захищеному каталозі операційної системи, права на доступ до якого налаштовані таким чином, що лише авторизовані користувачі мають можливість працювати з

цими файлами. Це запобігає несанкціонованому копіюванню чи зміненню бази даних ззовні[21].

Важливою складовою безпеки стала комплексна валідація введених користувачем даних. Кожне поле форми проходить перевірку на відповідність очікуваному формату, типу і діапазону значень. Такий підхід не лише запобігає збереженню некоректної або потенційно шкідливої інформації, але й мінімізує ризики SQL-ін’єкцій та інших атак, пов’язаних з маніпуляцією даними. Валідація здійснюється як на рівні клієнтського інтерфейсу, так і в бізнес-логіці застосунку, що підвищує надійність та цілісність інформації.

Крім цього, архітектура застосунку передбачає можливість подальшого розширення функціоналу для забезпечення додаткових заходів безпеки. Зокрема, планується впровадження автентифікації користувачів, що дозволить встановлювати індивідуальні права доступу, контролювати активність користувачів і запобігати несанкціонованому використанню системи. Також розглядається реалізація шифрування локальних даних із застосуванням сучасних криптографічних алгоритмів, що дозволить захистити інформацію навіть у разі фізичного доступу до пристрою з базою даних. Ці додаткові механізми зроблять застосунок відповідним до вимог сучасних нормативних актів щодо захисту персональних даних, таких як GDPR або HIPAA, хоча він і розрахований на локальну експлуатацію[22].

Таким чином, всі обрані технології, методи розробки та архітектурні рішення забезпечують не лише зручність і функціональність застосунку, а й надійний рівень інформаційної безпеки. Це є надзвичайно важливим для медичних інформаційних систем, де кожен випадок порушення конфіденційності може мати серйозні юридичні та етичні наслідки, а також негативний вплив на довіру пацієнтів та репутацію медичного закладу.

Висновок до розділу 3

Розробка автономного десктопного застосунку для приватної медичної практики стала результатом свідомого вибору сучасних, надійних і перевірених технологій, що відповідають високим вимогам функціональності,

зручності та безпеки. Використання мови програмування Java у поєднанні з потужним фреймворком JavaFX забезпечило створення кросплатформного, інтуїтивного та естетично привабливого інтерфейсу, адаптованого до потреб користувачів різного рівня технічної підготовки. База даних SQLite, обрана як рішення для локального зберігання інформації, надала простоту у використанні та мінімальні системні вимоги, що важливо для автономної роботи без постійного доступу до мережі. Модульна архітектура з чітким розподілом функцій між інтерфейсом, бізнес-логікою та рівнем роботи з базою даних забезпечує гнучкість, масштабованість і простоту підтримки застосунку[23].

Окремо слід відзначити системний підхід до забезпечення інформаційної безпеки, що є критично важливим для медичних інформаційних систем. Обмеження доступу, ретельна валідація даних, а також передбачені механізми автентифікації і шифрування формують надійний захист персональної інформації пацієнтів. Це дає змогу гарантувати відповідність сучасним стандартам конфіденційності, зберігаючи довіру користувачів і репутацію медичного закладу[23].

Таким чином, реалізований застосунок є збалансованим і комплексним рішенням, яке задовольняє вимоги приватної медичної практики щодо ефективності, зручності, безпеки та подальшого розвитку. Його архітектура та технологічний стек створюють міцну основу для подальшої модернізації та інтеграції з іншими інформаційними системами в майбутньому.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі описується процес створення програмного продукту, який відповідає поставленим вимогам. Розглядаються основні етапи розробки - від проєктування користувацького інтерфейсу до обробки даних пацієнтів і роботи з базою даних. Також тут детально пояснюється, як працювати з програмою, включаючи її ключові функції та принципи їх реалізації.

Особливу увагу приділено вибору інструментів для розробки: мова програмування Java, фреймворк JavaFX для створення інтерфейсу та база даних SQLite для збереження інформації. Пояснюється, як ці технології взаємодіють між собою, забезпечуючи стабільну, безпечну та зручну роботу програми.

Наприкінці розділу наведені можливості для подальшого розширення функціоналу, що дозволить зробити програму ще більш потужною та підвищити ефективність роботи користувачів.

4.1 Структура програмного застосунку

Програмний продукт для управління даними пацієнтів та лікарів складається з кількох основних компонентів, які взаємодіють між собою для забезпечення зручності користування та ефективної обробки даних.

clinic(
    controller(
        EditPatientController.java
        LoginController.java
        PatientController.java)
    model(
        Database.java
        Patient.java)
    resources(

```
                dark-theme.css
                light-theme.css )
        view(
                EditPatientWindow.fxml
                LoginView.fxml
                patient_view.fxml )
    )
```

Нижче наведено опис основних частин структури цього застосунку:

1. clinic

Коренева директорія, що містить усі компоненти застосунку.

2. controller

Ця директорія містить класи-контролери, які відповідають за обробку подій та взаємодію між інтерфейсом користувача та моделями даних.

- editPatientController.java: клас контролера для вікна редагування даних пацієнта. Він обробляє події, що виникають при редагуванні інформації про пацієнта, такі як збереження змін до бази даних, перевірка введених даних та оновлення інтерфейсу.
- loginController.java: контролер для вікна авторизації. Обробляє введення логіну та пароля, перевіряє правильність введених даних і здійснює процес аутентифікації користувача.
- patientController.java: контролер для основного вікна роботи з пацієнтами. Він відповідає за відображення списку пацієнтів у таблиці, додавання нових записів, фільтрацію даних, а також за зміну або видалення існуючих записів.

3. model

Ця директорія містить класи моделей, які відповідають за обробку даних і взаємодію з базою даних.

- database.java: клас, що реалізує взаємодію з базою даних. Він містить методи для підключення до бази даних, виконання SQL-запитів для додавання, оновлення та видалення даних.

- patient.java: модель пацієнта, що описує структуру даних про пацієнта (ім'я, дата народження, діагноз тощо). Цей клас використовується для зберігання та обробки інформації про конкретного пацієнта.

4. resources

Ця директорія містить файли ресурсів, зокрема стилі CSS для темної та світлої теми інтерфейсу.

- dark-theme.css: стиль для темної теми інтерфейсу користувача. У файлі описані кольори фону, тексту та елементів інтерфейсу, що використовуються в темній темі.
- light-theme.css: стиль для світлої теми інтерфейсу користувача. Цей файл визначає кольорову палітру для світлої теми програми, забезпечуючи зручне використання в умовах хорошого освітлення.

5. view

У цій директорії знаходяться файли FXML, які описують інтерфейс користувача. Вони містять XML-код для створення вікон, кнопок, полів вводу та інших елементів інтерфейсу.

- editPatientWindow.fxml: FXML файл для вікна редагування пацієнта. Описує елементи інтерфейсу для редагування даних пацієнта, такі як текстові поля для введення імені лікаря, діагнозу, телефону, а також кнопки для збереження змін.
- loginView.fxml: FXML файл для вікна авторизації. Містить елементи для введення логіну та пароля, а також кнопку для входу в систему.
- patient_view.fxml: FXML файл для головного вікна програми, в якому відображається список пацієнтів. Тут знаходяться таблиці для перегляду інформації про пацієнтів, поля для фільтрації та кнопки для додавання, редагування та видалення записів.

Взаємодія компонентів

- контролери обробляють події, що відбуваються в вікнах (описаних у FXML файлах). Наприклад, PatientController.java відповідає за обробку

введених даних у patient_view.fxml (наприклад, додавання нового пацієнта або фільтрація списку).

- моделі (як-от Patient.java та Database.java) забезпечують доступ до даних, які зберігаються в базі даних. Коли користувач додає нового пацієнта через інтерфейс, дані передаються в модель, де зберігаються в базі даних.
- ресурси (стилі CSS) дозволяють змінювати вигляд інтерфейсу в залежності від обраної теми. Це забезпечує гнучкість і зручність використання програми в різних умовах.

Таким чином, кожна частина застосунку виконує свою конкретну роль, взаємодіючи з іншими компонентами для створення цілісного і функціонального продукту.

4.2Структура бази даних

У застосунку використовується реляційна база даних для зберігання інформації про пацієнтів. Основна таблиця, що використовується для зберігання даних пацієнтів, називається patients. Ось опис її структури:

1. таблиця patients

Таблиця patients зберігає всю необхідну інформацію про пацієнтів, їх лікарів та медичні дані. Її структура виглядає наступним чином:

```
CREATE TABLE IF NOT EXISTS patients (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    doctor_name TEXT NOT NULL,
    doctor_specialty TEXT NOT NULL,
    visit_date DATE NOT NULL,
    name TEXT NOT NULL,
    birth_date DATE NOT NULL,
    phone TEXT NOT NULL,
    diagnosis TEXT NOT NULL,
    doctor_conclusion TEXT NOT NULL
);
```

2. опис полів таблиці:

- id (INTEGER PRIMARY KEY AUTOINCREMENT): унікальний ідентифікатор запису пацієнта. Це поле є основним ключем таблиці і автоматично інкрементується при додаванні нових записів.
- doctor_name (TEXT NOT NULL): ПІБ лікаря, який обслуговує пацієнта. Це обов'язкове поле.
- doctor_specialty (TEXT NOT NULL): спеціальність лікаря, який обслуговує пацієнта.Це обов'язкове поле.
- visit_date (DATE NOT NULL): дата візиту пацієнта до лікаря. Це обов'язкове поле, яке фіксує дату прийому.
- name (TEXT NOT NULL): ПІБ пацієнта. Це обов'язкове поле.
- birth_date (DATE NOT NULL): дата народження пацієнта. Це обов'язкове поле, яке допомагає визначити вік пацієнта.
- phone (TEXT NOT NULL): номер телефону пацієнта. Це поле також є обов'язковим і дозволяє зв'язатися з пацієнтом.
- diagnosis (TEXT NOT NULL): діагноз пацієнта, поставлений лікарем. Це обов'язкове поле, що містить медичну інформацію про стан пацієнта.
- doctor_conclusion (TEXT NOT NULL): заключення лікаря щодо стану пацієнта після візиту. Це поле обов'язкове і містить медичні рекомендації або висновки щодо подальшого лікування.

3. особливості структури таблиці:

- нормалізація: структура таблиці є достатньо нормалізованою, адже всі поля мають чітке призначення, і відсутні зайві залежності. Для подальшого розширення, можна було б додавати нові таблиці (наприклад, таблиця для лікарів або діагнозів), однак в даній реалізації все об'єднано в одну таблицю для простоти та зручності.
- індекси та пошук: у таблиці немає додаткових індексів, окрім основного індексу для поля id. Це може бути доповнено в майбутньому для прискорення пошукових запитів (наприклад, індексація за полями doctor_name, name, visit_date).

- валідація даних: усі поля в таблиці є обов'язковими для заповнення (не допускається значення NULL), що гарантує збереження важливих даних про пацієнтів.

