

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ПІДТРИМКИ РОБОТИ ЛІКАРНІ

SOFTWARE TO SUPPORT

THE OPERATION OF THE HOSPITAL

Виконав(ла): студент(ка) 4 курсу, групи

КНТ-112сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

СКОРІН С.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ІЛ'ЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СКОРІНА Сергія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення підтримки роботи лікарні.
Software to Support the Operation of the Hospital

керівник проєкту (роботи) к.т.н., доцент, ІЛЬЯШЕНКО Матвій Борисович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ІЛІЯШЕНКО М.Б., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 28 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Сергій СКОРІН
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Матвій ІЛІЯШЕНКО
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
101 с., 3 табл., 38 рис., 3 дод., 18 джерел.

C#, VISUAL STUDIO, ЛІКАРНЯ, МЕДИЧНИЙ ЗАКЛАД, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СЕРЕДОВИЩЕ РОЗРОБКИ.

Об'єкт дослідження – процес розробки програмного забезпечення для підтримки функціонування медичних закладів.

Предмет дослідження – програмні засоби для підтримки функціонування медичних закладів.

Мета роботи – розробка програмного забезпечення для підтримки роботи лікарні.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для підтримки роботи лікарні. Створено програмне забезпечення для підтримки процесу функціонування лікарні. Проведено тестування розробленої програмної системи для підтримки процесу функціонування лікарні.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для підтримки роботи лікарні за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – підтримка функціонування медичних закладів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 101 pages, 3 tables, 38 figures, 3 appendixes, 18 sources.

C#, VISUAL STUDIO, HOSPITAL, MEDICAL INSTITUTION,
PROGRAMMING LANGUAGE, SOFTWARE, DEVELOPMENT
ENVIRONMENT.

The object of research is the process of developing software for supporting the functioning of medical institutions.

The subject of the research is software for supporting the functioning of medical institutions.

The purpose of this work is to develop the software to support the functioning of a hospital.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for supporting the functioning of a hospital has been designed. The software to support the functioning of a hospital has been developed. The developed software system to support the functioning of a hospital has been tested.

Conclusions. The goal of the work has been achieved. The software tools to support the functioning of a hospital using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – supporting the functioning of medical institutions.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для підтримки роботи лікарні	11
1.2 Аналіз програмного забезпечення для підтримки роботи лікарні	24
1.3 Висновки за розділом 1	34
2 Матеріали і методи	36
2.1 Вибір мови програмування	36
2.2 Вибір середовища розробки для створення програмного забезпечення для підтримки роботи лікарні	39
2.3 Висновки за розділом 2	41
3 Опис програми	42
3.1 Структура програмного забезпечення для підтримки роботи лікарні	42
3.2 Функціонування програмного забезпечення для підтримки роботи лікарні	44
3.3 Розробка бази даних програмного забезпечення для підтримки роботи лікарні	47
3.4 Проєктування інтерфейсу програмного забезпечення для підтримки роботи лікарні	50
3.5 Висновки за розділом 3	54
4 Експлуатація, тестування та експериментальне дослідження програми	56
4.1 Призначення й умови застосування програми	56
4.2 Характеристики програми для підтримки роботи лікарні	57
4.3 Інструкція по експлуатації програми	58
4.3.1 Звернення до програми	58
4.3.2 Вхідні й вихідні дані	58
4.3.3 Повідомлення	59

4.4 Виконання програмного забезпечення для підтримки роботи лікарні	59
4.5 Тестування програмного забезпечення для підтримки роботи лікарні	64
4.6 Висновки за розділом 4	64
Висновки	65
Перелік джерел посилань	68
Додаток А Технічне завдання	70
Додаток Б Фрагмент тексту програми	74
Додаток В Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

WCF – Windows Communication Facility;

БД – база даних;

МЗ – медичний заклад;

ПЗ – програмне забезпечення.

Вступ

Програмне забезпечення для підтримки роботи лікарень є надзвичайно актуальним, оскільки воно дозволяє автоматизувати ключові процеси в медичних установах. Завдяки сучасним програмним рішенням лікарі та адміністративний персонал можуть ефективніше управляти записами пацієнтів, контролювати наявність медичних препаратів і оптимізувати роботу медичних установ. Це сприяє підвищенню якості надання медичних послуг, зменшенню часу на адміністративні завдання та покращенню взаємодії між лікарями, лабораторіями й аптеками [1], [2].

Крім того, впровадження програмного забезпечення для підтримки роботи з електронними медичними картками пацієнтів мінімізує ризик помилок і забезпечує швидкий доступ до історії хвороби пацієнта. Розвиток телемедицини дозволяє проводити консультації дистанційно, що особливо важливо для людей із віддалених регіонів або в умовах пандемій. Програмні рішення також допомагають лікарням відповідати вимогам регуляторних органів щодо збереження та захисту персональних даних пацієнтів [3], [4].

Загалом, сучасні програмні засоби дозволяють лікарням ефективніше використовувати ресурси, зменшують навантаження на персонал і підвищують якість медичного обслуговування, що робить їх важливим інструментом у розвитку охорони здоров'я [5], [6].

Проте деякі програми для підтримки роботи лікарні мають суттєві недоліки, які можуть ускладнювати їх впровадження та ефективне використання. Однією з головних проблем є висока вартість як початкового впровадження, так і подальшого обслуговування. Лікарням доводиться витратити значні кошти на придбання програмного забезпечення, налаштування програмної системи відповідно до власних потреб, а також на регулярні оновлення та підтримку. Ще одним недоліком є складність навчання персоналу, особливо якщо мова йде про медичних працівників, які не мають

досвіду роботи з подібними технологіями. Це може призводити до помилок у введенні даних, що в свою чергу впливає на якість діагностики та лікування. Додатково, лікарняні інформаційні системи часто не є універсальними, а їхня інтеграція з іншими медичними або державними реєстрами може бути складною або навіть неможливою без значних модифікацій. Окрім цього, деякі медичні заклади стикаються з опором змін з боку персоналу, який звик працювати за старими методами. Це може спричинити додаткові труднощі при впровадженні нових технологій та викликати небажання використовувати систему у повному обсязі. Крім того, деякі готові програмні рішення не завжди повністю відповідають потребам конкретної лікарні. Це змушує медичні установи або шукати компроміси, або витрачати додаткові ресурси на розробку спеціалізованих рішень, також ускладнює їхнє впровадження та супровід.

Тому актуальною є розробка програмного забезпечення для підтримки роботи лікарні. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для підтримки роботи лікарні.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для підтримки роботи лікарні;
- здійснити проєктування програмного забезпечення для підтримки роботи лікарні;
- створити програмне забезпечення для підтримки роботи лікарні;
- виконати тестування розробленого програмного забезпечення для підтримки роботи лікарні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для підтримки роботи лікарні

Програмне забезпечення для управління діяльністю лікарень відіграє ключову роль у забезпеченні безперервності медичного обслуговування. Його функціональність охоплює обробку та збереження інформації про пацієнтів, управління медичними записами, координацію процесів лікування та автоматизацію рутинних завдань, що підвищує ефективність роботи персоналу [1].

Оптимізація внутрішніх процесів дозволяє значно знизити ризик помилок, прискорити обробку запитів і забезпечити своєчасне надання медичних послуг. Завдяки централізованій системі зберігання та обміну даними лікарі та інший персонал можуть швидко отримувати доступ до необхідної інформації, що сприяє точнішій діагностиці та ефективному призначенню лікування [1].

Інтеграція різних підрозділів, таких як приймальне відділення, лабораторії, аптеки та фінансові служби, забезпечує зручну взаємодію між усіма учасниками медичного процесу. Це сприяє мінімізації паперової документації та покращує точність обліку даних. Використання такого програмного рішення дає змогу лікарням оптимізувати робочі процеси, зменшити адміністративне навантаження та приділяти більше уваги якості медичного обслуговування [1].

Автоматизація таких завдань, як запис на прийом, управління результатами аналізів та призначення медикаментів, забезпечує оперативність у прийнятті рішень. Лікарі можуть швидко отримувати доступ до електронних медичних карток пацієнтів, що допомагає уникнути затримок у процесі лікування та підвищує загальну ефективність системи охорони здоров'я [1].

Автоматизація процесів у лікарнях є важливою складовою сучасної медичної системи. Значні обсяги інформації про пацієнтів вимагають

ефективного управління, а координація послуг стає невід'ємною частиною якісного медичного обслуговування. Завдяки цифровим рішенням можна оптимізувати рутинні завдання, що дозволяє медичному персоналу більше уваги приділяти безпосередньо лікуванню [1].

Розвиток технологій створює нові очікування щодо доступності та швидкості надання медичної допомоги. Використання програмного забезпечення спрощує обмін даними, прискорює обробку запитів та забезпечує збереження інформації відповідно до вимог конфіденційності. Це сприяє підвищенню рівня безпеки для пацієнтів та покращенню якості медичних послуг [1].

Програмні системи в лікарнях дозволяють персоналу оперативно отримувати доступ до даних про пацієнтів, зокрема до історії їхніх звернень, попередніх діагнозів і призначених препаратів. Це сприяє ухваленню обґрунтованих рішень, зменшує ризик дублювання діагностики та допомагає уникнути можливих помилок при призначенні лікування [1].

Автоматизація обліку медичних препаратів забезпечує лікарям та медичним сестрам доступ до актуальної інформації про наявність ліків у різних відділеннях, що допомагає швидше забезпечувати пацієнтів необхідним лікуванням. Крім покращення якості медичних послуг, це також сприяє оптимізації витрат та підвищенню ефективності фінансового управління медичними закладами [1].

Інформаційні технології змінюють підхід до взаємодії між пацієнтами та медичними установами, роблячи процес лікування прозорішим і комфортнішим. Наявність усіх медичних записів у єдиній системі дозволяє пацієнтам бути більш поінформованими про свій стан здоров'я, що сприяє відповідальнішому ставленню до лікування та профілактики [1].

Автоматизація інформаційних процесів у медичних закладах за допомогою відповідного програмного забезпечення дозволяє значно підвищити ефективність роботи та зменшити кількість можливих помилок.

Використання сучасних технологій спрощує організацію розкладів, ведення обліку та формування звітності, що сприяє швидшому обслуговуванню пацієнтів і зменшенню часу очікування [1].

Електронна взаємодія між різними підрозділами медичного закладу, такими як лабораторії та аптеки, забезпечує оперативний обмін інформацією, що покращує координацію роботи. Оптимізується потік пацієнтів навіть у періоди пікового навантаження, що дає змогу більш ефективно керувати ресурсами лікарні. Наприклад, результати аналізів можуть автоматично відображатися у відповідних модулях, що прискорює процес ухвалення рішень і призначення лікування [1].

Фінансовий аспект впровадження програмних рішень у медичній сфері залежить від функціональності та масштабу закладу. Для невеликих клінік передбачені спрощені системи, які допомагають організувати запис на прийом і ведення фінансового обліку. У великих лікарнях застосовуються комплексні рішення, що охоплюють усі аспекти управління, включаючи контроль за медичними запасами, кадровий облік та моніторинг лікувальних процесів [1].

Завдяки цифровізації зменшуються витрати на паперову документацію та адміністративне управління, що сприяє раціональному використанню ресурсів. Використання автоматизованих систем дозволяє мінімізувати фінансові втрати, підвищити якість обслуговування та знизити навантаження на персонал [1].

Використання програмного забезпечення для підтримки роботи лікарень дозволяє значно підвищити ефективність медичних закладів. Запровадження цифрових рішень сприяє покращенню обробки інформації, спрощенню доступу до медичних даних і зменшенню паперової документації. Оптимізація робочих процесів дозволяє персоналу витрачати менше часу на адміністративні завдання, зосереджуючись на догляді за пацієнтами [1].

Впровадження сучасних систем забезпечує відповідність вимогам безпеки, гарантуючи захист персональних даних пацієнтів. Взаємодія між

різними відділами лікарні покращується, що особливо важливо для швидкого реагування у невідкладних ситуаціях. Завдяки цифровому керуванню потоками пацієнтів забезпечується ефективніше використання ресурсів і підвищується якість надання медичних послуг. Наприклад, результати аналізів можуть автоматично надходити до відповідних спеціалістів, скорочуючи час очікування і пришвидшуючи постановку діагнозу [1].

Медичні заклади, що впроваджують такі системи, отримують переваги не лише в організації роботи, а й у фінансовій сфері. Автоматизація дозволяє зменшити кількість помилок при оформленні документації, покращити процеси розрахунків та оптимізувати управління витратами. Раціональне використання технологій сприяє зростанню довіри з боку пацієнтів, підвищенню рівня обслуговування та формуванню позитивної репутації медичного закладу [1].

Програмне забезпечення розробляється з урахуванням потреб користувачів, тому його освоєння не потребує тривалого навчання. Інтуїтивно зрозумілий інтерфейс дає змогу швидко опанувати основні функції навіть тим працівникам, які не мають глибоких технічних знань. Більшість постачальників пропонують навчальні матеріали та підтримку, що дозволяє медичному персоналу швидко адаптуватися до нової системи і ефективно використовувати її у щоденній роботі [1].

На сучасному ринку представлено чимало цифрових рішень для автоматизації медичних закладів. Різні програмні комплекси допомагають керувати основними процесами лікарень, зокрема обліком пацієнтів, фінансовими операціями, розкладом прийомів і логістикою медичних послуг. Окремі системи призначені для ведення електронної документації, управління фінансами, координації лабораторних досліджень або підтримки комунікації з пацієнтами [1].

Комплексні медичні платформи сприяють кращій організації роботи медичних установ. Використання цифрових інструментів допомагає уникнути

перевантаження у приймальних відділеннях, оптимізує процес планування прийомів та покращує якість обслуговування. Автоматизація фінансового обліку зменшує ризики помилок у виставленні рахунків і контролі витрат. Системи для управління запасами забезпечують безперервну наявність необхідних матеріалів, що впливає на своєчасне надання медичних послуг [1].

Впровадження таких технологій дозволяє лікарням підвищити ефективність роботи, зменшити кількість адміністративних помилок і поліпшити якість лікування. Завдяки швидкому доступу до електронної історії хвороби лікарі можуть оперативніше ухвалювати рішення, що позитивно позначається на стані пацієнтів [1].

Ціна програмного забезпечення, яке забезпечує управління лікарняними процесами, може значно варіюватися залежно від його функціональності. Більш прості рішення з обмеженим набором можливостей потребують мінімальних витрат, тоді як комплексні платформи, що включають розширені інструменти, можуть вимагати суттєвих фінансових вкладень. Якщо система розробляється спеціально під потреби певного закладу, її створення зазвичай обходиться дорожче через необхідність глибокого налаштування під специфічні вимоги [1].

На остаточну вартість впливають також особливості інфраструктури закладу та необхідність інтеграції з іншими медичними або фінансовими системами. Заклади, які потребують посиленого рівня безпеки або масштабної мережевої взаємодії, стикаються з додатковими витратами на впровадження та обслуговування [1].

Використання таких рішень сприяє значному підвищенню ефективності лікарень. Автоматизація адміністративних процесів дозволяє зменшити витрати на персонал, скоротити кількість помилок і прискорити обслуговування пацієнтів. Завдяки цьому зменшуються експлуатаційні витрати, що позитивно впливає на фінансову стабільність закладу. Поліпшення

якості медичних послуг сприяє підвищенню рівня задоволеності пацієнтів, що формує довіру та може збільшити кількість звернень [1].

Програмні платформи, які використовуються в лікарнях, охоплюють широкий спектр функцій. Деякі з них орієнтовані на управління клінічними процесами, тоді як інші зосереджені на адмініструванні, логістиці або фінансових питаннях. Одні рішення допомагають контролювати графік прийомів, вести електронні записи, організовувати роботу персоналу, інші забезпечують контроль за обігом медикаментів, аналізами та діагностичними дослідженнями. Використання таких технологій сприяє підвищенню точності, зменшенню ризиків і оптимізації робочих процесів [1].

Системи для управління лікарнями, створені для великих медичних закладів, призначені для обробки значних обсягів інформації про пацієнтів і забезпечення безперебійної роботи всіх відділів. Вони інтегрують адміністративні, фінансові та медичні підрозділи, формуючи єдину інформаційну екосистему. Завдяки цьому вдосконалюється координація між підрозділами, що сприяє підвищенню ефективності медичних послуг та спрощує управління лікарняними процесами [1].

Такі рішення активно застосовуються в різних медичних установах, у тому числі державних. Використання подібного програмного забезпечення дозволяє значно спростити зберігання інформації, забезпечуючи швидкий доступ до медичних даних пацієнтів. Крім того, воно допомагає автоматизувати фінансові операції, керування розкладом прийомів і взаємодію між лікарняними підрозділами. Це сприяє підвищенню рівня обслуговування пацієнтів та оптимізації робочих процесів [1].

Для забезпечення конфіденційності інформації передбачені надійні засоби захисту, що мінімізують ризики витоку або несанкціонованого доступу. Таке програмне забезпечення дає змогу ефективно керувати історіями хвороб, організовувати графік прийомів і здійснювати контроль за фінансовими операціями, що покращує загальну продуктивність медичних закладів [1].

Інформаційні системи, призначені для підтримки лікарняних процесів, дозволяють значно спростити роботу медичного персоналу, зокрема у сфері обробки пацієнтських даних та повсякденних завдань. Використання централізованої платформи дає змогу оперативно отримувати доступ до медичних записів, результатів лабораторних досліджень та іншої важливої інформації, що сприяє скороченню часу на адміністративну роботу та підвищенню ефективності лікування [1].

Такі системи забезпечують безперебійну взаємодію між різними відділеннями медичного закладу, включаючи лабораторії та діагностичні центри, що дозволяє покращити координацію лікувального процесу. Автоматизований контроль за призначенням медикаментів і терапевтичними процедурами допомагає зменшити кількість можливих помилок і підвищити рівень безпеки пацієнтів [1].

Серед поширених рішень, що використовуються у сфері охорони здоров'я, виділяються комплексні платформи, які дозволяють ефективно організовувати робочі процеси та підтримувати співпрацю між фахівцями різних напрямків [1].

При виборі програмного забезпечення для лікарень важливим критерієм є баланс між витратами та функціональністю. Ефективні рішення повинні забезпечувати автоматизацію таких процесів, як планування прийомів, фінансовий облік та управління пацієнтськими даними, при цьому залишаючись доступними за вартістю. Завдяки таким технологіям медичні заклади можуть значно скоротити витрати та покращити якість обслуговування [1].

Лікарняні інформаційні платформи пропонують гнучкі рішення, що працюють у хмарному середовищі та доступні для закладів з обмеженими фінансовими можливостями. Завдяки цим технологіям навіть невеликі медичні установи можуть впроваджувати автоматизовані системи управління, не витрачаючи значних коштів [1].