4. процес взаємодії з таблицею:

- додавання записів: кожен новий пацієнт, який додається через інтерфейс програми, створює новий запис у таблиці patients. Для цього використовуються SQL запити на вставку даних.
- редагування та оновлення: користувач може редагувати інформацію про пацієнтів. Оновлення відбувається за допомогою SQL запитів на оновлення записів, де фільтрація за id дозволяє змінювати лише відповідні дані.
- видалення записів: у разі необхідності, користувач може видаляти записи про пацієнтів. Для цього використовується SQL запит на видалення, який також базується на унікальному id.
- пошук і фільтрація: для пошуку пацієнтів за різними критеріями (наприклад, за ім'ям пацієнта або діагнозом), використовується SQL запит на вибірку даних з відповідними умовами.

5. забезпечення цілісності даних:

за допомогою NOT NULL обмежень гарантується, що всі важливі поля будуть заповнені, а також мінімізується ймовірність помилок при взаємодії з даними.

4.3Опис початкового вікна зі входом

Початкове вікно відповідає за вхід користувача в систему рис. 4.1. Воно з’являється одразу після запуску програми і перевіряє, чи правильно введені логін та пароль. Якщо дані правильні, користувач потрапляє в основну частину програми.

У самому вікні є два поля: для імені користувача та для пароля. Щоб полегшити введення пароля, я додав кнопку, яка дозволяє тимчасово показати, що саме користувач набирає. Це зручно, бо іноді можна випадково натиснути

не ту клавішу, і без цієї функції важко помітити помилку. Натискаючи на іконку "око", можна перемикатися між прихованим і видимим паролем.

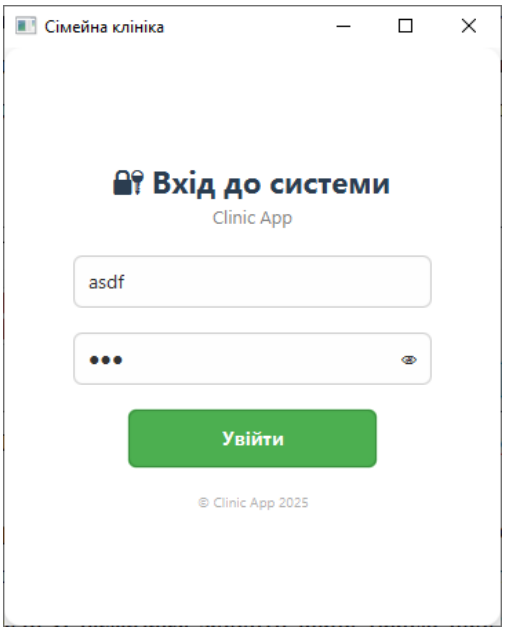

Рисунок 4.1 — Початкове вікно зі входом

Також я зробив так, щоб вхід можна було підтвердити не лише кнопкою, а й просто натиснувши Enter — це дрібниця, але робить користування зручнішим. Якщо логін або пароль введені неправильно, програма одразу повідомляє про це у вигляді спливаючого вікна з поясненням, що зображено на рис. 4.2.

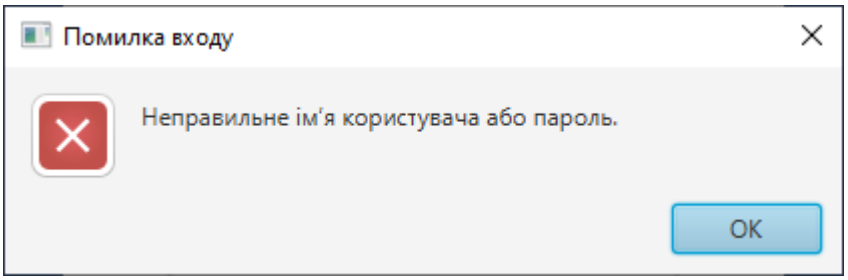

Рисунок 4.2 — Помилка, що видається при неправильному логіні або паролю

Весь інтерфейс зроблено простим і приємним: світлі кольори, заокруглені кути, легка тінь — усе для того, щоб вигляд був сучасний і не перевантажував очі. Я намагався зробити вікно таким, щоб ним було легко користуватись і щоб воно виглядало професійно.

4.4Опис основного вікна програми

Основне вікно програми, зображене на рис. 4.3, є центральним елементом інтерфейсу користувача після успішної авторизації. Це вікно надає доступ до всіх ключових функцій, необхідних для роботи з даними пацієнтів і управління медичними записами.

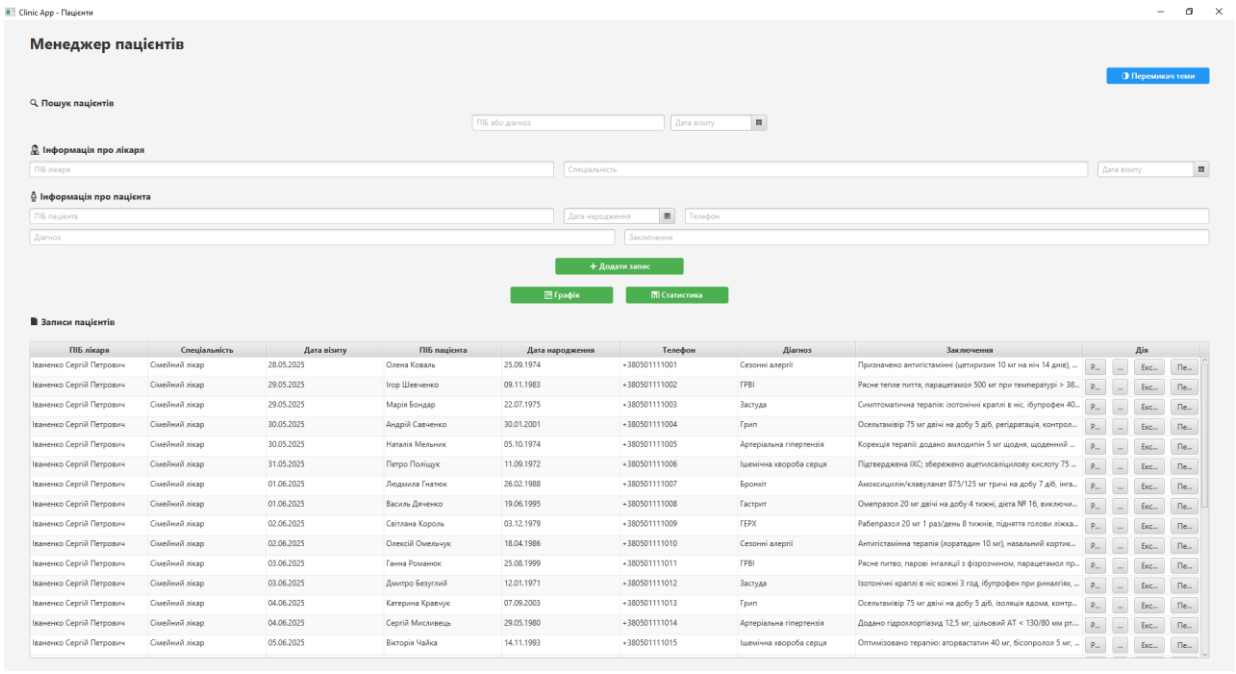

Рисунок 4.3 — Основне вікно програми

Основне вікно програми має зручний розмір і виконано у стильному та функціональному дизайні. У верхній частині вікна розташовані основні елементи навігації, включаючи перемикач теми, що дозволяє користувачеві змінювати зовнішній вигляд інтерфейсу між світлою та темною темою. Цей перемикач знаходиться у верхньому правому куті і реалізовано за допомогою ToggleButton з FXML. Користувач може натискати на кнопку для активації відповідної теми, що робить користування програмою комфортним в будь-яких умовах освітлення.

Для зміни теми в інтерфейсі застосовується метод toggleTheme в контролері PatientController.java, який виконує перемикання між двома стилями (світлим і темним).

Цей код дозволяє динамічно змінювати стилі без перезавантаження вікна, що робить інтерфейс зручним і сучасним. Таке рішення особливо важливе для

користувачів, які працюють в різних умовах освітлення, адже темна тема може знизити навантаження на очі в умовах слабкого освітлення.

Стилізація для світлої на та темної теми на рис. 4.5 розміщується в окремих CSS файлах, light-theme.css та dark-theme.css, що дозволяє зберегти чистоту коду та легко змінювати вигляд програми за допомогою змін стилів.

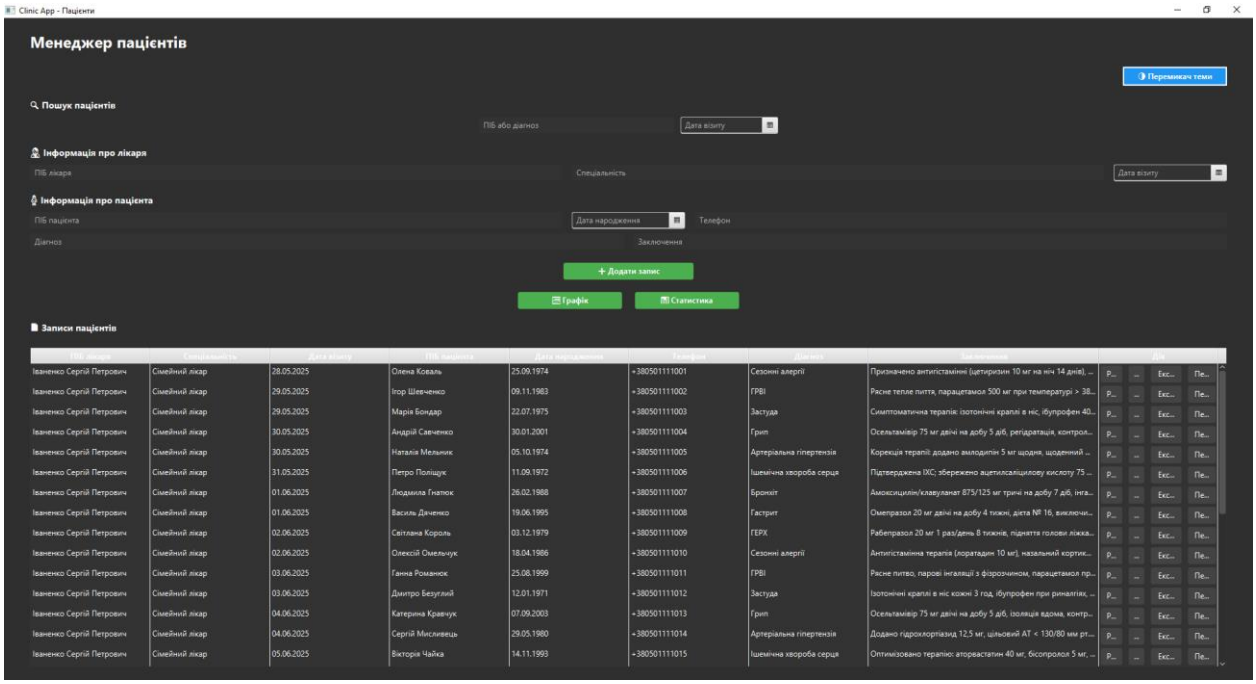

Рисунок 4.5 — Приклад темної теми

4.5Елементи пошуку

У центральній частині основного вікна знаходиться поле для пошуку пацієнтів. Це поле дає змогу користувачу швидко знаходити записи пацієнтів за іменем або діагнозом, що значно полегшує роботу медичних працівників, особливо коли в базі міститься велика кількість даних.