Багато лікарень віддають перевагу мобільним технологіям, що дозволяють лікарям швидко переглядати дані пацієнтів, отримувати доступ до електронних карток та взаємодіяти з іншими відділеннями. Автоматизовані системи керування медичними записами забезпечують швидкий доступ до історії хвороби, ефективну організацію даних та безперешкодну роботу персоналу [1].

Функціональність таких програмних рішень може бути адаптована до конкретних потреб лікарні, що сприяє покращенню координації між фармацевтичними, лабораторними та діагностичними підрозділами. Однак розробка та впровадження спеціалізованого програмного забезпечення потребує значних фінансових витрат, тому для невеликих закладів важливо обирати економічно вигідні варіанти [1].

Деякі провайдери пропонують модульний підхід до ліцензування, що дозволяє медичним установам оплачувати лише необхідні функції, зменшуючи витрати та зберігаючи ефективність. Завдяки цьому лікарні можуть формувати власну систему управління відповідно до потреб без зайвих фінансових навантажень [1].

Програмні рішення для керування лікарняними установами створені для оптимізації управління ресурсами, включаючи медичне обладнання та інфраструктуру. Вони дозволяють контролювати графіки обслуговування, слідкувати за станом запасів і забезпечувати ефективне управління лікарняними об'єктами. Завдяки впровадженню таких систем можливо підтримувати безперебійну роботу закладу, за умови належного технічного супроводу [1].

Подібне програмне забезпечення також сприяє дотриманню вимог безпеки та регламентів, що є ключовими для медичних установ. Автоматизований контроль інвентаризації допомагає персоналу оперативно визначати місцезнаходження обладнання та уникати його нераціонального використання [1].

Застосування цифрових платформ управління лікарнею позитивно впливає на якість медичних послуг та ефективність лікування. Існують комплексні системи, що об'єднують моніторинг обладнання, контроль потоку пацієнтів та ведення медичних записів, що допомагає лікарням швидко реагувати на критичні ситуації та покращувати загальну координацію роботи [1].

Програмне забезпечення, призначене для контролю за пацієнтами, дозволяє спростити ведення обліку їхнього перебування у медичному закладі. Всі процеси, пов'язані з госпіталізацією, випискою та лікуванням, інтегруються в єдину систему, що забезпечує оперативний доступ до необхідної інформації та сприяє ефективному наданню медичної допомоги [1].

Програмні рішення для догляду за пацієнтами включають інструменти, що допомагають організувати прийом, оцінку результатів медичних тестів та забезпечення комунікації між пацієнтами і медичним персоналом. Це дає змогу постійно тримати всіх учасників процесу в курсі лікування та подальших процедур, що забезпечує належний рівень координації та прозорості [1].

Використання таких програмних систем дозволяє автоматизувати реєстрацію пацієнтів і планування їхніх візитів до медичних установ. Це знижує навантаження на персонал і спрощує організацію потоку пацієнтів [1].

Програмне забезпечення для управління розкладом допомагає значно знизити витрати в лікарнях, одночасно оптимізуючи використання ресурсів. Такі рішення дозволяють ефективно управляти часом, координувати розклад медичних працівників і доступ до необхідних кабінетів та обладнання. Це важливо для великих медичних установ, де необхідно синхронізувати графіки кількох відділень та уникати непередбачуваних конфліктів у плануванні [1].

Завдяки таким системам, як програмне забезпечення для планування роботи медичних працівників, можна зменшити час очікування пацієнтів і покращити загальну ефективність роботи лікарні. Такі інструменти

забезпечують більш плавний і скоординований процес лікування, що в результаті підвищує якість медичних послуг і задоволеність пацієнтів [1].

Для забезпечення високоякісної та організованої медичної допомоги лікарням необхідно використовувати спеціалізовані системи для управління пацієнтами. Такі рішення охоплюють реєстрацію пацієнтів, обробку рахунків і управління медикаментами, об'єднуючи всі ці функції в одну платформу. Це дозволяє лікарям і медичним установам ефективно контролювати весь процес взаємодії з пацієнтами від моменту їх надходження до виписки [1].

Використання таких систем сприяє підвищенню точності ведення медичних даних та покращенню загальної роботи лікарень, оскільки спрощує доступ до необхідної інформації. Інтеграція з іншими медичними системами додає додаткову зручність, що робить ці рішення ще більш популярними серед медичних установ [1].

Одним із прикладів є програмне забезпечення, яке надає можливості для керування пацієнтами, виставлення рахунків та створення звітів, забезпечуючи повний цикл обробки інформації про пацієнтів [1].

Програмне забезпечення для управління лікарнею значно покращує ефективність роботи медичних установ, автоматизуючи багато процесів. Вибір оптимальної системи дозволяє значно підвищити продуктивність лікарні завдяки таким функціям, як управління пацієнтами, планування та виставлення рахунків [1].

Існує ряд програмних рішень, що пропонують як локальні, так і хмарні варіанти, надаючи гнучкість у виборі підходящої платформи для лікарні. Такі системи забезпечують зручний інтерфейс і потужні можливості інтеграції даних, що робить їх популярними серед медичних установ різного масштабу [1].

Для вибору оптимального програмного забезпечення для лікарні слід враховувати кілька важливих аспектів, таких як надійність захисту даних, зручність використання та здатність інтегруватися з іншими існуючими

системами. Важливим є також наявність підтримки та регулярних оновлень програмного забезпечення, що дозволяє зберігати актуальність системи на всіх етапах її експлуатації [1].

Крім того, зручність користувацького інтерфейсу та включення в систему ключових функцій, таких як облік, ведення пацієнтських записів та планування прийомів, дозволяє заощаджувати час працівникам лікарні. Система має бути гнучкою і масштабованою, що дозволяє адаптувати її під збільшення обсягу роботи лікарні. Також не менш важливою є забезпечення надійного обслуговування користувачів, яке включає підтримку в процесі налаштування системи та вирішення можливих технічних проблем [1].

Вибір програмного забезпечення для лікарні повинен також враховувати високий рівень безпеки даних, зокрема захист медичної інформації пацієнтів [1].

Існують безкоштовні варіанти програмного забезпечення для управління лікарнями, однак вони можуть не включати додаткові функції, доступні в комерційних рішеннях. Такі програми часто використовуються невеликими медичними закладами або організаціями, що мають обмежені ресурси. Зазвичай, базові функції включають реєстрацію пацієнтів та планування прийомів, але можуть бути обмежені в порівнянні з більш складними системами [1].

Безкоштовне програмне забезпечення, що пропонується для медичних установ, часто зосереджено на простоті використання та базових функціях для обробки даних пацієнтів. Однак воно може не мати функцій, таких як розширене управління медичними записами або автоматизація більш складних процесів. При цьому деякі з таких систем можуть ефективно задовольняти потреби невеликих медичних практик або організацій з обмеженим бюджетом [1].

Важливим аспектом сучасних інформаційних систем лікарень є їх здатність адаптуватися до нових вимог охорони здоров'я. Зокрема, системи

повинні включати функціонал для дистанційного моніторингу, інтеграції з іншими програмними платформами, підтримки штучного інтелекту для діагностики, а також для аналізу та прогнозування медичних даних. Такі інструменти можуть значно підвищити ефективність лікування та оптимізувати робочі процеси в медичних закладах [1].

Захист даних пацієнтів є важливим аспектом роботи будь-якої медичної інформаційної системи. Оскільки кіберзагрози продовжують зростати, лікарні потребують надійного захисту своєї ІТ-інфраструктури. Важливою частиною безпеки є шифрування даних, ідентифікація користувачів та використання додаткових методів захисту, таких як двофакторна користувачів [1].

При виборі програмного забезпечення для управління лікарнею важливо оцінювати продуктивність системи, її надійність, зручність у використанні та здатність інтегруватися з іншими системами. Це дозволяє не лише забезпечити ефективність роботи медичного персоналу, а й знизити адміністративний тягар і покращити загальний рівень обслуговування пацієнтів [1].

Програмне забезпечення для управління лікарнями стало важливою частиною медичних установ, оскільки воно відповідає потребам різних медичних працівників. Такі системи забезпечують стандартизацію процесів догляду за пацієнтами, що дозволяє полегшити доступ до медичних даних, історії хвороб та рахунків. Вони активно використовуються як у великих лікарнях, так і в невеликих клініках, покращуючи загальну ефективність медичного обслуговування [1], [2].

Дані системи підтримують різноманітні сфери використання, від забезпечення доступу до медичних записів для великих медичних закладів до організації віддаленого доступу для сільських медичних центрів. Вони також сприяють міжнародному управлінню пацієнтами, дозволяючи зберігати та обробляти медичні записи пацієнтів навіть за межами країни. Крім того, таке

програмне забезпечення є незамінним у випадках надзвичайних ситуацій, коли швидке і ефективне прийняття пацієнтів є критичним [1], [2].

Сучасні тенденції розвитку програмних рішень для лікарень спрямовані на інтеграцію технологій штучного інтелекту та автоматизації. Системи майбутнього будуть мати здатність ефективно обробляти великі обсяги даних, що дозволить лікарням більш точно стежити за пацієнтами, прогнозувати потреби в лікуванні та забезпечувати ефективніше лікування. Автоматизація процесів обробки даних дозволить зменшити навантаження на медичний персонал, знижуючи кількість повторюваних завдань і оптимізуючи робочі процеси [2], [3].

Додатково, технології мобільних рішень і віддаленого доступу до даних значно полегшують роботу медичного персоналу, дозволяючи здійснювати моніторинг пацієнтів у реальному часі та забезпечувати кращий доступ до необхідних відомостей звідусіль. Прогнозування технічного обслуговування систем також дає можливість своєчасно усувати неполадки, забезпечуючи безперебійну роботу лікарень [3], [4].

Програмне забезпечення для медичних ІТ-систем продовжує вдосконалюватися, забезпечуючи лікарні потужними інструментами для підвищення ефективності та точності збору даних. В очікуваному майбутньому ці рішення повинні стати ще більш стійкими, здатними задовольняти зростаючі вимоги охорони здоров'я, а також покращити процеси догляду за пацієнтами [3], [4].

Сучасні рішення для управління лікарнями дозволяють покращити результати лікування завдяки більш точному прогнозуванню та аналізу даних. Використання таких інструментів робить процес надання медичних послуг більш оптимізованим. Одним із напрямків розвитку є інтеграція технологій штучного інтелекту, що допомагає лікарям більш ефективно приймати рішення, а також полегшує доступ до медичних записів та необхідної інформації [5], [6].

Завдяки автоматизації та інтеграції мобільних рішень, лікарні можуть оперативно вирішувати завдання, що знижує необхідність у виконанні рутинних операцій і зменшує навантаження на медичний персонал. Впровадження новітніх інструментів, які використовують штучний інтелект для підтримки прийняття рішень, а також забезпечення моніторингу в реальному часі, стане значним кроком у розвитку медичних інформаційних систем [5], [6].

Програмне забезпечення для управління лікарнею допомагає також у покращенні безпеки даних пацієнтів, що є важливим аспектом у боротьбі з кіберзагрозами. Завдяки новим функціям, таким як автоматизовані процеси і підтримка прогнозової аналітики, лікарні зможуть ефективніше задовольняти потреби пацієнтів і забезпечувати якісне обслуговування. Технології майбутнього сприятимуть покращенню догляду та оптимізації роботи медичних установ, відкриваючи нові можливості для зростання й інновацій [7], [8].

Відзначено, що програмне забезпечення для управління діяльністю лікарень відіграє ключову роль у забезпеченні безперервності медичного обслуговування. Його функціональність охоплює обробку та збереження інформації про пацієнтів, управління медичними записами, координацію процесів лікування та автоматизацію рутинних завдань, що підвищує ефективність роботи персоналу.

1.2 Аналіз програмного забезпечення для підтримки роботи лікарні

Проаналізуємо програмне забезпечення для підтримки роботи лікарні [9]-[14].

Програмне забезпечення Caspio (рис. 1.1) дозволяє медичним установам розробляти власні цифрові рішення без необхідності складного

програмування. Завдяки цій платформі можна легко налаштовувати функціонал для оптимізації управління лікарняними процесами, включаючи адміністрування записів пацієнтів, контроль за запасами, фінансовий менеджмент і координацію надання медичних послуг. Забезпечуючи відповідність сучасним стандартам захисту медичних даних, Caspio гарантує безпечне зберігання інформації та її доступність у хмарному середовищі. Завдяки інтеграції з системами обміну медичними даними, цей інструмент підтримує ефективну взаємодію між різними відділеннями лікарні, допомагаючи в координації лікувальних заходів. [9], [10], [12].

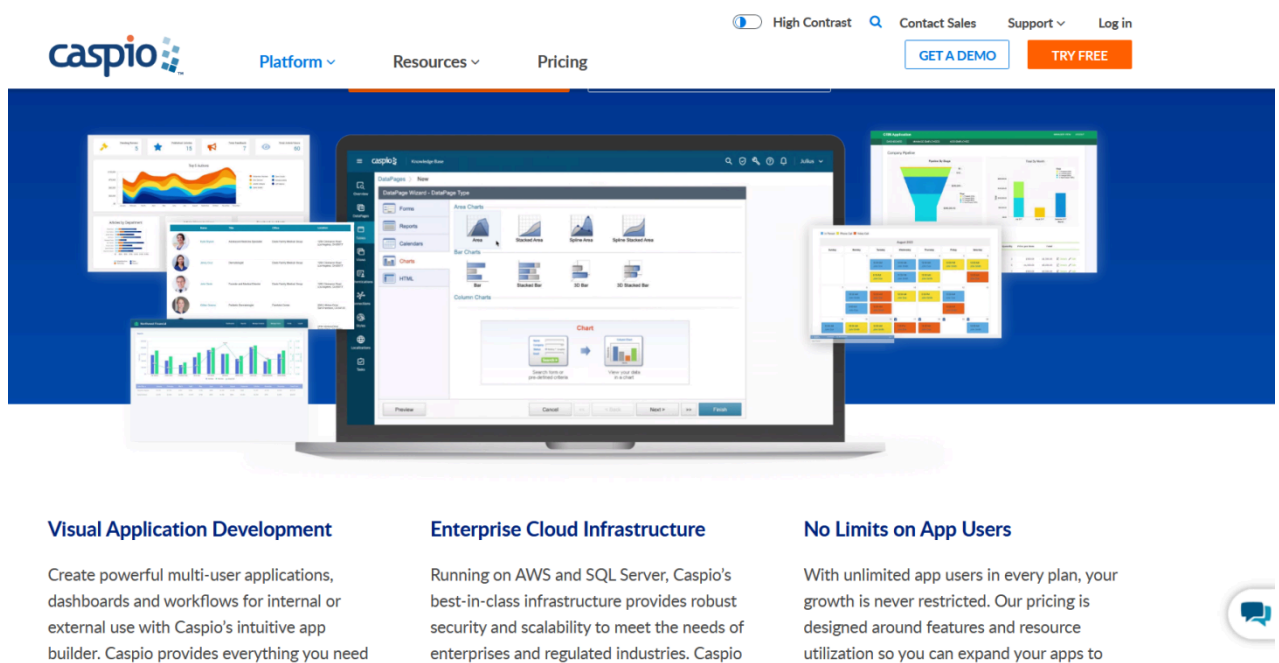


Рисунок 1.1 – Програмне забезпечення Caspio [12]

Функціонал програмного забезпечення Caspio включає можливість впровадження розкладу лікарів, автоматизацію процесів адміністрування та моніторинг стану пацієнтів у режимі реального часу. Гнучкість налаштувань дозволяє адаптувати рішення до специфічних потреб кожної лікарні, що робить цей продукт універсальним інструментом для забезпечення ефективної роботи медичних закладів. [9], [10], [12].

Основними характеристиками програмного забезпечення Caspio є такі [9], [10], [12]:

- гнучкість та налаштування: програмне забезпечення Caspio дозволяє створювати та адаптувати додатки відповідно до потреб конкретної лікарні. Завдяки інтуїтивному інтерфейсу та низькокодовій платформі, медичні установи можуть розробляти власні рішення без необхідності глибоких знань у програмуванні [9], [10], [12];

- хмарна архітектура: програмне забезпечення Caspio працює в безпечному хмарному середовищі, що забезпечує надійне зберігання даних та їх доступність з будь-якого пристрою. Це особливо важливо для лікарень, де потрібен швидкий доступ до медичних записів та оперативне управління ресурсами [9], [10], [12];

- безпека та відповідність стандартам: програмне забезпечення Caspio підтримує вимоги щодо захисту конфіденційних даних у сфері охорони здоров'я. Інструменти шифрування програмного забезпечення Caspio та контроль доступу дозволяють зберігати інформацію в безпечному середовищі, зменшуючи ризики витоку даних [9], [10], [12];

- інтеграція з медичними системами: програмне забезпечення Caspio підтримує сумісність із загальноприйнятими стандартами обміну медичними даними. Це сприяє безперешкодному обміну інформацією між відділеннями лікарні та зовнішніми організаціями, що покращує координацію лікування пацієнтів [9], [10], [12];

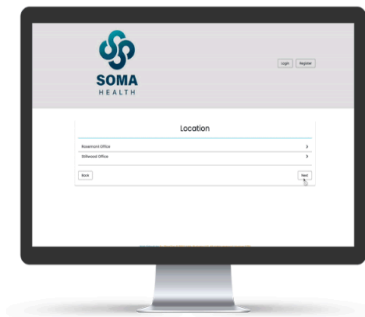
- автоматизація процесів: програмне забезпечення Caspio допомагає зменшити адміністративне навантаження за рахунок автоматизації багатьох лікарняних процесів. Завдяки цьому можна швидше обробляти заявки пацієнтів, управляти розкладом лікарів, контролювати запаси медичних препаратів та вести фінансовий облік [9], [10], [12];

- аналітика та звітність: програмне забезпечення Caspio надає розширені можливості аналітики, що дозволяє лікарням оцінювати

ефективність медичних послуг та приймати обґрунтовані рішення. Гнучкі інструменти створення звітів допомагають аналізувати тенденції та оптимізувати роботу закладу [9], [10], [12].

Серед недоліків програмного забезпечення Caspio можна відзначити обмежену гнучкість у реалізації складних медичних рішень, залежність від стабільного Інтернет з'єднання, можливі труднощі з інтеграцією зі специфічними медичними системами, необхідність навчання персоналу для ефективного використання функцій платформи, а також потенційно високі витрати на масштабування та розширення можливостей. Інтеграція програмного забезпечення Caspio з існуючими медичними системами, такими як електронні медичні картки чи лабораторні інформаційні системи, може потребувати додаткових ресурсів. Крім того, хоча інтерфейс орієнтований на користувачів без глибоких знань у програмуванні, складніші налаштування можуть вимагати залучення технічних спеціалістів. Також вартість розширених функцій та підтримки може стати значним фактором для лікарень з обмеженим бюджетом [9], [10], [12].