Поле для пошуку реалізовано за допомогою TextField в FXML, яке дозволяє користувачеві вводити текст для пошуку. Це поле зв’язано з відповідним контролером PatientController.java, де обробляється введений текст.

У цьому коді, як тільки користувач вводить текст у поле пошуку, метод filterPatients викликається для відображення результатів, що містять введений текст в імені або діагнозі пацієнта. Пошук виконується по всіх записах у

списку allPatients, і якщо дані відповідають умовам пошуку, вони додаються до таблиці patientTable для відображення.

Це забезпечує ефективний та швидкий доступ до необхідної інформації, що особливо важливо в медичних закладах, де оперативність є ключовою. Завдяки такому пошуку медичний персонал може за лічені секунди знайти потрібний запис серед великої кількості даних.

Приклад вдалого пошуку зображено на рис. 4.6

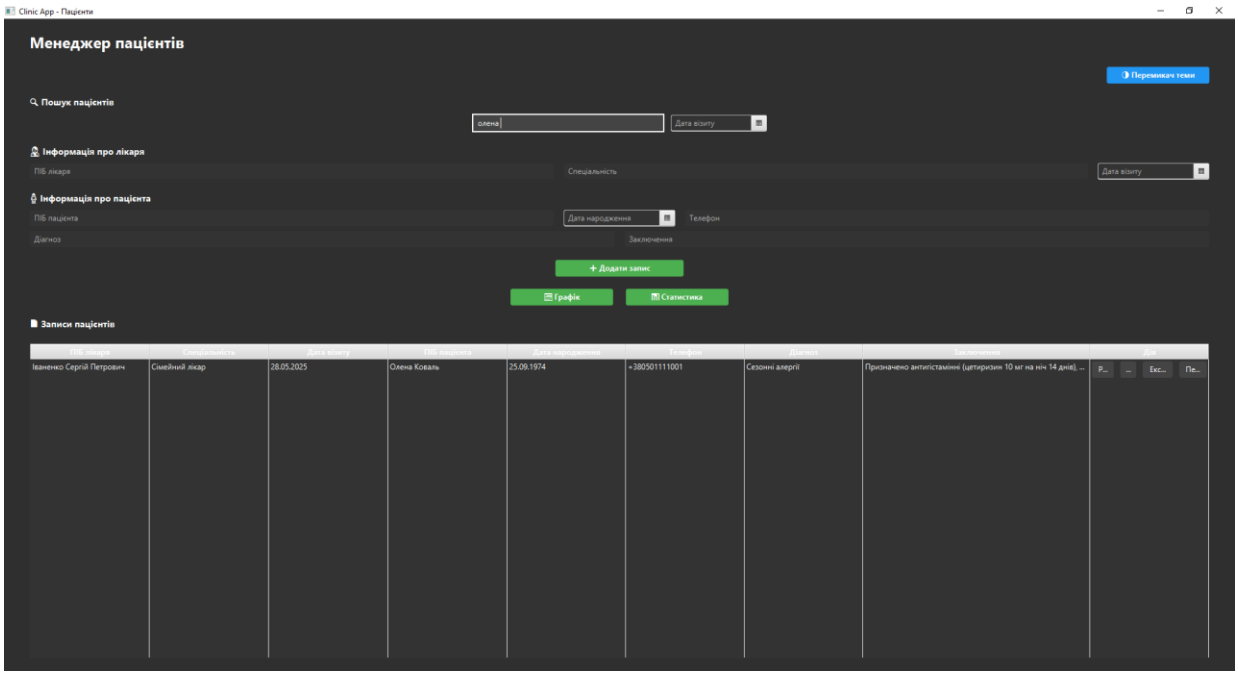

Рисунок 4.6 — Приклад пошуку за прізвищем пацієнта

4.6Форма для додавання нового запису

Нижче за полем пошуку в основному вікні розташована форма для додавання нового запису про прийом (рис. 4.7). Ця форма дозволяє медичному персоналу швидко вводити важливу інформацію про пацієнта, що забезпечує ефективне ведення бази даних.

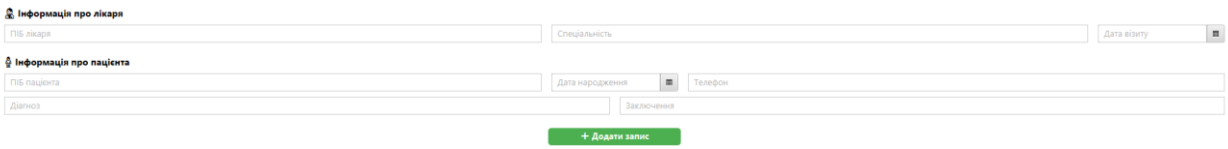

Рисунок 4.7 — Форма для введення нового запису

Форма зроблена просто й зрозуміло, щоб будь-хто — навіть користувач без технічного досвіду — міг без зайвих труднощів нею користуватись.

Уся форма розділена на дві частини: спочатку вводяться дані про лікаря, потім — про пацієнта. Для зручності кожна частина підписана. Це допомагає одразу зрозуміти, де і що потрібно заповнити. Окрім того, всередині кожного поля є підказки, які пояснюють, яку саме інформацію очікується ввести. Завдяки цьому немає потреби звертатися до інструкції — усе інтуїтивно зрозуміло.

Поля, які потрібно заповнити, охоплюють такі дані:

1. ім’я лікаря — повне ім’я лікаря, який проводить прийом.
2. спеціальність — напрямок, у якому працює лікар (наприклад, терапевт, невропатолог тощо).
3. дата візиту — коли саме пацієнт приходив на прийом.
4. ім’я пацієнта — його повне ім’я.
5. дата народження — щоб було зрозуміло, скільки пацієнту років.
6. номер телефону — на випадок, якщо потрібно зателефонувати.
7. діагноз — коротка інформація про те, з чим звернувся пацієнт.
8. заключення лікаря — підсумок або висновок після огляду.

Коли всі поля заповнені, потрібно натиснути кнопку «Додати запис». Якщо щось забуто — програма попередить і не дасть зберегти порожні поля. Це зроблено для того, щоб у базі не залишалась неповна інформація. Якщо ж усе заповнено правильно, програма збереже запис і,результат показано на рис. 4.8, якщо потрібно, запропонує створити ще один — наприклад, на повторний візит.

Загалом, ця форма проста, зручна і нічого зайвого в ній немає. Вона економить час і зменшує кількість помилок під час роботи з пацієнтами. Усе зроблено так, щоб користуватись було легко і приємно — навіть тим, хто не працює з комп’ютерами щодня.

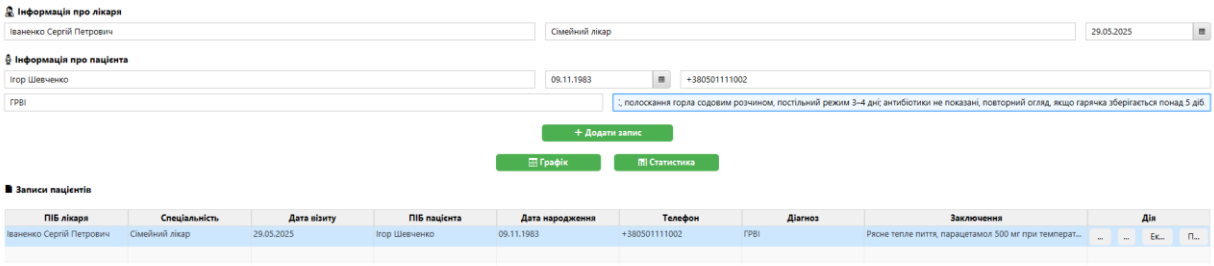

Рисунок 4.8 — Успішне додавання нового запису

4.7Таблиця з даними пацієнтів

У нижній частині основного вікна програми розміщується таблиця (рис. 4.9), яка відображає список усіх пацієнтів, що збережені в базі даних. Таблиця містить важливі стовпці для перегляду детальної інформації про пацієнтів та можливість управління цими записами. Серед стовпців:

1. ПІБ лікаря — відображає ім’я лікаря, який проводив прийом.
2. спеціальність лікаря — показує спеціальність лікаря.
3. дата візиту — відображає дату прийому пацієнта.
4. ПІБ пацієнта — надає інформацію про пацієнта.
5. дата народження пацієнта — відображає дату народження пацієнта.
6. телефон — контактний номер пацієнта.
7. діагноз — медичний діагноз пацієнта.
8. заключення лікаря — заключення лікаря за результатами прийому.
9. дія — стовпець для редагування, видалення, експорту або детального перегляду записів.

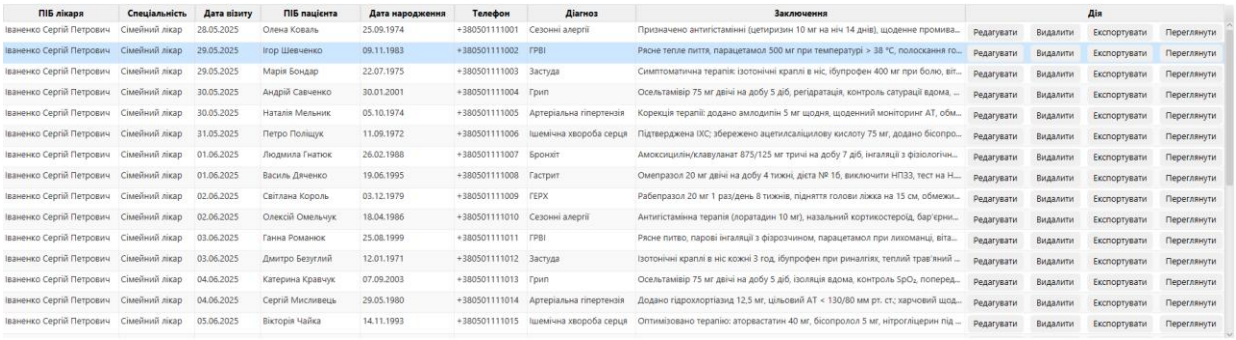

Рисунок 4.9 — Таблиця з даними пацієнтів

Ця таблиця дозволяє користувачам швидко знаходити та переглядати інформацію про пацієнтів. Вона реалізована за допомогою TableView в JavaFX.

Кожен запис пацієнта представляє окремий рядок таблиці, і користувач може вибрати будь-який рядок для подальшої роботи.

Ця таблиця складається з кількох стовпців, кожен з яких відповідає за відображення певної інформації. Крім того, є стовпець "Дія"(рис. 4.10), в якому розміщені кнопки для редагування, видалення, експортування та детального перегляду запису. Користувач може натискати ці кнопки, щоб виконати потрібні операції з записами пацієнтів.