Програмне забезпечення TimeTap (рис. 1.2) дозволяє медичним установам ефективно керувати графіками прийому, зменшуючи навантаження на адміністративний персонал. Завдяки інтегрованій системі управління календарем можна відстежувати розклад лікарів, координувати записи на прийом та автоматично надсилати нагадування пацієнтам, що сприяє підвищенню рівня обслуговування [9], [13].



CUSTOMIZE YOUR SCHEDULING PROCESS

Build a scheduling journey that is easy for customers to book appointments. Create selection fields for staff, services, or classes. You can create a comprehensive scheduling site that directs people to book the appointments they desire.

- Customize the flow of scheduling selections
- Design custom booking sites to match your brand
- Build custom booking forms

[→ LEARN MORE](#)

GET SOLUTIONS THAT SUPPORT YOUR BUSINESS

We've designed our software to be fully configurable to meet your complex scheduling requirements. From custom booking sites to powerful data reporting to our API and integrations - we provide more than a basic appointment scheduling software.



Рисунок 1.2 – Програмне забезпечення TimeTap [13]

Програмне забезпечення TimeTap дозволяє лікарням налаштовувати форми для бронювання з урахуванням специфіки медичних послуг. Функція автоматизації списків очікування допомагає швидко реагувати на зміни в розкладі та ефективно розподіляти вільні слоти. Завдяки синхронізації з календарями лікарів забезпечується точність планування, а можливість інтеграції з іншими інформаційними системами медичного закладу робить управління пацієнтськими записами ще зручнішим [9], [13].

Впровадження програмного забезпечення TimeTap дозволяє лікарням спростити обробку заявок, скоротити кількість пропущених прийомів і покращити взаємодію між персоналом і пацієнтами. Гнучкі налаштування допомагають адаптувати систему до потреб конкретної медичної установи, включаючи різні відділення, типи послуг і форми оплати. Завдяки дотриманню вимог щодо безпеки та конфіденційності медичних даних, TimeTap забезпечує надійний захист інформації та відповідність стандартам у сфері охорони здоров'я [9], [13].

Програмне забезпечення TimeTar має такі особливості [9], [13]:

- автоматизація запису на прийом: програмне забезпечення TimeTar спрощує процес бронювання візитів, знижує адміністративне навантаження та мінімізує помилки під час реєстрації пацієнтів [9], [13];

- гнучке налаштування форм запису: програмне забезпечення TimeTar дозволяє створювати персоналізовані анкети для пацієнтів відповідно до специфіки медичного закладу [9], [13];

- керування списками очікування: програмне забезпечення TimeTar автоматично організовує черги, пропонуючи вільні часові інтервали у разі скасування записів [9], [13];

- двостороння синхронізація календарів: програмне забезпечення TimeTar підтримує інтеграцію з різними сервісами для забезпечення актуального розкладу лікарів [9], [13];

- автоматичні нагадування: програмне забезпечення TimeTar надсилає пацієнтам повідомлення через SMS або електронну пошту для зменшення кількості пропущених прийомів [9], [13];

- безпека та конфіденційність: програмне забезпечення TimeTar відповідає медичним стандартам захисту даних, забезпечуючи безпечне зберігання інформації [9], [13];

- інтеграція з іншими системами: програмне забезпечення TimeTar може бути підключено до внутрішніх лікарняних інформаційних систем, електронних медичних карток та фінансових сервісів [9], [13];

- гнучкість масштабування: програмне забезпечення TimeTar підходить як для невеликих клінік, так і для великих медичних центрів із численними відділеннями та філіями [9], [13];

- оптимізація управління персоналом: програмне забезпечення TimeTar дає можливість координувати графіки роботи лікарів та адміністративного персоналу, підвищуючи ефективність планування [9], [13];

- підтримка платформ: програмне забезпечення TimeTar доступно у вигляді вебзастосунку та мобільних застосунків (iOS, Android) [9], [13];
- безпека: : програмне забезпечення TimeTar відповідає міжнародним стандартам захисту медичних даних [9], [13];
- масштабованість: : програмне забезпечення TimeTar підходить для лікарень будь-якого розміру, включаючи мережі медичних закладів [9], [13];
- функціонал онлайн оплати та виставлення рахунків: програмне забезпечення TimeTar дозволяє пацієнтам здійснювати платежі безпосередньо через платформу, що спрощує фінансові операції лікарні [9], [13].

Серед недоліків програмного забезпечення TimeTar можна відзначити обмежену функціональність безкоштовної версії, що ускладнює використання платформи без фінансових вкладень. Деякі користувачі стикаються з труднощами при налаштуванні інтеграцій з іншими медичними системами, що може вимагати додаткових технічних ресурсів. Інтерфейс, хоч і зручний, може здаватися перевантаженим для нових користувачів, що потребує часу на адаптацію. Крім того, ціна підписки може бути високою для невеликих клінік, особливо з урахуванням необхідності масштабування для великих медичних закладів [9], [13].

Програмне забезпечення SoftClinic (рис. 1.3) розроблено для забезпечення ефективної роботи лікарень, клінік та інших медичних установ, пропонуючи комплексне рішення для управління всіма аспектами охорони здоров'я. Ця платформа сприяє покращенню якості медичних послуг, спрощуючи процеси обліку пацієнтів, ведення електронних медичних записів, планування прийомів та аналізу даних. SoftClinic створено спеціально для підтримки медичних закладів, надаючи їм інструменти для оптимізації внутрішніх процесів і підвищення продуктивності роботи персоналу [9], [14].

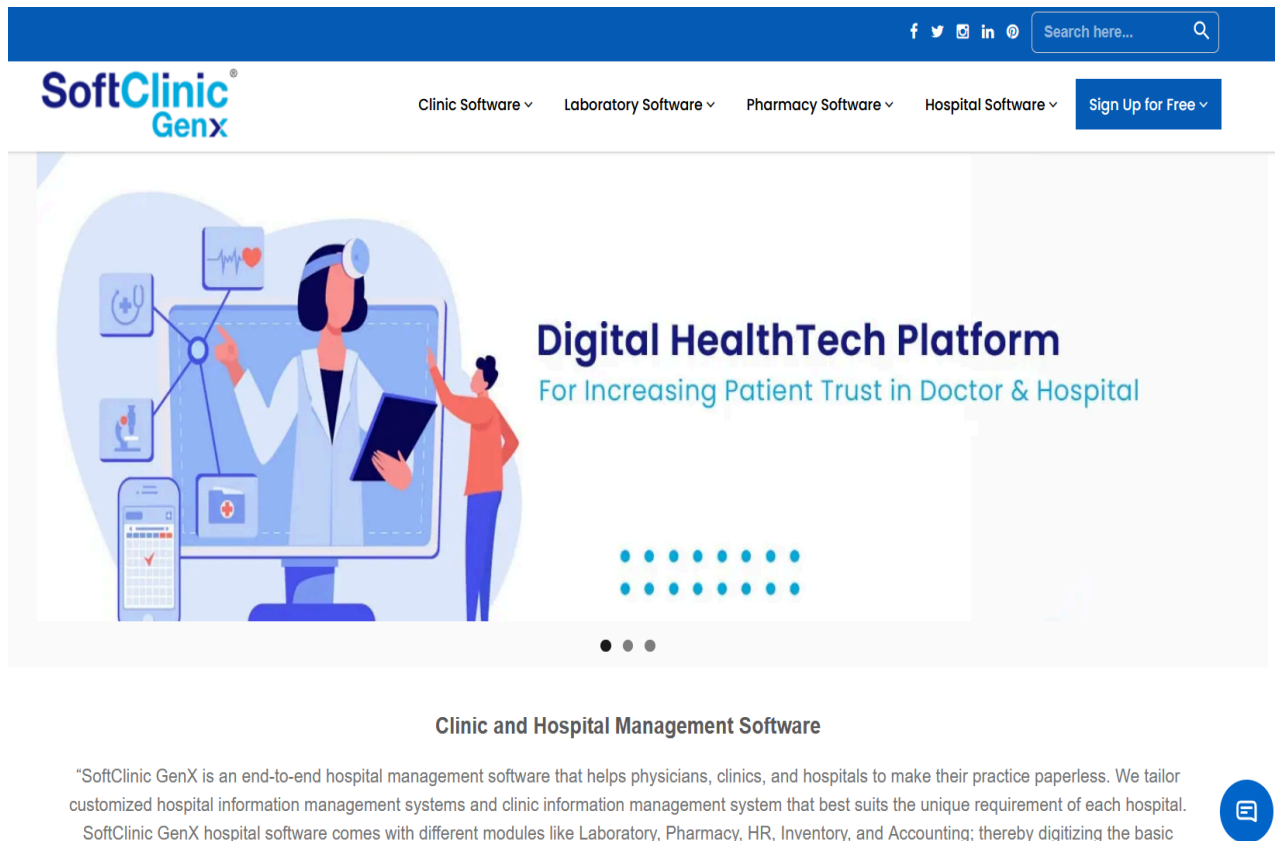


Рисунок 1.3 – Програмне забезпечення SoftClinic [14]

Використання програмного забезпечення SoftClinic дозволяє лікарням легко зберігати та обробляти дані про пацієнтів, включаючи електронні медичні записи, історію лікування та автоматизовані розрахунки, що сприяє швидкому ухваленню клінічних рішень. Для ефективного управління зустрічами передбачено систему планування, яка допомагає лікарям координувати візити, зменшуючи адміністративне навантаження та кількість пропущених прийомів. Завдяки автоматичним нагадуванням та можливості налаштування правил бронювання пацієнти отримують своєчасні повідомлення, що покращує їхню взаємодію з лікарнею [9], [14].

Медичні установи, які використовують SoftClinic, отримують доступ до гнучких інструментів звітності, що дозволяють створювати аналітичні звіти, прогнозувати тенденції та експортувати дані для подальшого аналізу. Це дає можливість адміністрації лікарень приймати обґрунтовані рішення щодо покращення надання медичних послуг. Вбудовані модулі, такі як аптека,

лабораторія, управління персоналом та фінансами, забезпечують комплексний підхід до організації роботи лікарні, роблячи SoftClinic надійним вибором для сучасних медичних закладів [9], [14].

Програмне забезпечення SoftClinic має такі особливості [9], [14]:

- засоби управління пацієнтами: програмне забезпечення SoftClinic надає можливості ведення електронних медичних записів з можливістю швидкого доступу, відстеження історії хвороб пацієнтів, включаючи попередні діагнози, призначене лікування та результати аналізів, автоматичні розрахунки для підтримки клінічних рішень [9], [14];

- планування та адміністрування: програмне забезпечення SoftClinic надає можливості запису на прийом, що дозволяє лікарям ефективно керувати розкладом, забезпечує автоматичні нагадування для пацієнтів через повідомлення та електронну пошту для зменшення кількості пропущених візитів. Налаштування спеціальних правил бронювання, що дає змогу адаптувати процес запису під потреби клініки [9], [14];

- медичний та фінансовий менеджмент: програмне забезпечення SoftClinic надає вбудовані модулі для управління аптекою, лабораторією, персоналом і фінансами. Програмне забезпечення SoftClinic надає інструменти для ведення бухгалтерії та контролю витрат лікарні, а також забезпечує генерацію електронних рахунків та підтримку страхових платежів [9], [14];

- звітування та аналітика: програмне забезпечення SoftClinic надає можливості формування звітів, що дають змогу аналізувати ефективність лікування та фінансові показники. Програмне забезпечення SoftClinic надає можливості експорту даних у різних форматах для подальшого аналізу, а також виявлення тенденцій та прогнозування результатів на основі медичних даних [9], [14];

- сумісність та інтеграція: програмне забезпечення SoftClinic надає можливості підтримки міжнародних стандартів безпеки медичних даних, інтеграції з іншими медичними інформаційними системами, а також

підтримки багато користувацького режиму для спільної роботи лікарів і персоналу [9], [14];

– глобальна доступність та масштабованість: програмне забезпечення SoftClinic надає можливості використання у клініках та лікарнях різного рівня, від невеликих медичних кабінетів до великих госпіталів. Забезпечується наявність багатомовної підтримки для міжнародних користувачів, а також доступ до системи через вебінтерфейс або мобільний застосунок [9], [14].

Серед недоліків програмного забезпечення SoftClinic можна відзначити складність освоєння для нових користувачів через велику кількість функцій та налаштувань, що потребують часу на адаптацію. Деякі користувачі зазначають, що інтерфейс не завжди інтуїтивно зрозумілий, особливо для медичного персоналу без досвіду роботи з подібними системами. Також можуть виникати труднощі з інтеграцією SoftClinic з іншими медичними інформаційними системами, що ускладнює обмін даними між різними платформами. Вартість використання програми, особливо для невеликих клінік, може бути досить високою, враховуючи необхідність постійної підтримки та оновлень. Крім того, можливі випадки технічних збоїв або затримок у роботі, що може впливати на швидкість обробки даних і прийняття рішень у лікарнях [9], [14].

Порівняльну характеристику програмного забезпечення для підтримки роботи лікарні наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для підтримки роботи лікарні

Критерій порівняння	Caspio [12]	TimeTap [13]	SoftClinic [14]
Управління записами пацієнтів	–	+	+
Синхронізація з календарями	+	+	–
Нагадування для пацієнтів	–	+	+
Аналітика та звітність	+	+–	+

Гнучкість налаштувань	+	+-	+
Модулі для лабораторії та аптеки	-	-	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для підтримки роботи лікарні. Проте деякі програми для підтримки роботи лікарні мають суттєві недоліки, які можуть ускладнювати їх впровадження та ефективне використання. Однією з головних проблем є висока вартість як початкового впровадження, так і подальшого обслуговування. Лікарням доводиться витрачати значні кошти на придбання програмного забезпечення, налаштування програмної системи відповідно до власних потреб, а також на регулярні оновлення та підтримку. Ще одним недоліком є складність навчання персоналу, особливо якщо мова йде про медичних працівників, які не мають досвіду роботи з подібними технологіями. Це може призводити до помилок у введенні даних, що в свою чергу впливає на якість діагностики та лікування. Додатково, лікарняні інформаційні системи часто не є універсальними, а їхня інтеграція з іншими медичними або державними реєстрами може бути складною або навіть неможливою без значних модифікацій. Окрім цього, деякі медичні заклади стикаються з опором змін з боку персоналу, який звик працювати за старими методами. Це може спричинити додаткові труднощі при впровадженні нових технологій та викликати небажання використовувати систему у повному обсязі. Крім того, деякі готові програмні рішення не завжди повністю відповідають потребам конкретної лікарні. Це змушує медичні установи або шукати компроміси, або витрачати додаткові ресурси на розробку спеціалізованих рішень, також ускладнює їхнє впровадження та супровід.

Тому актуальною є розробка програмного забезпечення для підтримки роботи лікарні.

При розробці програмного забезпечення для підтримки роботи лікарні необхідно забезпечити такі функціональні вимоги:

- підтримка можливості авторизації та реєстрації користувачів ПЗ лікарні;
- підтримка можливості отримання пацієнтом інформації про медичні обстеження;
- можливість вибору мови інтерфейсу користувача;
- можливість додавання нових лікарів до системи;
- підтримка можливості складання розкладу роботи лікарів;
- підтримка можливості запису на прийом до лікарів на вільний час згідно з їх розкладом роботи.

1.3 Висновки за розділом 1

Відзначено, що програмне забезпечення для управління діяльністю лікарень відіграє ключову роль у забезпеченні безперервності медичного обслуговування. Його функціональність охоплює обробку та збереження інформації про пацієнтів, управління медичними записами, координацію процесів лікування та автоматизацію рутинних завдань, що підвищує ефективність роботи персоналу.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для підтримки роботи медичних закладів. Проте деякі програми для підтримки роботи лікарні мають суттєві недоліки, які можуть ускладнювати їх впровадження та ефективне використання. Однією з головних проблем є висока вартість як початкового впровадження, так і подальшого обслуговування. Лікарням доводиться витратити значні кошти на придбання програмного забезпечення, налаштування програмної системи відповідно до власних потреб, а також на регулярні оновлення та підтримку. Ще одним недоліком є складність навчання персоналу, особливо якщо мова йде про медичних працівників, які не мають досвіду роботи з подібними технологіями. Це може призводити до помилок у

введенні даних, що в свою чергу впливає на якість діагностики та лікування. Додатково, лікарняні інформаційні системи часто не є універсальними, а їхня інтеграція з іншими медичними або державними реєстрами може бути складною або навіть неможливою без значних модифікацій. Окрім цього, деякі медичні заклади стикаються з опором змін з боку персоналу, який звик працювати за старими методами. Це може спричинити додаткові труднощі при впровадженні нових технологій та викликати небажання використовувати систему у повному обсязі. Крім того, деякі готові програмні рішення не завжди повністю відповідають потребам конкретної лікарні. Це змушує медичні установи або шукати компроміси, або витрачати додаткові ресурси на розробку спеціалізованих рішень, також ускладнює їхнє впровадження та супровід. Тому актуальною є розробка програмного забезпечення для підтримки роботи лікарні.

Сформульовано функціональні вимоги до програмного забезпечення для підтримки роботи лікарні.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для підтримки підтримки роботи лікарні було використано мову програмування C# [15], [16].

C# є об'єктно-орієнтованою мовою програмування, розробленою спеціально для екосистеми Microsoft .NET. Вона використовує синтаксис, подібний до C++ та Java, що забезпечує зручність для розробників, знайомих із цими мовами. Її статична типізація підтримує широкий спектр можливостей, зокрема розширення функціоналу, перевизначення операторів, механізми інкапсуляції, делегати, події, загальні типи, ітератори та технологію LINQ. Крім того, реалізована підтримка XML-даних [15], [16].

Концепції C# частково запозичені з інших мов, таких як C++, Delphi, Smalltalk, Java та C. Водночас ця мова має деякі обмеження, що впливають на характер програмного забезпечення. Наприклад, C++ не містить вбудованих механізмів багатопотоковості, керованого доступу до пам'яті або її автоматичного управління. Водночас C# працює у середовищі керованого коду, що забезпечує певний рівень безпеки, але водночас може піддаватися атакам через механізми «небезпечного» коду [15], [16].

Відмінності між C# і C++ простежуються також у підходах до компіляції та виконання програмного коду. C++ є системно-залежною мовою, яка генерує машинний код для конкретного процесора, тоді як C# компілюється у проміжний код, який виконується у середовищі Common Language Runtime (CLR). Завдяки цьому CLR оптимізує роботу програм, зменшуючи навантаження на систему та змінюючи порядок виконання команд відповідно до можливостей процесора. Однією з ключових функцій є автоматичне збирання сміття, що покращує управління пам'яттю та запобігає її витокам [15], [16].

Основою екосистеми .NET є програмна платформа .NET Framework, яка містить середовище виконання CLR та бібліотеку класів BCL. Завдяки цьому програми, створені на будь-якій мові, що підтримує .NET, мають доступ до функціоналу бібліотек, таких як створення об'єктів, виклик методів та використання стандартних класів [15], [16].

Для створення розподілених систем використовується Windows Communication Facility (WCF), що дозволяє взаємодіяти різними службам за допомогою асинхронного обміну даними. Служби можуть працювати у середовищі IIS або існувати як окремі програми, а клієнт-серверна модель забезпечує передачу запитів та відповідей. Обмін інформацією може відбуватися у простій формі (наприклад, передача символів чи слів у форматі XML) або включати складніші послідовності повідомлень [15], [16].

Зручність використання є однією з ключових переваг мови програмування C#, що забезпечує доступ до мережевих ресурсів, взаємодію із зовнішніми службами та підтримку механізмів зворотного виклику. Широкі можливості цієї мови програмування дозволяють розробляти програмне забезпечення для різних платформ, включаючи вебзастосунки, настільні та мобільні програми [15], [16].

Суттєвою перевагою мови C# є безпека та надійність. Оскільки медична інформація є критично важливою та конфіденційною, C# надає вбудовані механізми для безпечного зберігання та передачі даних, зокрема через шифрування, аутентифікацію користувачів та контроль доступу. Крім того, автоматичне управління пам'яттю (Garbage Collection) у C# зменшує ймовірність витоків пам'яті та інших помилок, що можуть призвести до збоїв у роботі програми [15], [16].

Ще одним важливим фактором вибору мови C# для розробки є її кросплатформеність через .NET Core. Якщо лікарня потребує вебзастосунків чи мобільних застосунків, то C# дозволяє створювати їх із використанням

ASP.NET Core для вебінтерфейсів і Xamarin (MAUI) для мобільних додатків, які можуть працювати на iOS та Android [15], [16].

Крім того, велика спільнота розробників і підтримка роблять C# хорошим вибором, оскільки завжди можна знайти документацію, приклади реалізацій та фахівців, які допоможуть у разі виникнення труднощів. Завдяки всім цим факторам C# є ефективним вибором для розробки медичних інформаційних систем, забезпечуючи стабільність, масштабованість та високий рівень безпеки [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для підтримки роботи лікарні наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для підтримки роботи лікарні

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Велика спільнота розробників	+	+	+—
Зручність синтаксису	+	—	+—
Підтримка об'єктно-орієнтоване програмування	+	+—	+
Зручність для створення ПЗ для підтримки роботи лікарні	+	—	—
Безпека та надійність	+	+—	+—

Отже, для реалізації програмного забезпечення для підтримки роботи лікарні обрано мову програмування C#, яка разом з середовищем Microsoft .NET забезпечує потужну інфраструктуру для створення надійних, масштабованих і захищених програмних систем. Багато лікарень використовують Windows-сервери та Microsoft SQL Server, що робить C#

оптимальним вибором завдяки його чудовій інтеграції з цими технологіями. Крім того, С# підтримує об'єктно-орієнтоване програмування, що дозволяє будувати модульні та легко підтримувані програмні системи. Це особливо важливо для медичних програмних систем, які містять багато взаємопов'язаних компонентів, таких як управління електронними медичними записами, планування прийомів, управління персоналом.

2.2 Вибір середовища розробки для створення програмного забезпечення для підтримки роботи лікарні

В якості середовища розробки для створення програмного забезпечення для підтримки роботи лікарні було обрано середовище Visual Studio [17], [18].

Середовище розробки Visual Studio є потужним середовищем розробки, призначеним для створення програмного забезпечення на платформі .NET. Це середовище підтримує широкий спектр мов програмування, зокрема С#, VB.NET, F#, а також мови веб-розробки, такі як JavaScript, TypeScript, HTML, CSS і SQL. Завдяки розширеній системі підказок і інструментів для рефакторингу забезпечується висока продуктивність і зручність написання коду [17], [18].

Середовище підтримує роботу на операційних системах Windows, macOS і Linux та інтегрується з платформами .NET Framework, .NET Core і .NET. Використання Visual Studio дозволяє розробляти як веб-додатки, так і мобільні та настільні програми, що потребують високої гнучкості та широкої сумісності [17], [18].

Інструменти Visual Studio включають підтримку налагодження, вбудовані можливості для тестування коду, а також інтеграцію з системами керування версіями, такими як Git. Крім того, в середовище вбудовані засоби автоматизації, що допомагають спростити процес розробки та пришвидшити виконання завдань [17], [18].

Завдяки підтримці численних розширень і модулів розширюється функціональність Visual Studio, що робить його ефективним рішенням для програмування під різні платформи. Інструменти для роботи з базами даних, зокрема SQL Server Management Studio, дозволяють виконувати запити, адмініструвати сховища даних та інтегрувати їх у процес розробки [17], [18].

Visual Studio надає широкий набір можливостей для створення якісного програмного забезпечення, забезпечуючи оптимізацію коду, автоматизацію процесів і підтримку сучасних методологій розробки. [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для підтримки роботи лікарні наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для підтримки роботи лікарні

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	NetBeans	Eclipse
Інтеграція з .NET	+	–	–
Розвинені інструменти для налагодження	+	+-	+
Засоби автоматичного тестування	+	+	+-
Продуктивність та швидкість роботи	+	+-	+-
Глибока інтеграція з Windows-середовищем	+	–	–

Таким чином, для створення програмного забезпечення для підтримки роботи лікарні обрано середовище розробки Visual Studio, що забезпечує

високу інтеграцію з технологіями .NET, потужні засоби налагодження, підтримку роботи з базами даних, а також зручні інструменти для роботи з графічними інтерфейсами.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для підтримки роботи лікарні обрано мову програмування C#, яка разом з середовищем Microsoft .NET забезпечує потужну інфраструктуру для створення надійних, масштабованих і захищених програмних систем. Багато лікарень використовують Windows-сервери та Microsoft SQL Server, що робить C# оптимальним вибором завдяки його чудовій інтеграції з цими технологіями. Крім того, C# підтримує об'єктно-орієнтоване програмування, що дозволяє будувати модульні та легко підтримувані програмні системи. Це особливо важливо для медичних програмних систем, які містять багато взаємопов'язаних компонентів, таких як управління електронними медичними записами, планування прийомів, управління персоналом.

Для створення програмного забезпечення для підтримки роботи лікарні обрано середовище розробки Visual Studio, що забезпечує високу інтеграцію з технологіями .NET, потужні засоби налагодження, підтримку роботи з базами даних, а також зручні інструменти для роботи з графічними інтерфейсами.

3 Опис програми

3.1 Структура програмного забезпечення для підтримки роботи лікарні

Структуру програмного забезпечення для підтримки роботи лікарні наведено на рис. 3.1.



Рисунок 3.1 – Структура програмного забезпечення для підтримки роботи лікарні

Програмне забезпечення для підтримки роботи лікарні складається з кількох взаємопов'язаних модулів, кожен із яких виконує певні функції для забезпечення ефективної роботи медичного закладу.

Основою програмної системи є модуль головної форми, який виступає центральним інтерфейсом взаємодії користувача з програмою. Через цей модуль забезпечується навігація між іншими модулями, а також доступ до ключових функцій, таких як реєстрація, призначення процедур та отримання інформації.

Важливу роль відіграє модуль входу до кабінету користувача, який відповідає за автентифікацію та авторизацію медичного персоналу й адміністрації лікарні. Він забезпечує контроль доступу до різних функціональних можливостей системи залежно від рівня прав користувача.

Для реєстрації нових пацієнтів використовується спеціальний модуль, що дозволяє вносити персональні дані, історію хвороби та контактну інформацію. Цей модуль інтегрується з базою даних, забезпечуючи швидкий пошук і збереження оновленої інформації.

Модуль роботи з даними про прийом пацієнта забезпечує управління всією інформацією, пов'язаною з відвідуванням лікарні. Його функціональність охоплює записування даних про прийом, їх збереження, оновлення та перегляд, що дає змогу медичному персоналу швидко отримувати необхідні відомості про пацієнта. У цьому модулі реєструються деталі кожного візиту, включаючи дату та час прийому, ім'я лікаря, спеціальність, причину звернення та результати обстежень. Він також містить інформацію про діагнози, призначені медикаменти та рекомендовані процедури, що дозволяє вести повну історію лікування пацієнта.

Призначення процедур реалізується за допомогою відповідного модуля, який дає змогу лікарям оформлювати призначення для пацієнтів, вказуючи

дату, тип процедур та необхідні рекомендації. Він також синхронізується з базою даних, що дозволяє автоматично оновлювати розклад медичних послуг.

Робота з даними про прийом пацієнта здійснюється через кілька модулів, що дозволяють лікарям і медичному персоналу переглядати історію відвідувань, діагнози, результати аналізів та інші важливі відомості. Інформація зберігається у централізованій базі даних, що забезпечує швидкий доступ до неї в режимі реального часу.

Модуль видачі інформації відповідає за формування звітів, виписок, направлень та інших документів, необхідних для медичного обслуговування. Він дозволяє автоматизувати процеси документообігу, зменшуючи навантаження на персонал.

Уся інформація обробляється й зберігається за допомогою модуля роботи з базою даних, який забезпечує надійне збереження та управління медичною інформацією. Він підтримує взаємодію між модулями, гарантує безпеку даних та дозволяє здійснювати їхнє резервне копіювання для запобігання втратам.

Завдяки такій структурі програмне забезпечення для лікарні забезпечує ефективну взаємодію між персоналом і пацієнтами, спрощує адміністративні процеси та підвищує якість медичних послуг.

3.2 Функціонування програмного забезпечення для підтримки роботи лікарні

Функціонування програмного забезпечення для підтримки роботи лікарні подамо за допомогою схеми, зображеної на рис. 3.2.

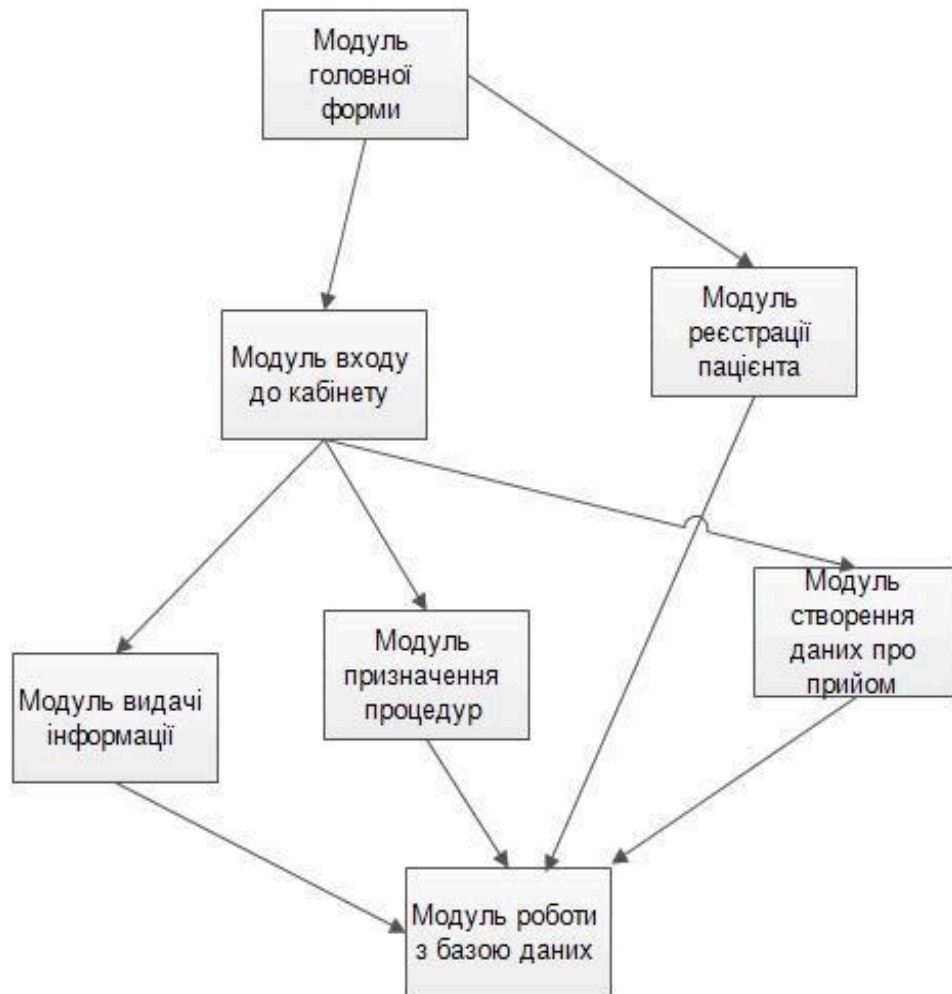


Рисунок 3.2 – Схема функціонування програмного забезпечення для підтримки роботи лікарні

Функціонування програмного забезпечення для підтримки роботи лікарні забезпечується взаємодією кількох ключових модулів, кожен з яких відповідає за певні аспекти управління медичними даними та процесами.

Робота починається з модуля входу до кабінету користувача, де медичний персонал та адміністрація проходять автентифікацію. Доступ до функціоналу залежить від рівня прав користувача, що забезпечує контроль та безпеку інформації. Після успішного входу відбувається перехід до головної форми, яка виступає централізованим інтерфейсом для навігації між основними модулями системи.

Реєстрація нових пацієнтів та збереження їхніх особистих даних здійснюється через модуль реєстрації пацієнтів. Він дозволяє фіксувати

контактну інформацію, історію хвороби та інші медичні дані, що необхідні для подальшого обслуговування. Вся інформація автоматично зберігається в модулі роботи з базою даних, забезпечуючи цілісність та доступність даних для медичного персоналу.

Призначення обстежень і процедур відбувається через відповідний модуль, який фіксує всі лікарські призначення та формує розклад процедур. Інтеграція з модулем роботи з даними про прийом пацієнта дозволяє швидко оновлювати інформацію про стан хворого, вносити результати обстежень та фіксувати нові діагнози.

Збережені дані про прийоми пацієнтів включають інформацію про лікаря, дату відвідування, призначене лікування та висновки. Цей модуль дозволяє вести повну історію лікування кожного пацієнта, а також синхронізується з іншими частинами системи для автоматичного оновлення медичних записів.

Модуль видачі інформації забезпечує доступ до звітів, статистичних даних та аналітики, що дозволяє адміністрації лікарні контролювати ефективність лікування, навантаження на лікарів та інші ключові показники роботи закладу.

Усі модулі взаємодіють із центральною базою даних, що гарантує узгодженість інформації та швидкий доступ до необхідних даних. Безпека та збереження конфіденційності пацієнтів підтримуються системою розмежування прав доступу, що дозволяє керувати інформацією лише уповноваженим користувачам.

Основними користувачами програмного забезпечення для підтримки роботи лікарні є пацієнти, лікарі та реєстратори. В залежності від типу користувача програма надає різні можливості. Функціонування програмного забезпечення для підтримки роботи лікарні для різних типів користувачів наведено на рис. 3.3.

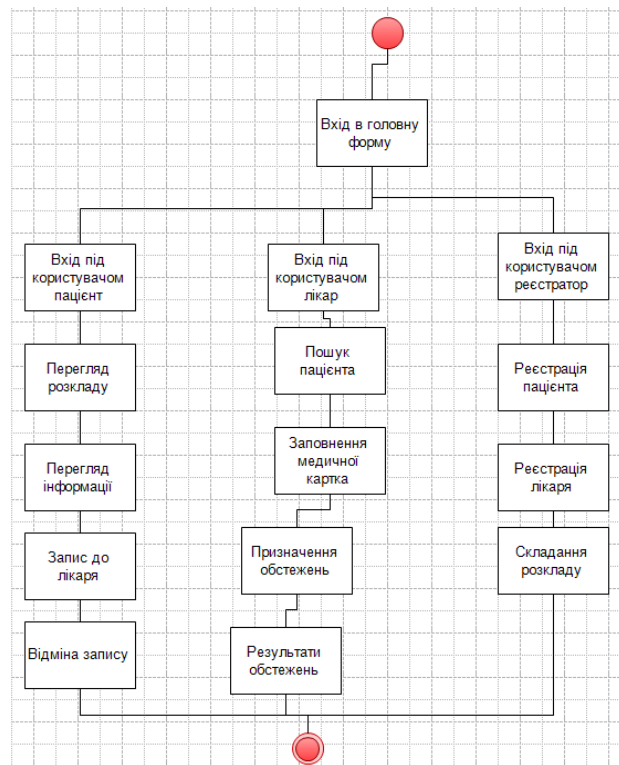


Рисунок 3.3 – Процес функціонування програмного забезпечення для різних типів користувачів

3.3 Розробка бази даних програмного забезпечення для підтримки роботи лікарні

База даних програмного забезпечення для підтримки роботи лікарні є централізованою системою, що зберігає та обробляє інформацію про пацієнтів, лікарів, прийоми, медичні записи, обстеження та розклад роботи персоналу. Схему бази даних програмного забезпечення для підтримки роботи лікарні наведено на рис. 3.4.

База даних програмного забезпечення для підтримки роботи лікарні містить кілька взаємопов'язаних таблиць, які забезпечують ефективну організацію даних та їхню швидку обробку. Базу даних розроблено за допомогою системи InterSystems Cache, що забезпечує високу продуктивність, масштабованість та підтримку об'єктно-реляційної моделі даних.