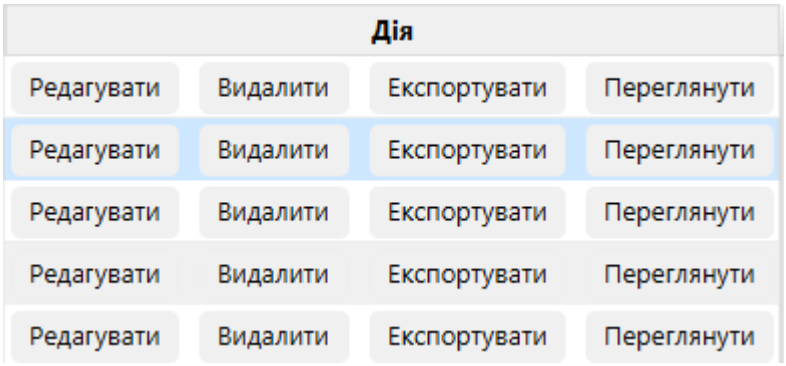

Рисунок 4.10 — Стовпець «Дія»

Обробка кнопок редагування, видалення, експортування та детального перегляду запису відбувається в контролері PatientController.java, де реалізовані відповідні методи.

Цей код обробляє натискання кнопок "Редагувати" та "Видалити". У випадку натискання кнопки "Редагувати" відкривається вікно для редагування вибраного запису, а при натисканні на "Видалити" запис видаляється з бази даних і таблиця оновлюється.

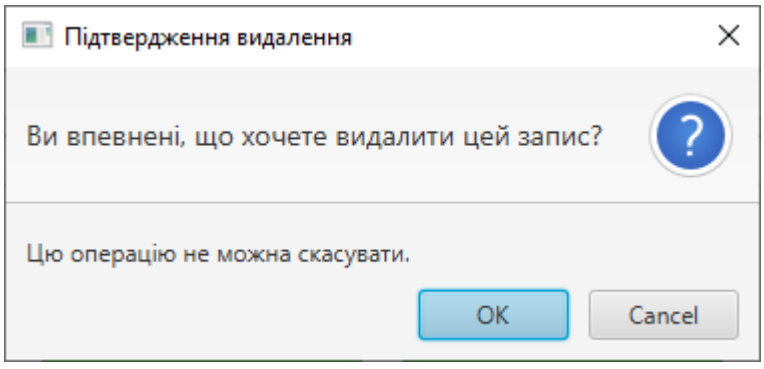

Рисунок 4.11 — Запит на підтвердження видалення запису

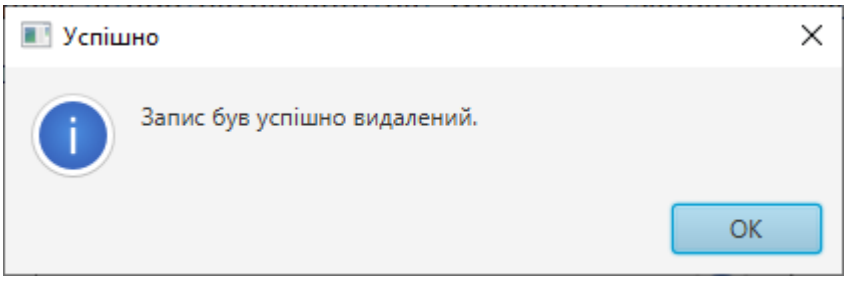

Рисунок 4.12 — Повідомлення про успішне видалення запису

4.8Вікно редагування запису

При натисканні на кнопку "Редагувати" в основному вікні, відкривається вікно для редагування, яке зображено на рис. 4.13 вибраного запису пацієнта. Це вікно дає змогу користувачам змінювати дані пацієнта, зокрема ім’я, дату народження, діагноз та інші важливі відомості. Воно включає форму, що відображає поточні значення для обраного пацієнта, даючи можливість змінити їх і зберегти нові значення в базі даних.

Структура вікна редагування:

форма, яка містить поля для введення даних:

- ім’я лікаря
- спеціальність лікаря
- дата візиту
- ім’я пацієнта
- дата народження пацієнта
- номер телефону
- діагноз
- заключення лікаря

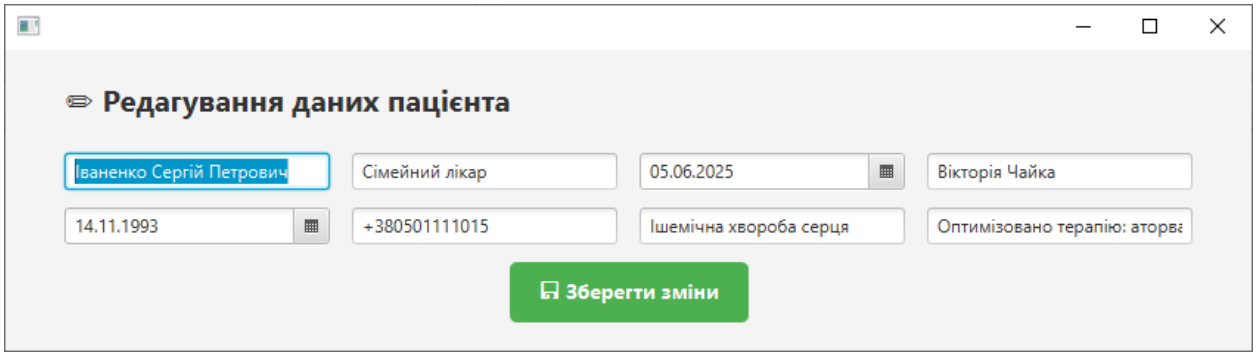

Рисунок 4.13 — Вікно редагування запису

Ці поля заповнюються поточними значеннями з вибраного запису пацієнта. Користувач може змінити будь-яке з цих полів.

Кнопка "Зберегти зміни".

Після редагування всіх необхідних полів, користувач може натискати кнопку "Зберегти зміни", щоб зберегти оновлену інформацію в базі даних. Ця кнопка активує метод контролера для оновлення запису в базі.

Опис функціоналу вікна редагування

Після того, як користувач натискає кнопку "Редагувати" в основному вікні, викликається метод, який відкриває вікно редагування.

Вікно редагування дозволяє ефективно оновлювати записи пацієнтів, надаючи простий та інтуїтивно зрозумілий інтерфейс для медичних працівників. Користувач може змінювати будь-яку інформацію, пов’язану з пацієнтом, і зберігати ці зміни в базі даних.

Таким чином, таблиця з даними пацієнтів є важливим елементом інтерфейсу, який надає зручний спосіб для перегляду, редагування та видалення записів.

Результат редагування зображено на рис. 4.14

Спеціальність лікаря	Дата візиту	ПІБ пацієнта
Кардіолог	2025-04-24	Іваненко Сергій

Спеціальність лікаря	Дата візиту	ПІБ пацієнта
Кардіолог	2025-04-29	Іваненко Сергій

Рисунок 4.14 — До та після редагування

4.9Експорт запису у PDF—документ

У програмі реалізована можливість експортувати запис про прийом пацієнта у PDF-документ, приклад звіт зображено на рисунку 4.15. Ця функція дозволяє швидко зберегти всю інформацію про візит у зручному вигляді — наприклад, щоб видати пацієнту офіційний висновок, надіслати його електронною поштою або просто зберегти у себе на комп’ютері.

Інформація про візит

Дані пацієнта

Ім'я:	Вікторія Чайка
Дата народження:	14.11.1993
Телефон:	+380501111015

Інформація про візит

Лікар:	Іваненко Сергій Петрович
Спеціальність:	Сімейний лікар
Дата візиту:	05.06.2025
Діагноз:	Ішемічна хвороба серця
Висновок:	Оптимізовано терапію: аторвастатин 40 мг, бісопролол 5 мг, нітрогліцерин під язик при болю; заплановано тредміл-тест через 3 місяці.

Дата створення: 04.06.2025 11:15

Дякуємо за довіру!

Рисунок 4.15—Приклад PDF-звіту

Користуватись цією функцією дуже просто. У таблиці, де відображаються всі записи пацієнтів, біля кожного з них є кнопка «Експортувати». Коли користувач натискає на неї, відкривається звичайне вікно збереження файлу. Програма автоматично пропонує назву документа, яка формується на основі імені пацієнта. Користувач може її змінити, або просто натиснути «Зберегти» — і PDF буде створено.

У самому файлі — уся основна інформація про візит. Спочатку вказуються дані пацієнта: його ім’я, дата народження, номер телефону. Потім — хто проводив прийом, тобто ім’я лікаря, його спеціальність і дата візиту. Після цього йде медична частина — діагноз і висновок, який лікар зробив під час консультації. В самому кінці додається дата створення файлу і короткий напис-подяка для пацієнта.

Документ виглядає охайно і просто: текст розділено на блоки, є підписи, все легко читається. Немає нічого зайвого, але вся суть — на місці. Це не просто технічний файл, а справді зрозумілий і зручний медичний документ.

Щоб забезпечити коректне відображення тексту, в тому числі українською мовою, я використав бібліотеку iText і додав підтримку кириличного шрифту. Це дає змогу створювати документ з українськими літерами, навіть якщо на комп’ютері немає відповідних системних шрифтів.

Загалом, ця функція дає змогу в кілька кліків створити акуратний документ з усією інформацією про прийом. Нічого не потрібно копіювати вручну — усе підтягується автоматично. Це зручно і для лікаря, і для пацієнта. І саме завдяки такій дрібній, але продуманій функції програма стає ближчою до реальної роботи в медичному закладі.

4.10 Вікно детального перегляду запису

У програмі передбачена можливість відкривати кожен запис про прийом у розгорнутому вигляді, приклад на рис. 4.16. Це зроблено для того, щоб користувач міг швидко переглянути всю пов’язану з візитом інформацію без необхідності прокручувати таблицю або шукати потрібне серед загального списку.

Рисунок 4.16—Приклад детального перегляду запису

Функція працює дуже просто. Біля кожного запису в таблиці є кнопка «Переглянути». Коли користувач натискає на неї, відкривається окреме вікно, де показано всі дані, які стосуються конкретного прийому. Інтерфейс цього вікна зроблений максимально зручним: жодних зайвих елементів, лише те, що потрібно — у зрозумілому вигляді.

У цьому вікні користувач бачить ім’я пацієнта, дату його народження, контактний номер, а також всю інформацію по візиту: хто лікар, яка його спеціальність, коли саме був прийом, який поставлено діагноз і яке заключення

залишив лікар. Усе структуровано та логічно розміщено, тому орієнтуватися дуже просто.

На відміну від основної таблиці, де більшість інформації стисло подано в один рядок, тут дані відображаються повністю. Наприклад, якщо заключення лікаря було довгим, воно може бути обрізане в таблиці, але в цьому вікні видно весь текст. Це особливо корисно, коли потрібно уважно перечитати рекомендації лікаря або швидко уточнити деталі візиту — без потреби відкривати редагування або експортувати PDF.

Форма перегляду — це лише для читання. Дані не можна змінити або випадково видалити, тому користувач може не хвилюватися, що щось зіпсує. Якщо ж потрібно внести зміни, існує окрема функція редагування.

Цей додатковий перегляд значно підвищує зручність користування. Особливо коли база вже містить десятки або сотні записів — відкривши такий перегляд, можна за кілька секунд зорієнтуватися в суті конкретного візиту, не витрачаючи часу на пошук дрібних деталей у таблиці.