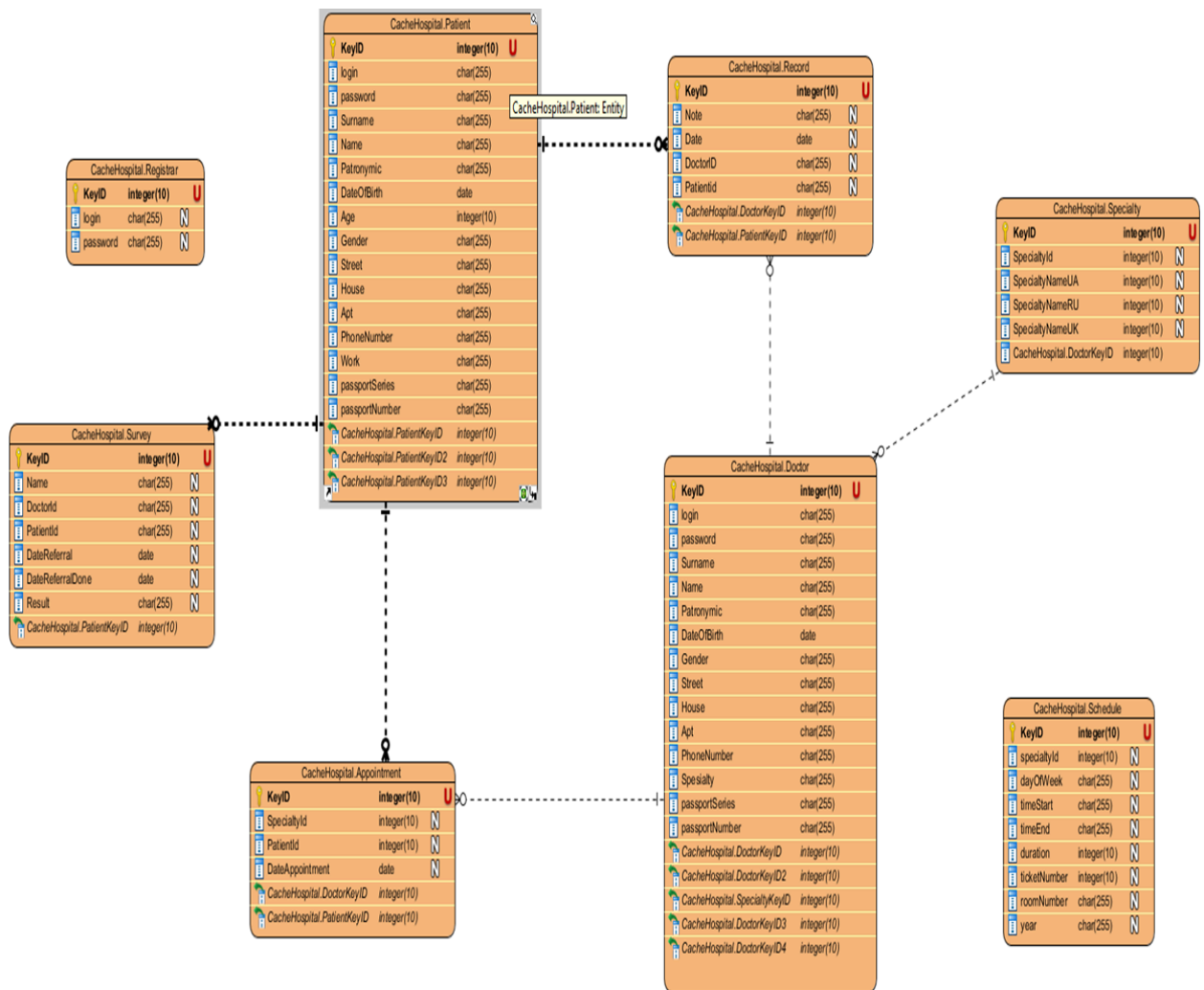


Рисунок 3.4 – Схема бази даних програмного забезпечення для підтримки роботи лікарні

Таблиця Пацієнт містить інформацію про кожного зареєстрованого відвідувача лікарні, включаючи особисті дані, адресу, контактний телефон та паспортні дані. Вона також включає логін та пароль для авторизації пацієнта в системі, що дозволяє забезпечити персоналізований доступ до електронних медичних записів та запису на прийом до лікаря.

Дані про лікарів зберігаються в таблиці Лікар. Тут міститься інформація про особисті дані медичних працівників, їхню спеціалізацію, контактні дані та паспортну інформацію. Лікарі також мають власні логін та пароль для входу в систему, що дозволяє здійснювати авторизований доступ до записів пацієнтів, внесення діагнозів та направлення на обстеження.

Кожен прийом пацієнта лікарем фіксується у таблиці Медичний запис. У ній зберігаються дані про дату прийому, лікаря, який проводив огляд, а також ідентифікатор пацієнта. Ця інформація забезпечує ведення електронної історії хвороби кожного відвідувача лікарні.

Направлення на медичні обстеження реєструються у таблиці Обстеження. Тут фіксується інформація про тип обстеження, лікаря, який виписав направлення, пацієнта, що проходить процедуру, дату направлення, дату виконання обстеження та його результати. Це дозволяє лікарям отримувати актуальні відомості про стан здоров'я пацієнтів та приймати обґрунтовані рішення щодо лікування.

Попередній запис на прийом до лікаря здійснюється через таблицю Запис на прийом. Вона містить інформацію про спеціаліста, пацієнта, а також дату та час запланованого візиту. Це дає змогу ефективно організувати робочий графік лікарів та уникнути черг у лікарні.

Таблиця Реєстратор містить облікові дані адміністративного персоналу, який здійснює керування записами пацієнтів та їх реєстрацією. Завдяки логіну та пароллю реєстратори мають можливість працювати із записами та вносити необхідні корективи.

Розклад роботи лікарів зберігається у таблиці Розклад. Вона містить інформацію про фахівця, дні тижня, коли він веде прийом, години роботи, тривалість одного прийому, кількість можливих записів на день, номер кабінету, а також рік, на який складено розклад.

Дані про спеціальності лікарів зберігаються у таблиці Спеціальність. У ній містяться унікальні ідентифікатори спеціальностей та їхні назви кількома мовами, що дозволяє адаптувати систему для багатомовного використання.

Завдяки чіткій структурованості бази даних забезпечується швидкий доступ до необхідної інформації, що дозволяє медичному персоналу ефективно працювати з даними пацієнтів, вести електронні історії хвороб, призначати процедури та керувати розкладом прийомів.

3.4 Проектування інтерфейсу програмного забезпечення для підтримки роботи лікарні

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки роботи лікарні виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для підтримки роботи лікарні враховані функціональні вимоги до програми:

- підтримка можливості авторизації та реєстрації користувачів ПЗ лікарні;
- підтримка можливості отримання пацієнтом інформації про медичні обстеження;
- можливість вибору мови інтерфейсу користувача;
- можливість додавання нових лікарів до системи;
- підтримка можливості складання розкладу роботи лікарів;
- підтримка можливості запису на прийом до лікарів на вільний час згідно з їх розкладом роботи.

Інтерфейс головної форми програмного забезпечення для підтримки роботи лікарні наведено на рис. 3.5.

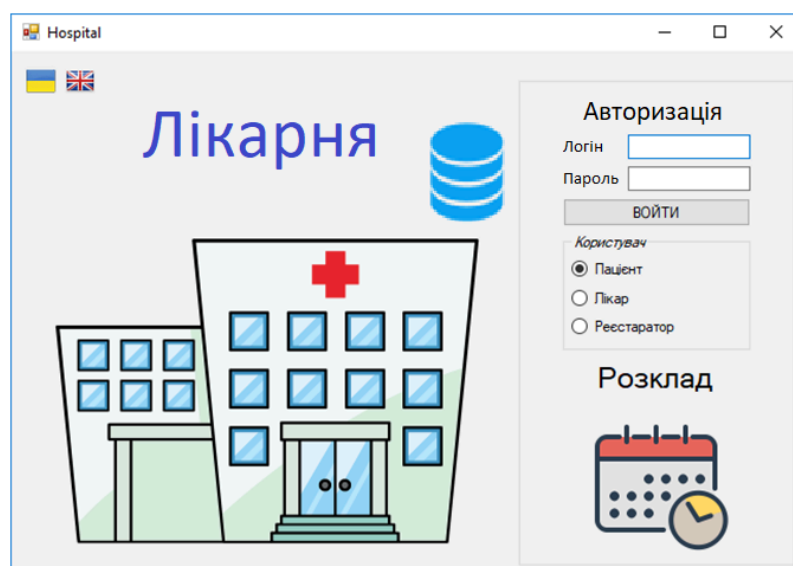


Рисунок 3.5 – Інтерфейс головної форми програмного забезпечення для підтримки роботи лікарні

Приклад вікна з обстеженнями користувача пацієнта наведено на рис. 3.6.

The screenshot shows a software window titled "Medical card" with a standard Windows interface (minimize, maximize, close buttons). The window has a tabbed interface with four tabs: "Main information", "Medical records", "Surveys", and "Appointments". The "Surveys" tab is currently selected. On the left side of the "Surveys" tab, there is a large blue circular icon containing a white clipboard with a pencil. To the right of this icon, there are two status indicators: a green checkmark icon next to the text "Count ending surveys 2" and a yellow checkmark icon next to the text "Count active surveys 1". Below these indicators, there are two sub-tabs: "Active surveys" and "Done surveys". The "Active surveys" sub-tab is selected. It contains a table with the following data:

Date of survey	02.04.2025
Name of survey	Загальний аналіз сечі
Name of doctor	Скорін Сергій Олександрович
Date finish survey	03.04.2025

Below the table, there is a section labeled "Result" containing the text:

гемоглобін - 40
билирубін - 30

At the bottom of the window, there is a navigation bar. On the left, it shows "1 / 2". On the right, there are four navigation buttons: a left arrow, a double left arrow, a double right arrow, and a right arrow.

Рисунок 3.6 – Приклад вікна з обстеженнями користувача пацієнта

Приклад вікна користувача з правами лікаря наведено на рис. 3.7.

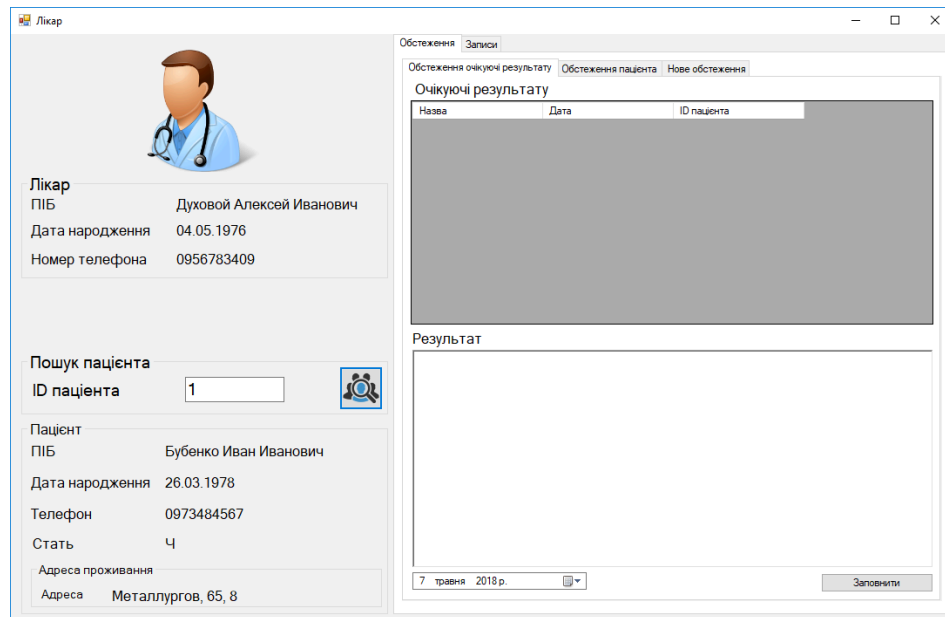


Рисунок 3.7 – Приклад вікна користувача з правами лікаря
Вікно користувача з правами адміністратора наведено на рис. 3.8.

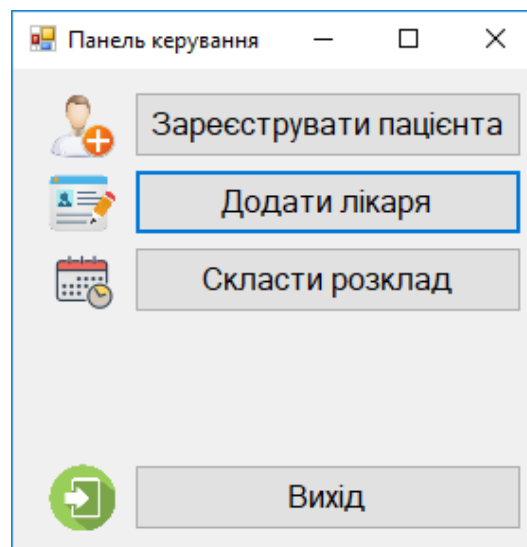


Рисунок 3.8 – Вікно користувача з правами адміністратора
Вікно складання розкладу роботи лікаря наведено на рис. 3.9.

Складання розкладу

Складання розкладу прийому лікарів

Лікар

№ кабінету

Робочі дні прийому

☐ Понеділок ☐ Четвер

☒ Вівторок ☐ П'ятниця

☐ Середа

Дні тижня

Понеділок

Початковий час

Кінцевий час

Четвер

Початковий час

Кінцевий час

Вівторок

Початковий час

Кінцевий час

П'ятниця

Початковий час

Кінцевий час

Середа

Початковий час

Кінцевий час

Рисунок 3.9 – Вікно складання розкладу роботи лікаря

Вікно зі створеним розкладом роботи лікарів наведено на рис. 3.10.

Кабінет	Спеціаліст	Понеділок	Вівторок	Середа	Четвер	П'ятниця
5	Офтальмолог	-	-	10 - 15	-	-
5	Терапевт	8 - 17	-	-	-	-
6	УЗИ-спеціаліст	10 - 12	-	-	-	-
5	Невролог	11 - 14	-	-	-	-
6	Хірург	11 - 18	-	-	-	-
7	Дерматолог	8 - 16	8 - 16	8 - 16	8 - 16	8 - 16
7	Гінеколог	8 - 16	-	-	8 - 16	8 - 16
7	Отоларинголог	-	11 - 15	-	-	-
9	Алерголог	-	-	-	-	8 - 18

Рисунок 3.10 – Вікно зі створеним розкладом роботи лікарів

Вікно запису пацієнта на прийом до лікаря наведено на рис. 3.11.

Запис на прийом до лікаря

Спеціаліст



- Алерголог
- Гінеколог
- Дерматолог
- Кардіолог
- Невролог
- Офтальмолог**
- Онколог

Дата та час



червень

Пн	Вт	Ср	Чт	Пт	Сб	Нд
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

Примітка

Немає запису

Час зайнятий

Час вільний



Тривалість прийому - 15 хвилин

27 червня

8.00 - 8.15	10.00 - 10.15	12.00 - 12.15	14.00 - 14.15	16.00 - 16.15
8.15 - 8.30	10.15 - 10.30	12.15 - 12.30	14.15 - 14.30	16.15 - 16.30
8.30 - 8.45	10.30 - 10.45	12.30 - 12.45	14.30 - 14.45	16.30 - 16.45
8.45 - 9.00	10.45 - 11.00	12.45 - 13.00	14.45 - 15.00	16.45 - 17.00
9.00 - 9.15	11.00 - 11.15	13.00 - 13.15	15.00 - 15.15	17.00 - 17.15
9.15 - 9.30	11.15 - 11.30	13.15 - 13.30	15.15 - 15.30	17.15 - 17.30
9.30 - 9.45	11.30 - 11.45	13.30 - 13.45	15.30 - 15.45	17.30 - 17.45
9.45 - 10.00	11.45 - 12.00	13.45 - 14.00	15.45 - 16.00	17.45 - 18.00

Рисунок 3.11 – Вікно запису пацієнта на прийом до лікаря

Вікно адміністратора для реєстрації пацієнта наведено на рис. 3.12.

Рисунок 3.12 – Вікно адміністратора для реєстрації пацієнта

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для підтримки роботи лікарні.

Запропоновано структуру програмного забезпечення для підтримки роботи лікарні. Визначено, що програмне забезпечення для підтримки роботи лікарні складається з кількох взаємопов'язаних модулів (модуль головної форми, модуль входу до кабінету користувача, модуль реєстрації нових пацієнтів, модуль роботи з даними про прийом пацієнта, модуль призначення процедур, модуль видачі інформації, модуль роботи з базою даних), кожен із яких виконує певні функції для забезпечення ефективної роботи медичного закладу.

Розроблено базу даних програмного забезпечення для підтримки роботи лікарні, яка є централізованою системою, що зберігає та обробляє інформацію про пацієнтів, лікарів, прийоми, медичні записи, обстеження та розклад роботи персоналу.

Описано функціонування програмного забезпечення для підтримки роботи лікарні.

Описано особливості реалізації програмного забезпечення для підтримки роботи лікарні. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки роботи лікарні.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки роботи лікарні. Програма написана на мові C#, яка разом з середовищем Microsoft .NET забезпечує потужну інфраструктуру для створення надійних, масштабованих і захищених програмних систем. В якості середовища розробки для створення програмного забезпечення для підтримки роботи лікарні обрано Visual Studio, що забезпечує високу інтеграцію з технологіями .NET, потужні засоби налагодження, підтримку роботи з базами даних, а також зручні інструменти для роботи з графічними інтерфейсами.

Програмне забезпечення для підтримки роботи лікарні забезпечує виконання таких функцій:

- підтримка можливості авторизації та реєстрації користувачів ПЗ лікарні;
- підтримка можливості отримання пацієнтом інформації про медичні обстеження;
- можливість вибору мови інтерфейсу користувача;
- можливість додавання нових лікарів до системи;
- підтримка можливості складання розкладу роботи лікарів;
- підтримка можливості запису на прийом до лікарів на вільний час згідно з їх розкладом роботи.

Для роботи програмного забезпечення для підтримки роботи лікарні потрібні такі технічні та програмні ресурси. Серверна частина потребує продуктивного процесора, який має щонайменше 4 фізичних ядра з тактовою частотою від 3.0 ГГц або більше. Оперативна пам'ять повинна становити не менше 16 ГБ для базової конфігурації та 32 ГБ і більше для великої лікарні з інтенсивним потоком даних. Це дозволить швидко завантажувати необхідні дані та підтримувати ефективну роботу бази даних. Жорсткий диск або

система зберігання даних має мати SSD-накопичувач об'ємом від 512 ГБ для малого закладу та від 2 ТБ для великої лікарні. Операційна система повинна бути надійною та підтримувати роботу серверних додатків. Для роботи з базою даних необхідне програмне забезпечення, що підтримує великі обсяги інформації та швидкий доступ, зокрема, InterSystems Caché, що забезпечує масштабованість та надійність.

Клієнтські комп'ютери потребують процесора з 2 або більше ядрами, оперативної пам'яті від 8 ГБ, SSD на 256 ГБ та екрану з роздільною здатністю не менше 1920x1080 пікселів. Це дозволяє комфортно працювати з медичними записами та швидко отримувати необхідну інформацію.

Також необхідно мати надійне мережеве підключення зі швидкістю не менше 100 Мбіт/с для внутрішньої мережі та доступу до зовнішніх ресурсів, а також систему резервного копіювання, що дозволяє уникнути втрати критично важливих даних.

Функціональні характеристики розробленого програмного забезпечення для підтримки роботи лікарні наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для підтримки роботи лікарні наведено у додатку Б.

4.2 Характеристики програми для підтримки роботи лікарні

Програмне забезпечення для підтримки роботи лікарні характеризується високою продуктивністю, гнучкістю та масштабованістю, що забезпечує ефективну обробку та збереження великої кількості медичних даних. Завдяки підтримці багатокористувацького доступу, система дозволяє одночасно працювати багатьом співробітникам, включаючи лікарів, реєстраторів та адміністрацію.

Функціональність програмного забезпечення охоплює управління записами пацієнтів, створення медичних карток, обробку результатів

обстежень та контроль розкладу прийому лікарів. Завдяки централізованій базі даних забезпечується швидкий доступ до інформації, що підвищує якість медичного обслуговування. Інтерфейс розроблено з урахуванням зручності використання, що дозволяє медичному персоналу легко орієнтуватися в системі та швидко виконувати необхідні операції.

Програмне забезпечення підтримує інтеграцію з електронними медичними записами, лабораторними системами та іншими медичними сервісами, що сприяє комплексному веденню пацієнтів. Безпека даних гарантується механізмами автентифікації, шифрування та розмежування прав доступу, що запобігає несанкціонованому використанню інформації.

Взаємодія з користувачами здійснюється через вебінтерфейс або спеціалізовані клієнтські додатки, що забезпечує доступ до системи з різних пристроїв. Завдяки підтримці хмарних технологій можливе збереження та обробка даних на віддалених серверах, що зменшує навантаження на локальні ресурси та забезпечує резервне копіювання інформації.

Програмне забезпечення характеризується високим рівнем надійності та стабільності роботи, що забезпечується оптимізованими алгоритмами обробки запитів та ефективним управлінням ресурсами. Система підтримує можливість розширення функціоналу відповідно до змін у вимогах медичних закладів, що дозволяє адаптувати її до конкретних потреб користувачів.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для підтримки роботи лікарні на серверній частини необхідно запустити відповідний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для підтримки роботи лікарні є інформація про бажану дату та час запису на прийом до лікаря, реєстраційні дані користувача та ін. Вихідними даними програмного забезпечення для підтримки роботи лікарні є інформація про доступні часові межі прийому лікаря, інформація про направлення на обстеження пацієнта, таблиця з загальним розкладом лікарів медичного закладу та ін.

4.3.3 Повідомлення

Користувач програмного забезпечення для підтримки роботи лікарні може отримати такі повідомлення: повідомлення про успішну реєстрацію, повідомлення про успішний запис до лікаря, повідомлення про проблеми зі входом до системи та ін.

4.4 Виконання програмного забезпечення для підтримки роботи лікарні

Після запуску програмного забезпечення для підтримки роботи лікарні користувачу відображається головна сторінка (рис. 4.1), де користувач має можливість увійти у систему та перейти до розкладу роботи лікарів медичного закладу. Також користувач може обрати мову інтерфейсу (українську або англійську, натиснувши на значок з відповідним прапором).

Користувач з правами пацієнта після входу має можливість доступу до чотирьох вкладок: основна інформація про пацієнта, медичні записи про пацієнта, обстеження та керування записом на прийом до лікарів. Приклад вікна з обстеженнями користувача пацієнта наведено на рис. 4.2.

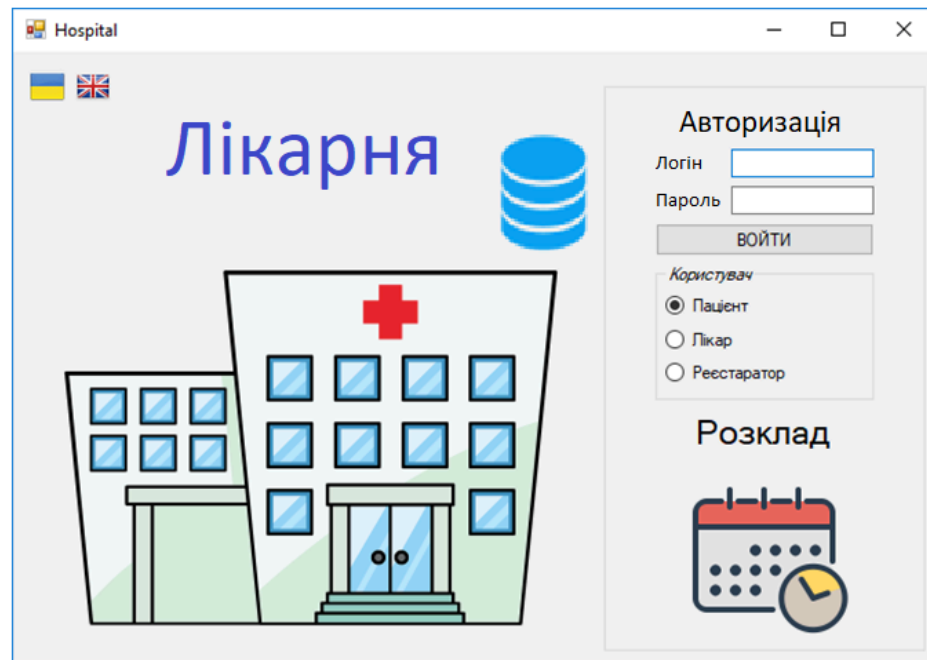


Рисунок 4.1 – Інтерфейс головної сторінки програмного забезпечення для підтримки роботи лікарні

Count ending surveys 2

Count active surveys 1

Active surveys	
Date of survey	02.04.2025
Name of survey	Загальний аналіз сечі
Name of doctor	Скорін Сергій Олександрович
Date finish survey	03.04.2025

Result

гемоглобін - 40
билирубін - 30

1 / 2

Рисунок 4.2 – Приклад вікна з обстеженнями користувача пацієнта

Користувач з правами лікаря після входу має можливість доступу до двох вкладок (рис. 4.3): медичні записи та обстеження пацієнтів. Лікар має можливість пошуку пацієнтів у базі даних за допомогою відповідного поля пошуку.

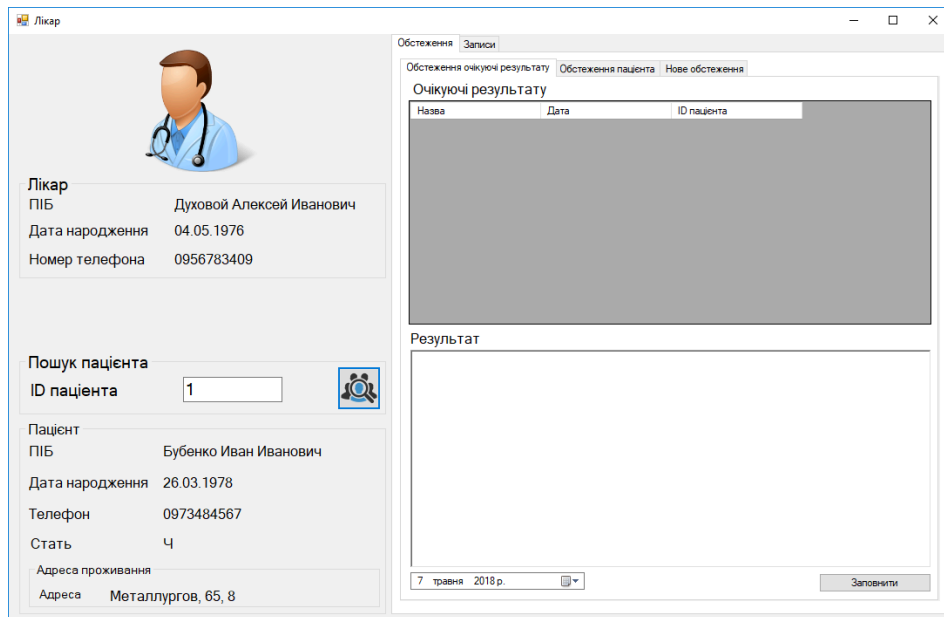


Рисунок 4.3 – Приклад вікна користувача з правами лікаря

Користувач з правами адміністратора лікарні після входу має можливість створення та редагування розкладу роботи лікарів медичного закладу, додавання нового лікаря у систему та можливість реєстрації пацієнтів (рис. 4.4).

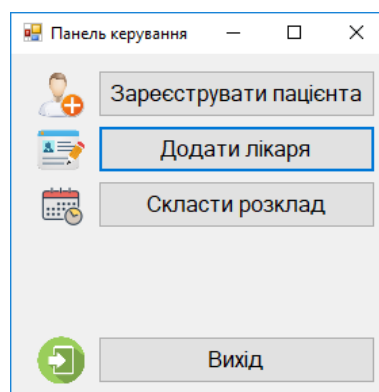


Рисунок 4.4 – Приклад вікна користувача з правами адміністратора

При натисканні на кнопку Скласти розклад відбувається перехід до відповідного вікна (рис. 4.5), де адміністратор обирає зі списку профіль лікаря (наприклад, офтальмолог), обирає робочі дні та вводить початковий та кінцевий час роботи.

Складання розкладу

Складання розкладу прийому лікарів

Лікар: Офтальмолог

№ кабінету: 6а

Робочі дні прийому

☐ Понеділок ☐ Четвер

☒ Вівторок ☐ П'ятниця

☐ Середа

Дні тижня

Понеділок

Початковий час: []

Кінцевий час: []

Вівторок

Початковий час: 9

Кінцевий час: 16

Середа

Початковий час: []

Кінцевий час: []

Четвер

Початковий час: []

Кінцевий час: []

П'ятниця

Початковий час: []

Кінцевий час: []

Скласти розклад

Рисунок 4.5 – Вікно складання розкладу роботи лікаря

Вікно зі створеним розкладом роботи лікарів наведено на рис. 4.6.

Розклад	Кабінет	Спеціаліст	Понеділок	Вівторок	Середа	Четвер	П'ятниця
	5	Офтальмолог	-	-	10 - 15	-	-
	5	Терапевт	8 - 17	-	-	-	-
	6	УЗИ-спеціаліст	10 - 12	-	-	-	-
	5	Невролог	11 - 14	-	-	-	-
	6	Хірург	11 - 18	-	-	-	-
	7	Дерматолог	8 - 16	8 - 16	8 - 16	8 - 16	8 - 16
	7	Гінеколог	8 - 16	-	-	8 - 16	8 - 16
	7	Отоларинголог	-	11 - 15	-	-	-
	9	Алерголог	-	-	-	-	8 - 18

Рисунок 4.6 – Вікно зі створеним розкладом роботи лікарів

Користувач з правами пацієнта має можливість зареєструватися на прийом до лікаря відповідно до складеного розкладу (рис. 4.7).

Запис на прийом до лікаря

Спеціаліст

Алерголог
Гінеколог
Дерматолог
Кардіолог
Невролог
Офтальмолог
Онколог

Дата та час

червень

Пн Вт Ср Чт Пт Сб Нд

28 29 30 31 1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 1

2 3 4 5 6 7 8

27 червня

8:00 - 8:15	10:00 - 10:15	12:00 - 12:15	14:00 - 14:15	16:00 - 16:15
8:15 - 8:30	10:15 - 10:30	12:15 - 12:30	14:15 - 14:30	16:15 - 16:30
8:30 - 8:45	10:30 - 10:45	12:30 - 12:45	14:30 - 14:45	16:30 - 16:45
8:45 - 9:00	10:45 - 11:00	12:45 - 13:00	14:45 - 15:00	16:45 - 17:00
9:00 - 9:15	11:00 - 11:15	13:00 - 13:15	15:00 - 15:15	17:00 - 17:15
9:15 - 9:30	11:15 - 11:30	13:15 - 13:30	15:15 - 15:30	17:15 - 17:30
9:30 - 9:45	11:30 - 11:45	13:30 - 13:45	15:30 - 15:45	17:30 - 17:45
9:45 - 10:00	11:45 - 12:00	13:45 - 14:00	15:45 - 16:00	17:45 - 18:00

Примітка

Немає запису Час зайнятий Час вільний

Тривалість прийому - 15 хвилин

Рисунок 4.7 – Вікно запису пацієнта на прийом до лікаря

Адміністратор програмного забезпечення для підтримки роботи лікарні має можливість реєстрації нових пацієнтів у системі (рис. 4.8).

Реєстрація пацієнта

Логін

Пароль

Повторіть пароль

Прізвище

Ім'я

По батькові

Стать
☐ Ч ☐ Ж

Дата народження 7 травня

Номер телефона

Адреса проживання

Вулиця

Будинок

Квартира

Паспортні дані

Серія

Номер

Місце роботи

Відміна Зареєструвати

Рисунок 4.8 – Вікно адміністратора для реєстрації пацієнта

4.5 Тестування програмного забезпечення для підтримки роботи лікарні

Виконано тестування програмного забезпечення для підтримки роботи лікарні.

Виявлено, що програма для підтримки роботи лікарні функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки роботи лікарні.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою роботи лікарні.

4.6 Висновки за розділом 4

Описано програмне забезпечення для підтримки роботи лікарні. Виконано тестування розробленого програмного забезпечення для підтримки роботи лікарні. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою роботи лікарні.

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для підтримки функціонування медичних закладів.

Відзначено, що програмне забезпечення для управління діяльністю лікарень відіграє ключову роль у забезпеченні безперервності медичного обслуговування. Його функціональність охоплює обробку та збереження інформації про пацієнтів, управління медичними записами, координацію процесів лікування та автоматизацію рутинних завдань, що підвищує ефективність роботи персоналу.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для підтримки роботи медичних закладів. Проте деякі програми для підтримки роботи лікарні мають суттєві недоліки, які можуть ускладнювати їх впровадження та ефективне використання. Однією з головних проблем є висока вартість як початкового впровадження, так і подальшого обслуговування. Лікарням доводиться витратити значні кошти на придбання програмного забезпечення, налаштування програмної системи відповідно до власних потреб, а також на регулярні оновлення та підтримку. Ще одним недоліком є складність навчання персоналу, особливо якщо мова йде про медичних працівників, які не мають досвіду роботи з подібними технологіями. Це може призводити до помилок у введенні даних, що в свою чергу впливає на якість діагностики та лікування. Додатково, лікарняні інформаційні системи часто не є універсальними, а їхня інтеграція з іншими медичними або державними реєстрами може бути складною або навіть неможливою без значних модифікацій. Окрім цього, деякі медичні заклади стикаються з опором змін з боку персоналу, який звик працювати за старими методами. Це може спричинити додаткові труднощі при впровадженні нових технологій та викликати небажання використовувати

систему у повному обсязі. Крім того, деякі готові програмні рішення не завжди повністю відповідають потребам конкретної лікарні. Це змушує медичні установи або шукати компроміси, або витрачати додаткові ресурси на розробку спеціалізованих рішень, також ускладнює їхнє впровадження та супровід. Тому актуальною є розробка програмного забезпечення для підтримки роботи лікарні.

Сформульовано функціональні вимоги до програмного забезпечення для підтримки роботи лікарні.

Для реалізації програмного забезпечення для підтримки роботи лікарні обрано мову програмування C#, яка разом з середовищем Microsoft .NET забезпечує потужну інфраструктуру для створення надійних, масштабованих і захищених програмних систем. Багато лікарень використовують Windows-сервери та Microsoft SQL Server, що робить C# оптимальним вибором завдяки його чудовій інтеграції з цими технологіями. Крім того, C# підтримує об'єктно-орієнтоване програмування, що дозволяє будувати модульні та легко підтримувані програмні системи. Це особливо важливо для медичних програмних систем, які містять багато взаємопов'язаних компонентів, таких як управління електронними медичними записами, планування прийомів, управління персоналом.

Для створення програмного забезпечення для підтримки роботи лікарні обрано середовище розробки Visual Studio, що забезпечує високу інтеграцію з технологіями .NET, потужні засоби налагодження, підтримку роботи з базами даних, а також зручні інструменти для роботи з графічними інтерфейсами.

Розроблено програмне забезпечення для підтримки роботи лікарні.

Запропоновано структуру програмного забезпечення для підтримки роботи лікарні. Визначено, що програмне забезпечення для підтримки роботи лікарні складається з кількох взаємопов'язаних модулів (модуль головної форми, модуль входу до кабінету користувача, модуль реєстрації нових пацієнтів, модуль роботи з даними про прийом пацієнта, модуль призначення

процедур, модуль видачі інформації, модуль роботи з базою даних), кожен із яких виконує певні функції для забезпечення ефективної роботи медичного закладу.

Розроблено базу даних програмного забезпечення для підтримки роботи лікарні, яка є централізованою системою, що зберігає та обробляє інформацію про пацієнтів, лікарів, прийоми, медичні записи, обстеження та розклад роботи персоналу.

Описано функціонування програмного забезпечення для підтримки роботи лікарні.

Описано особливості реалізації програмного забезпечення для підтримки роботи лікарні. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для підтримки роботи лікарні.

Виконано тестування розробленого програмного забезпечення для підтримки роботи лікарні. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою роботи лікарні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. What is a Hospital Management System and How is HMS Good for Businesses [Electronic resource]. – Access mode: <https://smartmedhx.ai/trends/what-is-a-hospital-management-system/>.
2. A Complete Guide to Hospital Management Software [Electronic resource]. – Access mode: https://www.linkedin.com/pulse/complete-guide-hospital-management-software-stacknova-globalcorp-ojhtc?trk=public_post.
3. Hospital Management Software Development [Electronic resource]. – Access mode: <https://appinventiv.com/blog/how-to-build-hospital-management-system-and-its-cost/>.
4. Everything You Need to Know About Hospital Management Software [Electronic resource]. – Access mode: <https://infostride.com/hospital-management-software-guide/>.
5. Types of Healthcare Software [Electronic resource]. – Access mode: <https://www.netguru.com/blog/healthcare-software-types>.
6. An overview of Hospital Management System [Electronic resource]. – Access mode: <https://www.emedicaresoft.com/news-and-blog/view/an-overview-of-hospital-management-system>.
7. Types of Healthcare Software [Electronic resource]. – Access mode: <https://www.pragmaticcoders.com/blog/trending-types-of-medical-software>.
8. A detailed view of Hospital Management System [Electronic resource]. – Access mode: <https://mocdoc.com/blog/a-detailed-view-of-hospital-management-system-hms>.
9. Hospital Management Software [Electronic resource]. – Access mode: <https://www.capterra.com/hospital-management-software/>.

10. Top 7 Hospital Management Software [Electronic resource]. – Access mode: <https://safetyculture.com/app/hospital-management-software/>.
11. Hospital Management Software [Electronic resource]. – Access mode: <https://www.appvizer.co.uk/health/hospital-mgt>.
12. Caspio [Electronic resource]. – Access mode: <https://www.caspio.com/>.
13. TimeTap [Electronic resource]. – Access mode: <https://www.timetap.com/>.
14. SoftClinic [Electronic resource]. – Access mode: <https://www.softclinicsoftware.com/>.
15. C#. The modern, innovative, open-source programming language for building all your apps. [Electronic resource]. – Access mode: <https://dotnet.microsoft.com/en-us/languages/csharp>.
16. A tour of the C# language [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>.
17. Introduction to Visual Studio [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/introduction-to-visual-studio/>.
18. What is Visual Studio? [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide>.