По суті, це простий, але дуже корисний інструмент, який робить роботу з даними більш гнучкою та приємною. Усе працює швидко, відкривається в окремому вікні, не блокує основну частину програми й дозволяє одразу повернутись до роботи з таблицею.

4.11 Вікно графік-візитів

У програмі є окрема можливість подивитися графік візитів, рис. 4.17 — тобто всі заплановані прийоми, зібрані в одному вікні. Це зроблено для того, щоб не шукати вручну кожен запис у таблиці, а одразу побачити, хто і коли приходить, і які лікарі працюють у той чи інший день.

Графік відкривається після натискання на кнопку «Графік». Відкривається нове вікно, в якому зручно зібрано всі візити. Дані відображаються у вигляді списку або умовного календаря — з розділенням по датах. Там чітко видно: на який день призначено прийом, хто саме приходить, до якого лікаря, з яким діагнозом або з якою метою.

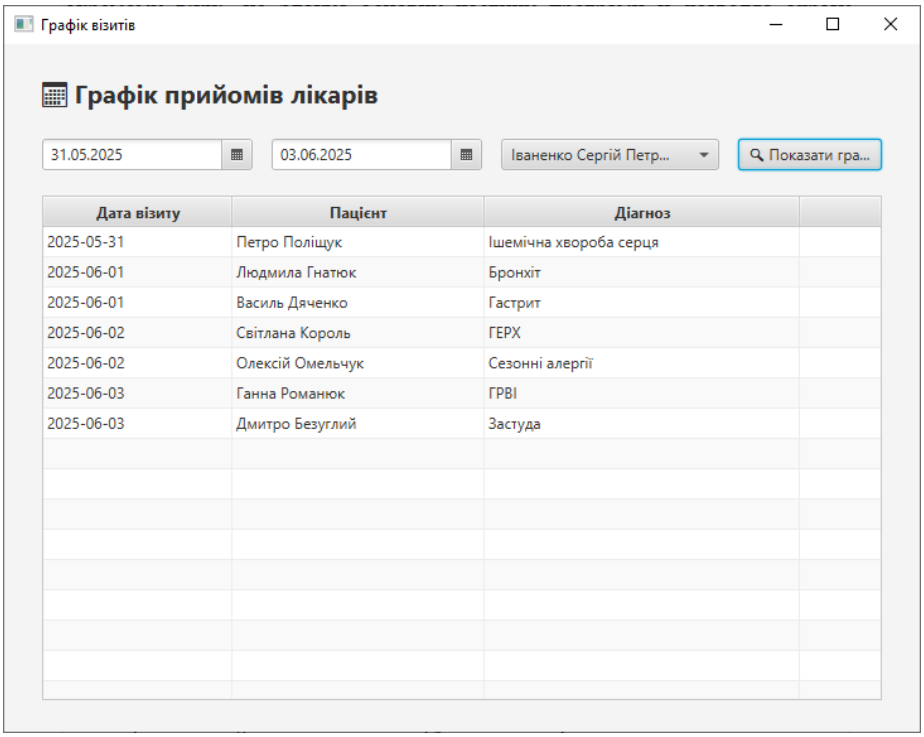

Рисунок 4.17—Графік візитів

Це вікно нічого не редагує — воно просто дозволяє переглянути ситуацію по прийомах. Якщо потрібно щось змінити, наприклад, перенести дату чи виправити дані — це робиться вже через головну таблицю. Але для перегляду — дуже зручно: не треба фільтрувати або шукати в загальному списку. Просто відкрив — і бачиш повну картину по днях.

Це дуже зручно, коли працюєш у клініці чи кабінеті, де щодня кілька пацієнтів, і потрібно орієнтуватися — хто буде далі, скільки ще прийомів залишилось, чи немає накладок. Так само можна швидко подивитися, чи є вільні дні або години для нового запису.

По суті, ця функція — це просто зручний перегляд усіх прийомів, але в більш наочному і впорядкованому вигляді. Вона не перевантажена зайвими деталями, і саме в цьому її перевага: все показано чітко, коротко і зрозуміло.

4.12 Статистика по записам

У програмі є окрема функція, яка дозволяє переглянути загальну статистику по пацієнтах. Вона відкривається в новому вікні, рис. 4.18, після натискання на кнопку «Статистика». Тут зібрана базова, але корисна

інформація, яка допомагає швидко оцінити ситуацію — скільки пацієнтів було, які діагнози трапляються найчастіше, і якого віку люди звертаються.

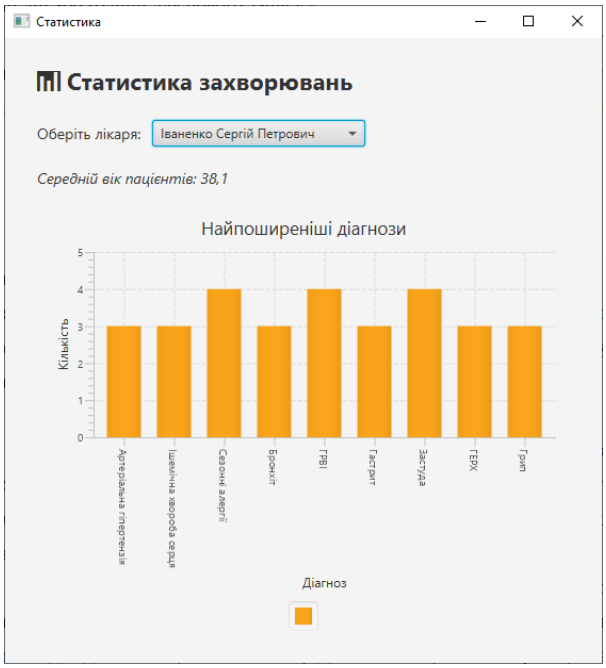

Рисунок 4.18—Вікно зі статистикою

У верхній частині вікна є список з іменами лікарів. Користувач може вибрати конкретного лікаря або залишити пункт «Усі лікарі», щоб подивитись загальну картину по всій клініці. Як тільки вибір змінюється — дані оновлюються автоматично.

Першим показником, який відображається, є середній вік пацієнтів. Програма сама рахує його на основі дат народження і показує значення з точністю до одного знаку після коми. Це дає загальне уявлення про те, з якою віковою групою частіше працює лікар.

Нижче — діаграма діагнозів. Вона показує, скільки разів зустрічався кожен діагноз у вибраного лікаря (або серед усіх). Це зроблено у вигляді стовпчиків, які легко порівнювати між собою. Назви діагнозів підписані знизу, а кількість звернень — зверху або по осі.

Вся ця інформація відображається спокійно, без зайвих деталей. Просто — щоб подивитися і швидко зрозуміти. Це може бути корисно як для лікаря, так і для адміністратора, який хоче бачити загальні цифри без потреби відкривати кожен запис окремо.

Функція не навантажує користувача — все працює автоматично. Обрав лікаря — і одразу бачиш, яка ситуація по ньому. Нічого вводити вручну не потрібно.

Загалом, ця функція створена не для складного аналізу, а для того, щоб у будь-який момент можна було глянути на загальну картину. Усе просто, наочно і корисно.

4.13 Взаємодія між елементами

Основне вікно надає користувачеві зручний інтерфейс для взаємодії з різними елементами програми. Наприклад, натискання кнопки "Додати запис" дозволяє зберегти введені дані з форми в базу даних, після чого таблиця автоматично оновлюється і відображає новий запис. Це забезпечує швидкий доступ до актуальної інформації.

Крім того, поле пошуку дозволяє користувачеві знайти потрібні записи в реальному часі. Як тільки введено запит, таблиця оновлюється, відображаючи тільки ті записи, що відповідають критеріям пошуку. Це значно спрощує роботу з великою кількістю даних.

Ще одна важлива функція — перемикач теми. Він дозволяє змінювати зовнішній вигляд інтерфейсу між світлою та темною темою. Користувач може без перезавантаження програми адаптувати інтерфейс під різні умови освітлення, що робить роботу з додатком комфортною в будь-який час доби.

4.14 Логіка обробки даних

Контролер PatientController.java відповідає за обробку всіх основних дій користувача в інтерфейсі. Це включає додавання нових записів пацієнтів, їх редагування, видалення та фільтрацію даних у таблиці за допомогою поля пошуку. Всі ці дії автоматично синхронізуються з базою даних, що забезпечує актуальність інформації в реальному часі.

Логіка роботи з даними гарантує, що кожна зміна, яку вносить користувач, коректно зберігається і відображається на екрані. Це дозволяє

лікарям швидко отримувати, додавати, редагувати або видаляти необхідні записи, що є важливим для ефективної роботи з пацієнтами.

4.15 Тестування програмного продукту

Після завершення реалізації основних функцій застосунку було проведено ручне тестування з метою перевірки його стабільності, правильності роботи та зручності використання. Основна увага приділялася тому, наскільки добре взаємодіють між собою компоненти системи і чи може користувач без проблем виконувати ключові дії — додавати, редагувати, видаляти та шукати записи пацієнтів.

Для перевірки роботи інтерфейсу та внутрішньої логіки застосунку використовувалися тестові дані. Було послідовно перевірено низку типових сценаріїв:

- авторизація — тестувалося введення як правильних, так і неправильних облікових даних, щоб переконатися в наявності коректної обробки помилок при вході.
- додавання запису — перевірялося, як поводить себе програма при повному або частковому заповненні форми. Окрема увага приділялася збереженню даних у базі після натискання кнопки "Додати запис".
- редагування — тестувалася можливість вибору пацієнта, відкриття відповідного вікна, внесення змін та їх збереження. Зміни перевірялися як у таблиці, так і безпосередньо в базі.
- видалення — перевірялася функція видалення запису з підтвердженням дії та подальшим оновленням таблиці без залишку видалених даних.
- пошук — тестувалося введення ключових слів для фільтрації даних у таблиці. Перевірялися варіанти пошуку як за іменем пацієнта, так і за діагнозом.
- зміна теми — перевірялася робота перемикача між світлою і темною темами без перезавантаження застосунку.

Усі тести проходили на середовищі Windows із встановленою Java 17. Під час перевірки були виявлені незначні помилки, пов’язані з валідацією введення — їх оперативно усунено.

У результаті тестування можна зробити висновок, що програмний продукт працює стабільно, виконує всі заплановані функції та готовий до практичного використання в умовах реального медичного закладу.