Додаток А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки роботи лікарні.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення підтримки роботи лікарні», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для підтримки роботи лікарні.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для підтримки роботи лікарні забезпечує виконання таких функцій:

- підтримка можливості авторизації та реєстрації користувачів ПЗ лікарні;
- підтримка можливості отримання пацієнтом інформації про медичні обстеження;
- можливість вибору мови інтерфейсу користувача;
- можливість додавання нових лікарів до системи;
- підтримка можливості складання розкладу роботи лікарів;

– підтримка можливості запису на прийом до лікарів на вільний час згідно з їх розкладом роботи.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для підтримки роботи лікарні повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення для підтримки роботи лікарні повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для підтримки роботи лікарні необхідна наявність персонального комп'ютера.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для підтримки роботи лікарні потрібні такі технічні та програмні ресурси. Серверна частина потребує продуктивного процесора, який має щонайменше 4 фізичних ядра з тактовою частотою від 3.0 ГГц або більше. Оперативна пам'ять повинна становити не менше 16 ГБ для базової конфігурації та 32 ГБ і більше для великої лікарні з інтенсивним потоком даних. Це дозволить швидко завантажувати необхідні дані та підтримувати ефективну роботу бази даних. Жорсткий диск або система зберігання даних має мати SSD-накопичувач об'ємом від 512 ГБ для

малого закладу та від 2 ТБ для великої лікарні. Операційна система повинна бути надійною та підтримувати роботу серверних додатків. Для роботи з базою даних необхідне програмне забезпечення, що підтримує великі обсяги інформації та швидкий доступ, зокрема, InterSystems Caché, що забезпечує масштабованість та надійність.

Клієнтські комп'ютери потребують процесора з 2 або більше ядрами, оперативної пам'яті від 8 ГБ, SSD на 256 ГБ та екрану з роздільною здатністю не менше 1920x1080 пікселів. Це дозволяє комфортно працювати з медичними записами та швидко отримувати необхідну інформацію.

A.3.6 Вимоги до маркування і пакування

Програма для підтримки роботи лікарні може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для підтримки роботи лікарні».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для підтримки роботи лікарні є інформація про бажану дату та час запису на прийом до лікаря, реєстраційні дані користувача та ін.

Вихідними даними програмного забезпечення для підтримки роботи лікарні є інформація про доступні часові межі прийому лікаря, інформація про направлення на обстеження пацієнта, таблиця з загальним розкладом лікарів медичного закладу та ін.

Додаток Б
Фрагмент тексту програми

```

namespace CLikarnya
{
    public partial class MainForm : Form
    {
        string mova = "UA";
        string pomylkavvodu = "Помилка паролю та логіну!";
        string oshbk = "Помилка";
        string usph = "Успішний вхід у програму";
        string lgn = "Вхід";
        string wlcm = "Ви у системі";

        public MainForm()
        {
            InitializeComponent();
            txtBxPsswrд.PswrdChar = '*';
        }

        private void buttonLgn_Click(object sender, EventArgs e)
        {
            if (rdBttnPtnt.Checked)
            {
                try
                {
                    ChKnction ChKnct = new ChKnction();
                    ChKnct.ConnectionString =
ChKnctionInfo.Server
                        + ChKnctionInfo.Port +
ChKnctionInfo.Namespace
                        + ChKnctionInfo.Pswrd +
ChKnctionInfo.KrstvchID;
                    ChKnct.Open();
                    int id = 1;
                    CLikarnya.Ptnrt ptnrt =
CLikarnya.Ptnrt.OpenId(ChKnct, id.ToString());
                    while(CLikarnya.Ptnrt.ExistsId(ChKnct,
id.ToString()) == true)
                    {
                        if (txtBxK.Text == ptnrt.lgn &&
txtBxPsswrд.Text == ptnrt.Pswrd)
                        {
                            MessageBox.Show(wlcm + ptnrt.Name + "
" + ptnrt.Prizvch + "!");
                            PtnrtCard ptnrtCard = new
PtnrtCard(id, mova);
                            ptnrtCard.Show();
                            ptnrt.Close();
                            ChKnct.Close();
                            return;

```

```

        }
        id++;
        ptnrt = CLikarnya.Ptnrt.OpenId(ChKnct,
id.ToString());
    }
    MessageBox.Show(pomylkavvodu, oshbk,
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    catch (Exception eKn)
    {
        MessageBox.Show("Помилка StvorZvyazk :" +
eKn.Message);
    }
}
if (rdBtnDctr.Checked)
{
    try
    {
        ChKnction ChKnct = new ChKnction();
        ChKnct.ConnectionString =
ChKnctionInfo.Server
            + ChKnctionInfo.Port +
ChKnctionInfo.Namespace
            + ChKnctionInfo.Pswrd +
ChKnctionInfo.KrstvchID;
        ChKnct.Open();
        int id = 1;
        CLikarnya.Dktr dktr =
CLikarnya.Dktr.OpenId(ChKnct, id.ToString());
        while (CLikarnya.Dktr.ExistsId(ChKnct,
id.ToString()) == true)
        {
            if (txtBxK.Text == dktr.lgn &&
txtBxPswrd.Text == dktr.Pswrd)
            {
                MessageBox.Show(wlcm + dktr.Name + "
" + dktr.Prizvch + "!");
                DktrCard dktrCard = new DktrCard(id,
mova);
                dktrCard.Show();
                dktr.Close();
                ChKnct.Close();
                return;
            }
            id++;
            dktr = CLikarnya.Dktr.OpenId(ChKnct,
id.ToString());
        }
    }
}

```

```

        MessageBox.Show(pomylkavvodu, oshbk,
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    catch (Exception eKn)
    {
        MessageBox.Show("Помилка StvorZvyazk :" +
eKn.Message);
    }
}
if (radioButtonRegistrar.Checked)
{
    try
    {
        ChKnction ChKnct = new ChKnction();
        ChKnct.ConnectionString =
ChKnctionInfo.Server
            + ChKnctionInfo.Port +
ChKnctionInfo.Namespace
            + ChKnctionInfo.Pswrd +
ChKnctionInfo.KrstvchID;
        ChKnct.Open();
        CLikarnya.Registrar admin =
CLikarnya.Registrar.OpenId(ChKnct, "1");
        if (txtBxK.Text == admin.lgn &&
txtBxPswrd.Text == admin.Pswrd)
        {
            MessageBox.Show(usph, lgn);
            registration rgFrm = new
registration(this, mova);
            rgFrm.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show(pomylkavvodu, oshbk,
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
        admin.Close();
        ChKnct.Close();
    }
    catch (Exception eKn)
    {
        MessageBox.Show("Помилка StvorZvyazk :" +
eKn.Message);
    }
}
}

private void pctrBxUA_Click(object sender, EventArgs e)

```

```

{
    mova = "UA";
    pomylkavvodu = "Помилка паролю та логіну!";
    oshbk = "Помилка";
    usph = "Успішний вхід у програму";
    lgn = "Вхід";
    wlcm = "Ви у системі";
    title.Text = "Поліклініка";
    lblAuthorization.Text = "Авторизація";
    lblLgn.Text = "Логін";
    lblPswrd.Text = "Пароль";
    lblSchdl.Text = "Розклад";
    buttonLgn.Text = "ВХІД";
    groupBoxUser.Text = "Користувач";
    rdBtnPtnt.Text = "Пацієнт";
    rdBtnDctr.Text = "Лікар";
    radioButtonRegistrar.Text = "Реєстратор";
}

private void pctrBxRU_Click(object sender, EventArgs e)
{
    mova = "RU";
    pomylkavvodu = "Некоректні логін та пароль!";
    oshbk = "Помилка";
    usph = "Вхід виконано";
    lgn = "Вхід";
    wlcm = "Вітаємо";
    title.Text = "Лікарня";
    lblAuthorization.Text = "Авторизація";
    lblLgn.Text = "Логін";
    lblPswrd.Text = "Пароль";
    lblSchdl.Text = "Розклад";
    buttonLgn.Text = "Вхід";
    groupBoxUser.Text = "Користувач";
    rdBtnPtnt.Text = "Пацієнт";
    rdBtnDctr.Text = "Лікар";
    radioButtonRegistrar.Text = "Реєстратор";
}

private void pctrBxUK_Click(object sender, EventArgs e)
{
    mova = "UK";
    pomylkavvodu = "Incorrect lgn and Pswrd
combination!";
    oshbk = "Oshbk";
    usph = "Sign in successfully";
    lgn = "Lgn";
    wlcm = "Wlcm ";
    title.Text = "Likarnya";

```

```

        lblAuthorization.Text = "Авторизація";
        lblLgn.Text = "Lgn";
        lblPswrd.Text = "Pswrd";
        lblSchdl.Text = "Rozkld";
        buttonLgn.Text = "LGN";
        groupBoxUser.Text = "User";
        rdBtnPtnt.Text = "Ptnrt";
        rdBtnDctr.Text = "Dktr";
        radioButtonRegistrar.Text = "Реєстрація";
    }

    private void pctrBxTimeTable_Click(object sender,
EventArgs e)
    {
        TimeTable timetable = new TimeTable(mova);
        timetable.Show();
    }

    private void pctrBxTimeTable_MouseEnter(object sender,
EventArgs e)
    {
        pctrBxTimeTable.Image =
        CLikarnya.Properties.Resources.RozkldActive;
    }

    private void pctrBxTimeTable_MouseLeave(object sender,
EventArgs e)
    {
        pctrBxTimeTable.Image =
        CLikarnya.Properties.Resources.Rozkld;
    }

    private void buttonLgn_MouseEnter(object sender,
EventArgs e)
    {
        Likarnya.Image =
        CLikarnya.Properties.Resources.Likarnya2;
    }

    private void buttonLgn_MouseLeave(object sender,
EventArgs e)
    {
        Likarnya.Image =
        CLikarnya.Properties.Resources.Likarnya;
    }
}

public partial class PtnrtCard : Form
{

```

```

List<string> rcrdsNote = new List<string>();
List<string> rcrdsDate = new List<string>();
List<string> rcrdsDktr = new List<string>();
int rcrdNmbr = 0;
int srvNew = 0;
int srvDone = 0;
int srvNewCurrent = 1;
int srvDoneCurrent = 1;
List<string> srvNewName = new List<string>();
List<DateTime> srvNewDateReferral = new List<DateTime>();
List<string> srvNewDktr = new List<string>();
List<string> srvDoneName = new List<string>();
List<DateTime> srvDoneDateReferral = new
List<DateTime>();
List<string> srvDoneDktr = new List<string>();
List<DateTime> srvDoneDateReferralDone = new
List<DateTime>();
List<string> srvDoneResult = new List<string>();
int ptnrtId;
string mova;

string rcrdCncl1, rcrdCncl2, rcrdCncl3, rcrdCncl4;

public PtnrtCard(int id, string lang)
{
    InitializeComponent();
    ptnrtId = id;
    mova = lang;
    if (mova == "UA")
    {
        this.Text = "Медкарта";
        tbPgInfo.Text = "Основна інформація";
        tbPgRecord.Text = "Медичні записи";
        tbPgObsteg.Text = "Обстеження";
        tbPgAppointment.Text = "Запис на прийом";
        lblPtnrtCard.Text = "Медкарта";
        lblName.Text = "Пацієнт";
        lblDateBirth.Text = "Дата народження";
        lblTextGender.Text = "Стать";
        groupBoxAddres.Text = "Адреса проживання";
        lbl1Street.Text = "Вулиця";
        lbl1House.Text = "Будинок";
        lbl1Apt.Text = "Квартира";
        lblDate.Text = "Дата";
        lblDktr.Text = "ПІБ лікаря";
        lblSrvDone.Text = "Обстеження:";
        lblObsteg.Text = "Кількість обстежень на черзі:";
        tbPgActiveObsteg.Text = "Активні обстеження";
        lbl1ActiveObstegName.Text = "Назва обстеження";
    }
}

```

```

lblDoneObstegDate.Text = "Дата призначення";
lblDoneObstegDktr.Text = "Лікар";
groupBoxResultObsteg.Text = "Результат";
lblHistoryAppointment.Text = "Історія записів";

dtGrdVwAppntmntActv.Columns[0].HeaderText =
"Дата";
dtGrdVwAppntmntActv.Columns[1].HeaderText =
"Час";
dtGrdVwAppntmntActv.Columns[2].HeaderText =
"Фахівець";
dtGrdVwAppntmntHstr.Columns[0].HeaderText = "Дата
запису";
dtGrdVwAppntmntHstr.Columns[1].HeaderText = "Час
запису";
dtGrdVwAppntmntHstr.Columns[2].HeaderText =
"Фахівець";

rcrdCncl1 = "Запис ";
rcrdCncl2 = " на ";
rcrdCncl3 = " до лікаря ";
rcrdCncl4 = " видалено";

}
else if (mova == "UK")
{
    this.Text = "Medical card";
    tbPgInfo.Text = "Main information";
    tbPgRecord.Text = "Medical rcrds";
    tbPgObsteg.Text = "Obstegs";
    tbPgAppointment.Text = "Appointments";
    lblPtnrtCard.Text = "Medical card";
    lblName.Text = "Ptnrt";
    lblDateBirth.Text = "Date of birth";
    lblTextGender.Text = "Gender";
    groupBoxAddres.Text = "Addres";
    lbl1Street.Text = "Street";
    lbl1House.Text = "House";
    lbl1Apt.Text = "Apt";
    lblDate.Text = "Date";
    lblDktr.Text = "Name dktr";
    lblSrvDone.Text = "Count ending obstegs";
    lblObsteg.Text = "Count active obstegs";
    tbPgActiveObsteg.Text = "Active obstegs";
    tbPgDoneObsteg.Text = "Done obstegs";
    lbl1ActiveObstegDate.Text = "Date of obsteg";
    lbl1ActiveObstegName.Text = "Name of obsteg";
    lbl1ActiveObstegDktr.Text = "Name dktr";
    lblDoneObstegDate.Text = "Date of obsteg";

```

```

        lblDoneObstegName.Text = "Name of obsteg";
        lblDoneObstegDktr.Text = "Name of dktr";
        lblDoneObstegFinishDate.Text = "Date finish
obsteg";

        groupBoxResultObsteg.Text = "Result";
        lblAcvtiveAppointment.Text = "Active
appointment";

        lblHistoryAppointment.Text = "History
appointment";

        dtGrdVwAppntmntActv.Columns[0].HeaderText = "Date
of appointment";
        dtGrdVwAppntmntActv.Columns[1].HeaderText = "Time
of appointment";
        dtGrdVwAppntmntActv.Columns[2].HeaderText =
"Spclt";
        dtGrdVwAppntmntHstr.Columns[0].HeaderText = "Date
of appointment";
        dtGrdVwAppntmntHstr.Columns[1].HeaderText = "Time
of appointment";
        dtGrdVwAppntmntHstr.Columns[2].HeaderText =
"Spclt";

        rcrdCncl1 = "Record ";
        rcrdCncl2 = " on ";
        rcrdCncl3 = " to dktr ";
        rcrdCncl4 = " canceled";
    }

    try
    {
        ChKnction ChKnct = new ChKnction();
        ChKnct.ConnectionString = ChKnctionInfo.Server
            + ChKnctionInfo.Port +
ChKnctionInfo.Namespace
            + ChKnctionInfo.Pswrd +
ChKnctionInfo.KrstvchID;
        ChKnct.Open();

        int recordId = 1;
        CLikarnya.Record record =
CLikarnya.Record.OpenId(ChKnct, recordId.ToString());
        while (CLikarnya.Record.ExistsId(ChKnct,
recordId.ToString()) == true)
        {
            if (record.PtnrtId == id.ToString())
            {
                rcrdsNote.Add(record.Note);
            }
        }
    }
}

```

```

rcrdsDate.Add(record.Date.Value.ToShortDateString());
        CLikarnya.Dktr dktr =
CLikarnya.Dktr.OpenId(ChKnct, record.DktrId);
        rcrdsDktr.Add(dktr.Prizvch + " " +
dktr.Name + " " + dktr.Patronymic);
        dktr.Close();
    }
    recordId++;
    record = CLikarnya.Record.OpenId(ChKnct,
recordId.ToString());
    }
    if (rcrdsNote.Count != 0)
    {
        rcrdNmbr = 1;
        textBoxRecordCurrent.Text =
rcrdNmbr.ToString();
        if (rcrdsNote.Count > 1)
            buttonNextRecord.Enabled = true;
        lblRecord.Text = rcrdsNote[0];
        lblRecordDate.Text = rcrdsDate[0].ToString();
        lblDktrRecord.Text = rcrdsDktr[0].ToString();
    }

    CLikarnya.Ptnrt ptnrt =
CLikarnya.Ptnrt.OpenId(ChKnct, id.ToString());
    lblSNP.Text = ptnrt.Prizvch + " " + ptnrt.Name +
" " + ptnrt.Patronymic;
    lblBornDate.Text =
ptnrt.DateOfBirth.Value.ToShortDateString();
    lblPhoneNumber.Text = ptnrt.PhoneNumber;
    lblGender.Text = ptnrt.Gender;
    lblStreet.Text = ptnrt.Street;
    lblHouse.Text = ptnrt.House;
    lblApartment.Text = ptnrt.Apt;

    int obstegId = 1;
    CLikarnya.Obsteg obsteg =
CLikarnya.Obsteg.OpenId(ChKnct, obstegId.ToString());
    while (CLikarnya.Obsteg.ExistsId(ChKnct,
obstegId.ToString()) == true)
    {
        if (obsteg.PtnrtId == id.ToString())
        {
            if (obsteg.Result == null)
            {
                srvNew++;
                srvNewName.Add(obsteg.Name);
            }
        }
    }

```

```

srvNewDateReferral.Add(obsteg.DateReferral);
        CLikarnya.Dktr dktr =
CLikarnya.Dktr.OpenId(ChKnct, obsteg.DktrId);
        srvNewDktr.Add(dktr.Prizvch + " " +
dktr.Name + " " + dktr.Patronymic);
        dktr.Close();
    }
    else if (obsteg.Result != null)
    {
        srvDone++;
        srvDoneName.Add(obsteg.Name);

srvDoneDateReferral.Add(obsteg.DateReferral);
        CLikarnya.Dktr dktr =
CLikarnya.Dktr.OpenId(ChKnct, obsteg.DktrId);
        srvDoneDktr.Add(dktr.Prizvch + " " +
dktr.Name + " " + dktr.Patronymic);
        dktr.Close();

srvDoneDateReferralDone.Add(obsteg.DateReferralDone);
        srvDoneResult.Add(obsteg.Result);
    }
    }
    obstegId++;
    obsteg = CLikarnya.Obsteg.OpenId(ChKnct,
obstegId.ToString());
    }
    lblObsteg.Text += " " + srvNew.ToString();
    lblSrvDone.Text += " " + srvDone.ToString();
    textBoxObstegCountNew.Text = srvNew.ToString();
    textBoxObstegCountDone.Text = srvDone.ToString();
    textBoxRecordCount.Text =
rcrdsNote.Count.ToString();

// заполнение первой записи запланированных
обследований
if (srvNew != 0)
{
    buttonObstegHome.Enabled = true;
    buttonObstegEnd.Enabled = true;

    //srvNmbr = 1;
    textBoxObstegCurrentNew.Text = "1";
    if (srvNew > 1)
        buttonObstegNext.Enabled = true;

    lblDateSrvNew.Text = srvNewName[0];