Усі результати тестування були занесені до табл. 4.1

Таблиця 4.1

Результати тестування основних функцій програмного застосунку

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Результат	Примітки
1	Вхід з правильними даними	Логін: doctor, Пароль: password	Перехід до основного вікна	Успішно	—
2	Вхід з неправильним паролем	Логін: doctor, Пароль: wrongpassword	Повідомлення про помилку	Успішно	—
3	Додавання нового пацієнта	Заповнені всі поля форми	Новий запис з’являється в таблиці	Успішно	—
4	Додавання з порожніми обов’язковими полями	Порожнє поле "Ім’я пацієнта"	Повідомлення про помилку	Успішно	Реалізована валідація
5	Редагування запису пацієнта	Змінено діагноз	Дані оновлюються в таблиці та базі даних	Успішно	—
6	Видалення пацієнта	Натиснуто кнопку "Видалити"	Запис зникає з таблиці та бази даних	Успішно	—
7	Пошук за ім’ям пацієнта	Петренко	Таблиця показує лише відповідні записи	Успішно	—
8	Перемикання теми	Світла/темна	Інтерфейс змінює тему без перезавантаження	Успішно	—

4.16 Можливості для розширення

Хоча розроблений програмний продукт вже виконує основні функції для роботи з медичними картками пацієнтів, є кілька напрямків, які можуть значно покращити програму і зробити її більш зручною та універсальною в реальних умовах медичної практики. Ось кілька ідей для подальшого розвитку:

- імпорт. Додавання функції імпорту даних дозволить швидко перенести записи з інших систем. Це буде корисно для медичних установ, що вже мають базу даних або хочуть зробити масове оновлення інформації[25].
- хмарне зберігання. Інтеграція з хмарними сервісами, такими як Google Drive чи Dropbox, дозволить зберігати резервні копії даних або забезпечити доступ до них з будь-якого пристрою. Це буде зручно для медичних установ, що працюють в умовах, де потрібна гнучкість у доступі до інформації[26].
- багатокористувацький режим і система доступу. Для клінік і лікарень, де працюють кілька медичних працівників, можна додати можливість налаштування прав доступу для різних користувачів. Наприклад, лікарі можуть мати доступ до редагування записів, медсестри — лише для додавання нових візитів, а адміністратори — до управління користувачами та статистики.
- інтеграція з календарем і нагадуваннями. Інтеграція з календарем дозволить лікарям додавати візити до свого робочого графіка та отримувати нагадування. Це допоможе уникнути пропусків важливих візитів або забутих зустрічей з пацієнтами.
- підтримка багатомовності. Для клінік, де працює різномовний персонал або є пацієнти з різних країн, важливо додати підтримку кількох мов в інтерфейсі. Це зробить програму доступнішою і зрозумілішою для всіх користувачів.
- аналіз даних і штучний інтелект. Впровадження технологій штучного інтелекту дозволить автоматично аналізувати дані пацієнтів, прогнозувати можливі ускладнення чи давати рекомендації щодо

лікування. Це стане важливим кроком у розвитку програми, яка може підтримувати лікарів у прийнятті рішень на основі медичних даних[26].

Ці можливості дозволять зробити програму більш зручною, гнучкою і адаптованою до потреб медичних установ різного масштабу. Вони також підвищать ефективність роботи медичних працівників і забезпечать більшу безпеку та зручність у роботі з даними пацієнтів.

4.17 Захист інформації

Захист даних у складному медичному додатку є критично важливою функцією, оскільки він оперує чутливою інформацією, такою як персональні дані пацієнтів, медичний анамнез, результати обстежень та історія хвороб. Будь-яке порушення конфіденційності або цілісності таких даних може мати серйозні правові, етичні та медичні наслідки. З метою забезпечення високого рівня безпеки в розробленому програмному забезпеченні було реалізовано низку ефективних заходів, спрямованих на запобігання несанкціонованому доступу, втраті або модифікації даних[27].

Зокрема, база даних зберігається локально в захищеному каталозі операційної системи, доступ до якого мають лише авторизовані користувачі. Додаток працює з ексклюзивними правами доступу до файлів, що унеможливлює втручання з боку сторонніх програм або процесів. Усі введені дані проходять обов’язкову валідацію, що дозволяє мінімізувати ризики технічних атак, таких як SQL-ін’єкції. Хоча наразі система не підтримує багаторівневу автентифікацію або шифрування, її архітектура дозволяє легко інтегрувати ці функції в майбутньому[28].

Таким чином, навіть на етапі базової реалізації були враховані ключові принципи інформаційної безпеки — конфіденційність, цілісність та доступність — що є необхідною умовою для роботи з медичною інформацією відповідно до сучасних стандартів і регламентів, таких як GDPR та HIPAA.

4.18 Захист даних під час зберігання

Інформація про пацієнтів зберігається в локальній базі даних SQLite, яка забезпечує безпечне та швидке зберігання даних. Незважаючи на те, що сама база даних не пропонує шифрування як базову функцію, для додаткового захисту цінних даних (зокрема, персональних даних пацієнтів) передбачається включення функції шифрування. Це забезпечить додатковий рівень безпеки для конфіденційних даних, що зберігаються локально. Крім того, локальне зберігання даних запобігає загрозам витоку даних через мережу.

4.19 Захист даних під час передачі

Оскільки додаток спочатку розроблявся з акцентом на автономну роботу без необхідності постійного інтернет-з’єднання, захист даних під час їх передавання через мережу наразі не є пріоритетним завданням. Уся взаємодія з даними відбувається локально, що значно знижує ризики перехоплення інформації під час передачі.

Проте в межах подальшого розвитку програмного забезпечення розглядається можливість впровадження функціоналу віддаленого доступу або синхронізації даних між кількома пристроями. У такому випадку передбачається використання сучасних криптографічних засобів, зокрема захищених мережевих протоколів, таких як HTTPS, для забезпечення шифрування трафіку та автентифікації сторін. Це дозволить зберегти високий рівень конфіденційності та захисту даних навіть у разі використання додатку в багатокористувацькому або хмарному середовищі.

4.20 Розрахунок економічного ефекту

У цьому підрозділі оцінюються ключові характеристики програмного забезпечення на етапі проєктування з метою вибору оптимальної стратегії

розробки, що враховує технічну та економічну доцільність. Особлива увага приділяється сумісності, функціональності та витратності рішень[29].

Для обґрунтування технічних рішень застосовано функціонально-вартісний аналіз, який дозволяє оцінити співвідношення між витратами на створення програмного продукту та корисністю його функцій. Метод дає змогу визначити ефективні варіанти реалізації та шляхи зниження витрат без втрати якості[29].

З метою структурування альтернативних технічних підходів була сформована морфологічна карта рис 4.19, що охоплює всі розглянуті конфігурації системи[29].

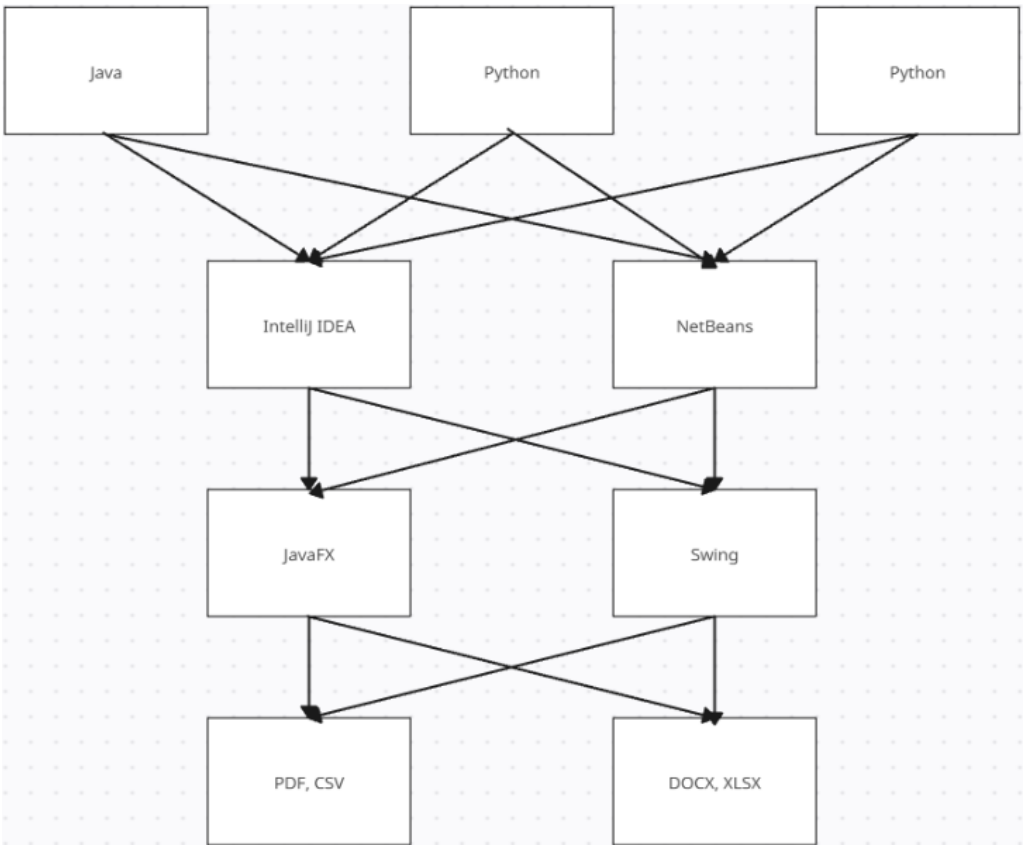

Рисунок 4.19— Морфологічна карта

Отже, найдоцільнішим є вибір першого варіанту реалізації програми, оскільки він продемонстрував найвищий техніко-економічний рівень. Проведений функціонально-вартісний аналіз дозволяє стверджувати, що серед двох відібраних альтернатив саме цей варіант є оптимальним за співвідношенням витрат і якості[29].

Цей варіант реалізації програмного продукту має такі параметри:

1. вибір мови програмування Java;
2. вибір середовища розробки IntelliJ IDEA;\
3. вибір інструментів для створення графічного інтерфейсу користувача JavaFX;
4. вибір формату збереження та експорту медичних даних PDF, CSV.

Даний варіант виконання програмного комплексу гарантує швидку та не ресурсомістку реалізацію вбудованого програмного забезпечення для проекту[29].

Витрати на реалізацію обраного варіанта програмного продукту складають 196 407.12 грн[29].

Висновок до розділу 4

Під час практичної частини роботи було розроблено повнофункціональний десктопний застосунок для ведення електронних карток пацієнтів приватної клініки. Реалізовано зручний та інтуїтивно зрозумілий графічний інтерфейс користувача з можливістю додавання, редагування, видалення та фільтрації записів. Усі введені дані зберігаються у локальній базі даних SQLite, що забезпечує автономність роботи програми без потреби в підключенні до мережі[35].

Особливу увагу було приділено зручності користування застосунком у різних умовах, зокрема реалізовано перемикач між темною та світлою темами. Інструменти, використані при розробці, такі як Java, JavaFX та SQLite, довели свою ефективність і забезпечили стабільну роботу програми. Структура застосунку є логічно організованою та легко розширюваною, що відкриває можливості для подальшого вдосконалення та масштабування функціоналу.

У результаті, створене програмне забезпечення відповідає поставленим задачам та може бути ефективно використане в роботі сімейного лікаря для швидкого доступу до даних пацієнтів та їх обліку. Також проаналізовано визначення економічного ефекту, зокрема його основні етапи та отримані результати[36].

РОЗДІЛ 5
УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ РОБОТИ

Розроблений десктопний застосунок для сімейного лікаря, створений із використанням платформи JavaFX, має на меті забезпечення зручного та ефективного інструменту для обліку пацієнтів у межах локальної медичної практики. У процесі реалізації було враховано основні потреби лікаря, який працює у приватній клініці або амбулаторії, де відсутня потреба у складних мережевих рішеннях чи централізованих медичних інформаційних системах.

5.1 Аналіз функціональності розробленого застосунку

Розроблений застосунок створений для щоденної роботи сімейного лікаря, який веде облік своїх пацієнтів у приватному кабінеті або невеликій амбулаторії. Усі функції в програмі зібрані навколо реальних потреб лікаря: щоб зручно зберігати, знаходити, переглядати й оновлювати медичну інформацію без зайвих кроків і складнощів. Інтерфейс залишився простим, зрозумілим і при цьому досить функціональним, щоб покрити все, що потрібно в повсякденній роботі.

Робота починається з авторизації. Це звичайне вікно з полями для логіну та паролю, яке відкривається одразу після запуску. Такий підхід допомагає обмежити доступ до даних і уникнути ситуацій, коли інформацію побачить стороння особа. Після входу користувач потрапляє в основне вікно.

Головна частина інтерфейсу — це таблиця з усіма записами пацієнтів. У ній зібрано всю основну інформацію: імена, дати народження, контактні номери, лікарі, діагнози, дати візитів, заключення тощо. Усе подано стисло, але зрозуміло. Якщо потрібно побачити більше — передбачено детальний перегляд запису, який відкривається у новому вікні. Там зібрано всю інформацію про конкретний прийом, зручно структуровано й подано у вигляді картки. Це дозволяє переглянути запис без ризику щось випадково змінити або видалити.

Щоб з великою кількістю записів було легко працювати, у програмі є пошук та фільтрація. Достатньо ввести частину імені або діагнозу — і список миттєво оновиться. Також є можливість фільтрувати за датою візиту, що особливо корисно при щоденній роботі. Пошук працює швидко, без затримок, навіть якщо база вже велика.

Кожен запис можна у будь-який момент відредагувати. Для цього достатньо натиснути кнопку «Редагувати» — відкривається знайома форма, де можна оновити контактні дані, діагноз, заключення, лікаря або інші поля. Якщо інформація стала неактуальною — запис можна просто видалити. Для додавання нового запису є окрема зручна форма з обов’язковими полями. Після збереження програма одразу може запропонувати створити ще один візит для того ж пацієнта — наприклад, якщо потрібна повторна консультація. Усі дані зберігаються в локальній базі автоматично, без додаткових натискань кнопок.

Застосунок працює на основі вбудованої бази даних SQLite. Це означає, що йому не потрібен інтернет або сервер — усі дані зберігаються локально. Це зручно, якщо лікар працює у віддаленому населеному пункті або просто не хоче залежати від мережі.

Також у програмі реалізовано експорт у PDF. За допомогою цієї функції можна створити охайний документ із повною інформацією про прийом: хто лікар, хто пацієнт, коли був візит, який діагноз і яке заключення. Документ створюється автоматично — користувачеві потрібно лише вибрати місце збереження. PDF можна роздрукувати, зберегти чи надіслати колезі або пацієнту.

Ще одна корисна можливість — перегляд графіку візитів. Це окреме вікно, де показано всі прийоми, відсортовані по датах. Завдяки цьому користувач бачить, хто коли приходив, які лікарі працюють у певні дні, і чи є ще вільні вікна для запису. Графік зручний у використанні й дозволяє швидко зорієнтуватися в навантаженні.

Крім того, програма дозволяє подивитися статистику по візитах. У вікні статистики можна вибрати будь-якого лікаря (або подивитися загальні дані) і побачити, який середній вік пацієнтів, з якими діагнозами звертаються

найчастіше, скільки пацієнтів було прийнято. Уся інформація відображається у вигляді зрозумілих графіків — ніяких складних налаштувань, усе оновлюється автоматично.

Для більшого комфорту користувача в застосунку також передбачено темну й світлу тему інтерфейсу. Можна перемикатися між ними в один клік — наприклад, щоб було зручно працювати в темному кабінеті або навпаки, при яскравому освітленні.

У результаті — програма охоплює всі основні функції, які реально використовуються в щоденній практиці. Вона дозволяє зручно додавати, переглядати, редагувати, фільтрувати, зберігати, планувати й аналізувати дані про пацієнтів. І все це — без складних налаштувань чи технічних знань. Саме тому застосунок може стати основним робочим інструментом для лікаря, який веде приватну практику або працює в невеликій клініці.

5.2Порівняння з існуючими аналогами

У процесі розробки програмного забезпечення було проведено ґрунтовний аналіз наявних рішень, які вже використовуються в медичній галузі Усі ці платформи мають спільну рису — вони орієнтовані на комплексне обслуговування великих медичних закладів і передбачають широкий спектр функцій. Здебільшого до їх можливостей входить ведення електронних медичних карток, формування графіків прийому, облік фінансових операцій, інтеграція з лабораторіями, виписка рецептів, генерація статистичних звітів тощо. Це потужні системи, які розраховані на роботу в межах лікарень, клінік або мереж медичних установ[30].

Однак така багатофункціональність часто супроводжується низкою суттєвих обмежень. По-перше, більшість із цих рішень є комерційними та потребують значних фінансових витрат — як на початкову ліцензію, так і на регулярну абонентську плату. По-друге, вони зазвичай вимагають складного налаштування та наявності серверної інфраструктури, а також кваліфікованих фахівців для адміністрування. Для невеликої приватної практики або індивідуального лікаря це може бути надмірним як з організаційної, так і з

фінансової точки зору. Крім того, інтерфейс таких систем часто є перевантаженим і не завжди інтуїтивно зрозумілим, особливо для користувачів, які не мають глибоких IT-знань[30].

На цьому фоні розроблений застосунок демонструє іншу філософію — він максимально орієнтований на простоту, автономність і швидкий доступ до основної інформації без зайвого функціонального навантаження. Застосунок не потребує встановлення сторонніх компонентів, серверів чи підключення до Інтернету — усе зберігається локально, що забезпечує як незалежність від зовнішніх чинників, так і високий рівень приватності. Це робить програму ідеальним рішенням для сімейного лікаря, який працює в умовах обмежених ресурсів, наприклад, у невеликій клініці чи приватному кабінеті в передмісті.

Ще однією вагомою перевагою є простий і зрозумілий інтерфейс. Усі елементи керування логічно згруповані, функції доступні буквально в кілька кліків, і це дозволяє сконцентруватися саме на роботі з пацієнтами, а не на освоєнні програмного забезпечення. Таким чином, на відміну від складних медичних платформ, які часто перевантажені функціоналом, наш застосунок є вузькоспеціалізованим, але водночас повністю задовольняє реальні потреби сімейного лікаря в щоденній практиці.

Загалом, порівняння з аналогічними продуктами дозволяє чітко побачити нішу, яку займає розроблене рішення. Це простий, легкий, доступний і автономний інструмент, що забезпечує базову, але вичерпну функціональність для ведення обліку пацієнтів, і може ефективно використовуватися в тих умовах, де використання великих медичних платформ є недоцільним або неможливим.

5.3Переваги розробленого рішення

Розроблений застосунок має низку переваг, які роблять його зручним і практичним інструментом для щоденної роботи сімейного лікаря. Одне з найважливіших — це простота у використанні. Інтерфейс програми інтуїтивно зрозумілий: усі поля логічно розміщені, підписані, розділені на блоки за змістом, що дозволяє легко зорієнтуватися навіть користувачу без технічної

підготовки. Меню не перевантажене, а загальний вигляд — мінімалістичний. Завдяки цьому увага зосереджується не на самому застосунку, а на роботі з пацієнтами.

Програма створена на основі JavaFX, тому має сучасний і приємний зовнішній вигляд. Інтерфейс підтримує темну та світлу теми — це зручно, якщо лікар працює в різних умовах освітлення або проводить прийоми у вечірній час. Перемикання режимів доступне в один клік. Крім того, застосунок коректно виглядає як на стандартному ноутбуці, так і на великому моніторі, тому проблем із масштабуванням чи відображенням не виникає.

Важливою перевагою є й те, що програма працює повністю автономно. Всі дані зберігаються локально — у базі даних SQLite, яка вбудована в сам застосунок. Це означає, що для роботи не потрібен інтернет або підключення до серверів. Програма ідеально підходить для лікарів, які працюють у сільській місцевості, на виїзді або в домашньому кабінеті. Крім того, такий підхід гарантує захист персональних даних — уся інформація залишається на пристрої лікаря.

Внутрішня структура застосунку побудована за принципом MVC (Model-View-Controller). Це означає, що проєкт легко підтримувати і за потреби розширювати. Наприклад, у майбутньому можна буде додати можливість друку медичних карток, перегляд календаря візитів, інтеграцію з лабораторними системами — і все це без переробки основної логіки. Це дає програмі запас гнучкості, що важливо, якщо медична практика з часом буде змінюватися або розширюватися[31].

Окремо варто згадати функцію експорту у PDF. Вона дозволяє швидко зберегти повну інформацію про візит у вигляді документа, який можна роздрукувати, зберегти в архів або передати колезі. Для цього не потрібно нічого налаштовувати — достатньо лише вибрати файл і місце для збереження. Така проста, але корисна функція значно полегшує документообіг.

У підсумку — застосунок поєднує зручність, незалежність від зовнішніх факторів, гнучкість у розвитку і простоту в роботі. Завдяки цьому він може стати надійним помічником для лікаря, який хоче організувати свою роботу ефективно, не витрачаючи час на технічні труднощі.

5.4 Недоліки та обмеження

Попри зручність і функціональність, під час використання застосунку помітно кілька обмежень, які важливо враховувати, особливо якщо йдеться про подальший розвиток проєкту.

Програма створена з розрахунком на одного користувача. Вона добре працює в умовах приватної практики, коли один лікар веде облік пацієнтів на своєму комп’ютері. Але в ситуаціях, де потрібно колективне використання — наприклад, у невеликій клініці або медичному центрі — бракує можливості спільного доступу до даних чи синхронізації між кількома пристроями.

Ще одне питання — це безпека. Застосунок має базовий захист: дані зберігаються локально, а вхід до програми здійснюється через авторизацію. Проте, якщо хтось отримає фізичний доступ до комп’ютера, він може отримати і доступ до бази, оскільки додаткових рівнів захисту не передбачено. Для чутливої медичної інформації цього може бути недостатньо.

Також варто згадати, що в поточній версії не реалізовано автоматичне резервне копіювання. Уся інформація зберігається в одному файлі, без дублювання чи створення архівної копії. У разі збою або втрати доступу до пристрою існує ризик повної втрати всіх записів. Для лікаря, який щодня працює з пацієнтами, це може бути критично.

Окрім цього, програма поки що не підтримує інтеграцію з іншими медичними системами — наприклад, електронними картками, лабораторіями чи державними сервісами. Це не заважає роботі в автономному режимі, але може стати обмеженням, якщо лікар захоче обмінюватися даними з іншими фахівцями або підключитись до єдиної системи.

Усі ці моменти не знижують цінність програми — вона вже зараз дає змогу ефективно працювати з пацієнтами. Але якщо ставити за мету її подальший розвиток, варто звернути увагу на багатокористувацький доступ, підвищення рівня захисту, створення резервних копій і можливість інтеграції з іншими сервісами. При цьому головне — зберегти простоту і зручність, які є основною перевагою застосунку.

Висновок до розділу 5

Загалом, результати реалізації проєкту демонструють успішне вирішення поставленої задачі. Було створено стабільний, надійний та зручний у використанні інструмент для сімейного лікаря, який дозволяє ефективно вести базовий електронний облік пацієнтів у повністю автономному режимі без потреби в постійному інтернет-з'єднанні. Інтерфейс програми є інтуїтивно зрозумілим, що мінімізує час на навчання та адаптацію користувачів. Застосунок забезпечує базові функції електронного документообігу, включно з додаванням, редагуванням, переглядом і пошуком медичних записів.

Проведений функціональний та порівняльний аналіз підтвердив доцільність створення такого рішення саме для малих приватних практик, де відсутня інфраструктура для впровадження складних та дороговартісних інформаційних систем. Розробка закриває потребу у простому, але ефективному інструменті, який підвищує продуктивність лікаря та покращує якість обслуговування пацієнтів у локальному масштабі.

Незважаючи на наявні обмеження, такі як відсутність мережевої синхронізації або резервного копіювання, застосунок має значний потенціал для подальшого вдосконалення. У перспективі можливе розширення функціональності, зокрема реалізація мережевої взаємодії між кількома користувачами, впровадження багаторівневого контролю доступу, шифрування медичних даних, а також створення автоматизованих систем резервного копіювання для підвищення надійності та безпеки збереженої інформації.

Таким чином, отримані результати не лише підтверджують актуальність обраного напряму, але й можуть бути використані як платформа для подальшої модернізації, масштабування або адаптації програмного забезпечення до потреб ширшого кола медичних фахівців — від лікарів загальної практики до вузькопрофільних спеціалістів.

ЗАГАЛЬНІ ВИСНОВКИ

У процесі виконання дипломної роботи було створено автономний десктопний застосунок для ведення електронних медичних карток пацієнтів, адаптований під потреби сімейного лікаря. Реалізоване рішення охоплює ключові аспекти щоденної медичної практики — від введення інформації про пацієнтів до зручного управління базою даних — із фокусом на автономність, безпеку та зручність використання.

Основні результати:

- розроблено архітектуру застосунку, що включає інтерфейс користувача (JavaFX), бізнес-логіку (Java) та зберігання даних (SQLite). Це забезпечило стабільну та швидку роботу навіть без доступу до Інтернету.
- створено інтуїтивний графічний інтерфейс, який враховує потреби лікарів — із можливістю додавання, редагування, пошуку та видалення пацієнтів, перемикання теми (світлої/темної), а також експорту звітів.
- забезпечено повноцінну взаємодію з базою даних, включаючи структуру таблиці, механізми валідації даних, перевірку на коректність введення та обробку типових помилок користувача.
- реалізовано модульну архітектуру, яка дозволяє незалежно розширювати та тестувати окремі компоненти програми.
- проведено всебічне тестування — модульне, інтеграційне та системне — що дозволило виявити й усунути помилки до впровадження.
- забезпечено захист даних пацієнтів, реалізовано валідацію полів введення, обмежено доступ до локальної бази, передбачено подальше впровадження автентифікації та шифрування даних.

Оцінні характеристики розробленого рішення:

- зручність використання — інтерфейс створено відповідно до реальних сценаріїв роботи лікаря, без перевантаження непотрібною функціональністю.
- безпека і відповідність практиці — збереження даних відбувається локально, із забезпеченням контролю доступу до бази.

- інноваційність — автономність та офлайн-режим є суттєвою перевагою для використання в умовах обмеженої інфраструктури.
- масштабованість — архітектура дозволяє інтегрувати нові модулі або підключати систему до інших медичних платформ.

Таким чином, реалізований застосунок забезпечує ефективне ведення електронних медичних записів на локальному рівні, підвищуючи якість та швидкість медичного обслуговування в умовах реальної практики.

Висновки

У дипломній роботі на тему «Програмний застосунок сімейного лікаря для управління медичними записами в приватній клініці приміського типу» було реалізовано повнофункціональний програмний продукт, орієнтований на сімейну медичну практику. Рішення відповідає сучасним вимогам до цифровізації охорони здоров’я, враховує обмеження існуючих систем і забезпечує автономну роботу без втрати функціональності.

Основні підсумки виконаної роботи:

- проведено аналіз існуючих аналогів (Medtech, Epic, OpenMRS) та визначено ключові проблеми: складність інтеграції, залежність від Інтернету, низька швидкодія.
- обґрунтовано вибір технологій (Java, JavaFX, SQLite), які забезпечили гнучкість, безпеку та зручність підтримки.
- реалізовано повний цикл роботи із медичними записами: введення, редагування, фільтрація, збереження, пошук, експорт.
- забезпечено високий рівень безпеки даних – через валідацію введення, обмеження доступу та готовність до майбутнього впровадження автентифікації.
- підготовлено технічну документацію та проведено пілотне впровадження в умовах наближених до реального використання.

Загалом, дипломна робота підтвердила досягнення поставленої мети — створення автономного, зручного та надійного програмного застосунку, який здатен суттєво підвищити ефективність щоденної роботи сімейного лікаря в умовах приватної медичної практики. Розроблене рішення не лише спрощує

ведення електронної медичної документації, а й забезпечує стабільний доступ до даних незалежно від наявності інтернет-з’єднання, що є особливо важливим у регіонах із нестабільною інфраструктурою.

Окрім практичної цінності, отримані результати мають потенціал для подальшого розвитку — як у вигляді вдосконалення функціональності, так і через масштабування на ширший сегмент медичних закладів. Таким чином, робота заклала фундамент для розвитку локальних медичних ІТ-рішень в Україні, орієнтованих на автономність, безпеку та доступність у використанні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CRMium – порівняння MIS та SRM у медичній галузі [Електронний ресурс]. – Режим доступу: https://crmium.com/uk/mis-vs-srm-avtomatyzacziya-prodazhiv-u-medychnij-galuzi/
2. Medtech Global – офіційний сайт [Електронний ресурс]. – Режим доступу: https://medtechglobal.com/
3. Epic Systems – офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.epic.com/
4. OpenMRS – офіційний сайт [Електронний ресурс]. – Режим доступу: https://openmrs.org/
5. BlueBrix – хмарне EHR без підключення до інтернету [Електронний ресурс]. – Режим доступу: https://bluebrix.health/blogs/cloud-based-ehr-offline-without-internet
6. Thinkitive – оптимізація продуктивності обміну даними EHR [Електронний ресурс]. – Режим доступу: https://www.thinkitive.com/blog/performance-optimization-for-high-volume-ehr-data-exchange/
7. Evergreens – методології розробки ПЗ [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ua/articles/software-development-metodologies.html
8. Foxminded – структурне програмування [Електронний ресурс]. – Режим доступу: https://foxminded.ua/strukturne-prohramuvannia/
9. Dan IT – об’єктно-орієнтоване програмування [Електронний ресурс]. – Режим доступу: https://dan-it.com.ua/uk/blog/chto-takoe-obektno-orientirovannoe-programmirovanie-principy-preimushhestva-i-nedostatki/
10. Мартін Р. С. Чистий код. – Київ: Вид. дім «Фабула», 2019
11. Znannya.org – шаблон Singleton [Електронний ресурс]. – Режим доступу: http://www.znannya.org/?view=patterns-singleton
12. Наік К., Трипаті П. Тестування та забезпечення якості програмного забезпечення. – Wiley, 2008
13. JetBrains Help – офіційна документація [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/help/idea/getting-started.html
14. OpenJFX Documentation – офіційний сайт [Електронний ресурс]. – Режим доступу: https://openjfx.io/openjfx-docs/

15. Java Database Connectivity — Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Java_Database_Connectivity
16. EPAM Campus – офіційний блог [Електронний ресурс]. – Режим доступу: https://campus.epam.ua/ua/blog/581
17. Досвід використання СУБД SQLite в інформаційних системах [Електронний ресурс]. – Режим доступу: https://dspace.nuph.edu.ua/bitstream/123456789/16940/1/221-229.pdf
18. Система для приватної клініки – приклад дипломної роботи [Електронний ресурс]. – Режим доступу: https://ena.lpnu.ua/items/353e9053-107b-4329-adb1-0e6b129ef6c3
19. ITVDN – порівняння C# та Java [Електронний ресурс]. – Режим доступу: https://itvdn.com/ua/blog/article/csharp_vs_java
20. Freeman E., Robson E. Head First Design Patterns. – O'Reilly Media, 2004
21. ISMS.online – ISO 27001: Annex A.9 – Access Control [Електронний ресурс]. – Режим доступу: https://www.isms.online/iso-27001/annex-a-9-access-control/
22. Hostpro – типи шифрування та алгоритми [Електронний ресурс]. – Режим доступу: https://hostpro.ua/wiki/ua/security/encryption-types-algorithms/
23. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Prentice Hall, 2017
24. AP-Impulse – SQLite JDBC для JavaFX: покрокова інструкція [Електронний ресурс]. – Режим доступу: https://www.ap-impulse.com/sqlite-jdbc-for-javafx-dbeaver-step-112/
25. Stack Overflow – export and import SQLite database [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions/56843045/export-and-import-sqlite-database
26. Google Cloud – Backup and Disaster Recovery [Електронний ресурс]. – Режим доступу: https://cloud.google.com/backup-disaster-recovery
27. HHS.gov – безпека інформації в охороні здоров’я [Електронний ресурс]. – Режим доступу: https://www.hhs.gov/hipaa/for-professionals/security/index.html
28. Foxminded – SQL-ін’єкції [Електронний ресурс]. – Режим доступу: https://foxminded.ua/sql-iniektsii/
29. Пашін В. п, Романов В. в, Єгорова Н. в Методичні вказівки до виконання економіко-організаційного розділу дипломних проектів (робіт) бакалаврів і спеціалістів для студентів Інституту прикладного системного аналізу з

дисципліни “Економіка та організація виробництва” для студентів. Київ : НТУУ “КПІ”, 2011.

30. Evergreens – медичні інформаційні системи [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ua/articles/medical-information-systems.html
31. Brander – шаблон MVC [Електронний ресурс]. – Режим доступу: https://brander.ua/technologies/mvc
32. SelectHub – порівняння EpicCare та OpenMRS [Електронний ресурс]. – Режим доступу: https://www.selecthub.com/ehr-software/epiccare-vs-openmrs/
33. AHRQ – EHR Usability Toolkit: Background Report [Електронний ресурс]. – Режим доступу: https://digital.ahrq.gov/sites/default/files/docs/citation/EHR_Usability_Toolkit_Background_Report.pdf
34. MDPI – A Comparative Study of Open-Source EHR Systems [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/2073-431X/13/2/41
35. Електронна бібліотека КПІ – дослідження щодо EHR систем [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/25b59e05-2f5b-481c-99b3-18cc5154d39c/content
36. Петренко О. М. Оцінка економічного ефекту впровадження електронних медичних систем // Журнал медичної інформатики. – 2021. – Т. 15, № 3. – С. 45–52.