```

```

        lblNameSrvNew.Text =
srvNewDateReferral[0].ToLongDateString();
        lblDktrSrvNew.Text = srvNewDktr[0];
    }

    if (srvDone != 0)
    {
        buttonObstegHomeDone.Enabled = true;
        buttonObstegEndDone.Enabled = true;

        textBoxObstegCurrentDone.Text = "1";
        if (srvDone > 1)
            buttonObstegNextDone.Enabled = true;

        lblDateSrvDone.Text =
srvDoneDateReferral[0].ToLongDateString();
        lblNameSrvDone.Text = srvDoneName[0];
        lblDktrSrvDone.Text = srvDoneDktr[0];
        lblDateDoneSrvDone.Text =
srvDoneDateReferralDone[0].ToLongDateString();
        richTextBoxResultSrvDone.Text =
srvDoneResult[0];
    }

    dtGrdVwAppntmntActv.RowHeadersVisible = false;
    dtGrdVwAppntmntHstr.RowHeadersVisible = false;
    int appointmentID = 1;
    int hAct = 0, hHist = 0;
    CLikarnya.Appointment appointment;
    CLikarnya.Spclt spec;
    while (CLikarnya.Appointment.ExistsId(ChKnct,
appointmentID.ToString()) == true)
    {
        appointment =
CLikarnya.Appointment.OpenId(ChKnct, appointmentID.ToString());
        if (appointment.PtnrtId == ptnrtId &&
appointment.DateAppointment > DateTime.Now)
        {
            dtGrdVwAppntmntActv.Rows.Add();
            dtGrdVwAppntmntActv[0, hAct].Value =
appointment.DateAppointment.Value.ToShortDateString();
            dtGrdVwAppntmntActv[1, hAct].Value =
appointment.DateAppointment.Value.ToShortTimeString();
            spec = CLikarnya.Spclt.OpenId(ChKnct,
(appointment.SpcltId + 1).ToString());
            if (movs == "UA")
            {
                dtGrdVwAppntmntActv[2, hAct].Value =
spec.SpcltNameUA;

```

```

        }
        else if (mova == "RU")
        {
            dtGrdVwAppntmntActv[2, hAct].Value =
spec.SpcltNameRU;
        }
        else if (mova == "UK")
        {
            dtGrdVwAppntmntActv[2, hAct].Value =
spec.SpcltNameUK;
        }
        dtGrdVwAppntmntActv[3, hAct].Value =
appointmentID;
        hAct++;
    }
    if (appointment.PtnrtId == ptnrtId &&
appointment.DateAppointment <= DateTime.Now)
    {
        dtGrdVwAppntmntHstr.Rows.Add();
        dtGrdVwAppntmntHstr[0, hHist].Value =
appointment.DateAppointment.Value.ToShortDateString();
        dtGrdVwAppntmntHstr[1, hHist].Value =
appointment.DateAppointment.Value.ToShortTimeString();
        spec = CLikarnya.Spclt.OpenId(ChKnct,
(appointment.SpcltId + 1).ToString());
        dtGrdVwAppntmntHstr[2, hHist].Value =
spec.SpcltNameUA;
        hHist++;
    }
    appointmentID++;
}

ChKnct.Close();
}
catch (Exception eKn)
{
    MessageBox.Show("Помилка StvorZvyazk :" +
eKn.Message);
}

private void buttonNextRecord_Click(object sender,
EventArgs e)
{
    buttonPerviousRecord.Enabled = true;
    rcrdNmbr++;
    if(rcrdNmbr == rcrdsNote.Count)
        buttonNextRecord.Enabled = false;
    textBoxRecordCurrent.Text = rcrdNmbr.ToString();
}

```

```

        lblRecord.Text = rcrdsNote[rcrdNmbr - 1];
        lblRecordDate.Text = rcrdsDate[rcrdNmbr -
1].ToString();
        lblDktrRecord.Text = rcrdsDktr[rcrdNmbr -
1].ToString();
    }

    private void buttonPerviousRecord_Click(object sender,
EventArgs e)
    {
        buttonNextRecord.Enabled = true;
        rcrdNmbr--;
        if (rcrdNmbr == 1)
            buttonPerviousRecord.Enabled = false;
        textBoxRecordCurrent.Text = rcrdNmbr.ToString();

        lblRecord.Text = rcrdsNote[rcrdNmbr - 1];
        lblRecordDate.Text = rcrdsDate[rcrdNmbr -
1].ToString();
        lblDktrRecord.Text = rcrdsDktr[rcrdNmbr -
1].ToString();
    }

    private void buttonObstegNext_Click(object sender,
EventArgs e)
    {
        buttonObstegPervious.Enabled = true;
        srvNewCurrent++;
        if (srvNewCurrent == srvNewName.Count)
            buttonObstegNext.Enabled = false;
        textBoxObstegCurrentNew.Text =
srvNewCurrent.ToString();

        lblDateSrvNew.Text = srvNewDateReferral[srvNewCurrent
- 1].ToShortDateString();
        lblNameSrvNew.Text = srvNewName[srvNewCurrent - 1];
        lblDktrSrvNew.Text = srvNewDktr[srvNewCurrent - 1];
    }

    private void buttonObstegPervious_Click(object sender,
EventArgs e)
    {
        buttonObstegNext.Enabled = true;
        srvNewCurrent--;
        if (srvNewCurrent == 1)
            buttonObstegPervious.Enabled = false;
        textBoxObstegCurrentNew.Text =
srvNewCurrent.ToString();

```

```

        lblDateSrvNew.Text = srvNewDateReferral[srvNewCurrent
- 1].ToShortDateString();
        lblNameSrvNew.Text = srvNewName[srvNewCurrent - 1];
        lblDktrSrvNew.Text = srvNewDktr[srvNewCurrent - 1];
    }

    private void buttonGetAppiontment_Click(object sender,
EventArgs e)
    {
        AppointmentDktr rec = new AppointmentDktr(ptnrtId,
        mova);
        rec.Show();
    }

    private void pctrBxGetAppointment_Click(object sender,
EventArgs e)
    {
        AppointmentDktr rec = new AppointmentDktr(ptnrtId,
        mova);
        rec.Show();
    }

    private void pctrBxCancelAppointment_Click(object sender,
EventArgs e)
    {
        try
        {
            ChKnction ChKnct = new ChKnction();
            ChKnct.ConnectionString = ChKnctionInfo.Server
                + ChKnctionInfo.Port +
            ChKnctionInfo.Namespace
                + ChKnctionInfo.Pswrd +
            ChKnctionInfo.KrstvchID;
            ChKnct.Open();

            CLikarnya.Appointment appointmentDelete =
            CLikarnya.Appointment.OpenId(ChKnct, dtGrdVwAppntmntActv[3,
            dtGrdVwAppntmntActv.CurrentCell.RowIndex].Value.ToString());
            appointmentDelete.SpcltId = -1;
            appointmentDelete.PtnrtId = -1;
            appointmentDelete.DateAppointment = new
            DateTime(1999, 01, 01, 00, 00, 00);
            appointmentDelete.Save();
            appointmentDelete.Close();

            ChKnct.Close();

```

```

        MessageBox.Show(rcrdCncl1 +
dtGrdVwAppntmntActv[0,
dtGrdVwAppntmntActv.CurrentRow.Index].Value.ToString() +
rcrdCncl2 + dtGrdVwAppntmntActv[1,
dtGrdVwAppntmntActv.CurrentRow.Index].Value.ToString() +
rcrdCncl3 + dtGrdVwAppntmntActv[2,
dtGrdVwAppntmntActv.CurrentRow.Index].Value.ToString() +
rcrdCncl4);
    }
    catch (Exception eKn)
    {
        MessageBox.Show("Помилка StvorZvyazk :" +
eKn.Message);
    }
}

private void buttonObstegHome_Click(object sender,
EventArgs e)
{
    buttonObstegNext.Enabled = true;
    srvNewCurrent = 1;
    textBoxObstegCurrentNew.Text = 1.ToString();
    buttonObstegPervious.Enabled = false;
}

private void buttonObstegEnd_Click(object sender,
EventArgs e)
{
    buttonObstegPervious.Enabled = true;
    srvNewCurrent = srvNewName.Count;
    textBoxObstegCurrentNew.Text =
srvNewName.Count.ToString();
    buttonObstegNext.Enabled = false;
}

private void buttonObstegNextDone_Click(object sender,
EventArgs e)
{
    buttonObstegPerviousDone.Enabled = true;
    srvDoneCurrent++;
    if (srvDoneCurrent == srvDoneName.Count)
        buttonObstegNextDone.Enabled = false;
    textBoxObstegCurrentDone.Text =
srvDoneCurrent.ToString();

    lblDateSrvDone.Text =
srvDoneDateReferral[srvDoneCurrent - 1].ToShortDateString();
    lblNameSrvDone.Text = srvDoneName[srvDoneCurrent -
1];

```

```

        lblDktrSrvDone.Text = srvDoneDktr[srvDoneCurrent -
1];
        lblDateDoneSrvDone.Text =
srvDoneDateReferralDone[srvDoneCurrent - 1].ToShortDateString();
        richTextBoxResultSrvDone.Text =
srvDoneResult[srvDoneCurrent - 1];
    }

    private void buttonObstegPerviousDone_Click(object
sender, EventArgs e)
    {
        buttonObstegNextDone.Enabled = true;
        srvDoneCurrent--;
        if (srvDoneCurrent == 1)
            buttonObstegPerviousDone.Enabled = false;
        textBoxObstegCurrentDone.Text =
srvDoneCurrent.ToString();

        lblDateSrvDone.Text =
srvDoneDateReferral[srvDoneCurrent - 1].ToShortDateString();
        lblNameSrvDone.Text = srvDoneName[srvDoneCurrent -
1];
        lblDktrSrvDone.Text = srvDoneDktr[srvDoneCurrent -
1];
        lblDateDoneSrvDone.Text =
srvDoneDateReferralDone[srvDoneCurrent - 1].ToShortDateString();
        richTextBoxResultSrvDone.Text =
srvDoneResult[srvDoneCurrent - 1];
    }

    private void buttonObstegHomeDone_Click(object sender,
EventArgs e)
    {
        buttonObstegNextDone.Enabled = true;
        srvDoneCurrent = 1;
        buttonObstegPerviousDone.Enabled = false;
        textBoxObstegCurrentDone.Text =
srvDoneCurrent.ToString();

        lblDateSrvDone.Text =
srvDoneDateReferral[srvDoneCurrent - 1].ToShortDateString();
        lblNameSrvDone.Text = srvDoneName[srvDoneCurrent -
1];
        lblDktrSrvDone.Text = srvDoneDktr[srvDoneCurrent -
1];
        lblDateDoneSrvDone.Text =
srvDoneDateReferralDone[srvDoneCurrent - 1].ToShortDateString();
        richTextBoxResultSrvDone.Text =
srvDoneResult[srvDoneCurrent - 1];
    }

```

```

    }

    private void buttonObstegEndDone_Click(object sender,
EventArgs e)
    {
        buttonObstegPerviousDone.Enabled = true;
        srvDoneCurrent = srvDoneName.Count;
        buttonObstegNextDone.Enabled = false;
        textBoxObstegCurrentDone.Text =
srvDoneCurrent.ToString();

        lblDateSrvDone.Text =
srvDoneDateReferral[srvDoneCurrent - 1].ToShortDateString();
        lblNameSrvDone.Text = srvDoneName[srvDoneCurrent -
1];
        lblDktrSrvDone.Text = srvDoneDktr[srvDoneCurrent -
1];
        lblDateDoneSrvDone.Text =
srvDoneDateReferralDone[srvDoneCurrent - 1].ToShortDateString();
        richTextBoxResultSrvDone.Text =
srvDoneResult[srvDoneCurrent - 1];
    }

    private void buttonRecordHome_Click(object sender,
EventArgs e)
    {
        buttonNextRecord.Enabled = true;
        rcrdNmbr = 1;
        buttonPerviousRecord.Enabled = false;
        textBoxRecordCurrent.Text = rcrdNmbr.ToString();

        lblRecord.Text = rcrdsNote[rcrdNmbr - 1];
        lblRecordDate.Text = rcrdsDate[rcrdNmbr -
1].ToString();
        lblDktrRecord.Text = rcrdsDktr[rcrdNmbr -
1].ToString();
    }

    private void buttonRecordEnd_Click(object sender,
EventArgs e)
    {
        buttonPerviousRecord.Enabled = true;
        rcrdNmbr = rcrdsNote.Count;
        buttonNextRecord.Enabled = false;
        textBoxRecordCurrent.Text = rcrdNmbr.ToString();

        lblRecord.Text = rcrdsNote[rcrdNmbr - 1];
        lblRecordDate.Text = rcrdsDate[rcrdNmbr -
1].ToString();
    }

```

```

        lblDktrRecord.Text = rcrdsDktr[rcrdNmbr -
1].ToString();
    }
}

private void checkBoxFrd_CheckedChanged(object sender,
EventArgs e)
{
    if (checkBoxFrd.Checked == true)
    {
        groupBoxFrd.Enabled = true;
    }
    else
    {
        comboBoxFrdStart.ResetText();
        comboBoxFrdEnd.ResetText();
        groupBoxFrd.Enabled = false;
    }
}

private void comboBoxSpclt_SelectedIndexChanged(object
sender, EventArgs e)
{
    if (comboBoxSpclt.SelectedIndex != -1)
    {
        buttonSetRozkld.Enabled = true;
    }
    else
    {
        buttonSetRozkld.Enabled = false;
    }
}
}

```

Додаток В
Слайди презентації

**Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

Програмне забезпечення підтримки роботи лікарні

Виконав: Сергій СКОРІН

ст. групи КНТ-112сп

Керівник: Матвій ІЛЬЯШЕНКО

к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- * Об'єкт дослідження – процес розробки програмного забезпечення для підтримки функціонування медичних закладів.
- * Предмет дослідження – програмні засоби для підтримки функціонування медичних закладів.
- * Метою роботи є розробка програмного забезпечення для підтримки роботи лікарні.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для підтримки роботи лікарні;
- здійснити проектування програмного забезпечення для підтримки роботи лікарні;
- створити програмне забезпечення для підтримки роботи лікарні;
- виконати тестування розробленого програмного забезпечення для підтримки роботи лікарні.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Caspio	TimeTap	SoftClinic
Управління записами пацієнтів	–	+	+
Синхронізація з календарями	+	+	–
Нагадування для пацієнтів	–	+	+
Аналітика та звітність	+	+–	+
Гнучкість налаштувань	+	+–	+
Модулі для лабораторії та аптеки	–	–	+

4

Рисунок В.4 – Слайд 4

Постановка завдання

При розробці програмного забезпечення для підтримки роботи лікарні необхідно забезпечити такі функціональні вимоги:

- підтримка можливості авторизації та реєстрації користувачів ПЗ лікарні;
- підтримка можливості отримання пацієнтом інформації про медичні обстеження;
- можливість вибору мови інтерфейсу користувача;
- можливість додавання нових лікарів до системи;
- підтримка можливості складання розкладу роботи лікарів;
- підтримка можливості запису на прийом до лікарів на вільний час згідно з їх розкладом роботи.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Велика спільнота розробників	+	+	+–
Зручність синтаксису	+	–	+–
Підтримка об'єктно-орієнтоване програмування	+	+–	+
Зручність для створення ПЗ для підтримки роботи лікарні	+	–	–
Безпека та надійність	+	+–	+–

6

Рисунок В.6 – Слайд 6

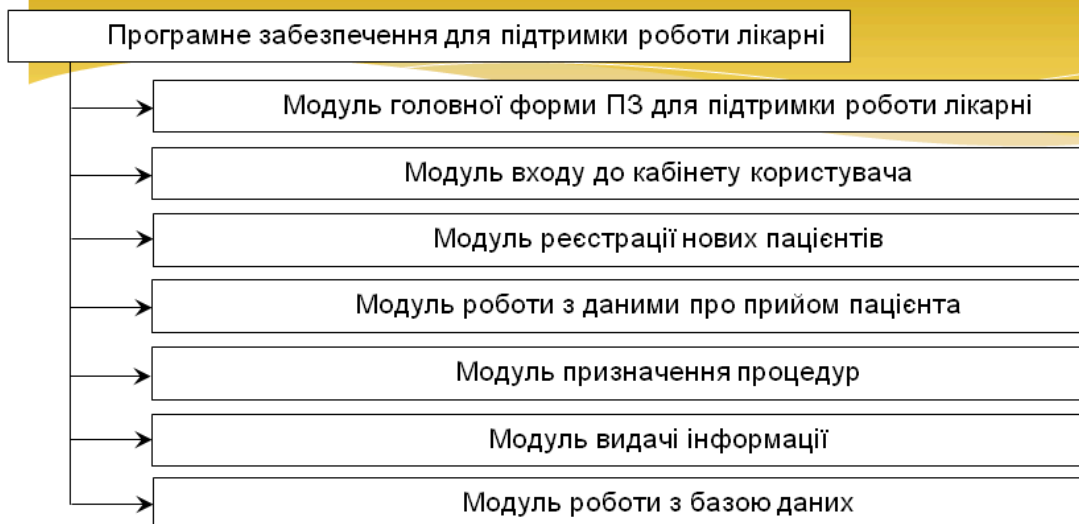
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	NetBeans	Eclipse
Інтеграція з .NET	+	–	–
Розвинені інструменти для налагодження	+	+–	+
Засоби автоматичного тестування	+	+	+–
Продуктивність та швидкість роботи	+	+–	+–
Глибока інтеграція з Windows-середовищем	+	–	–

7

Рисунок В.7 – Слайд 7

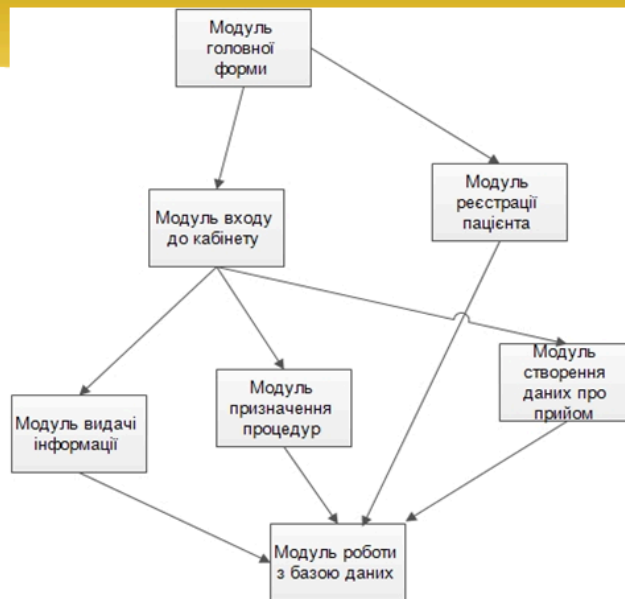
Структура програми



8

Рисунок В.8 – Слайд 8

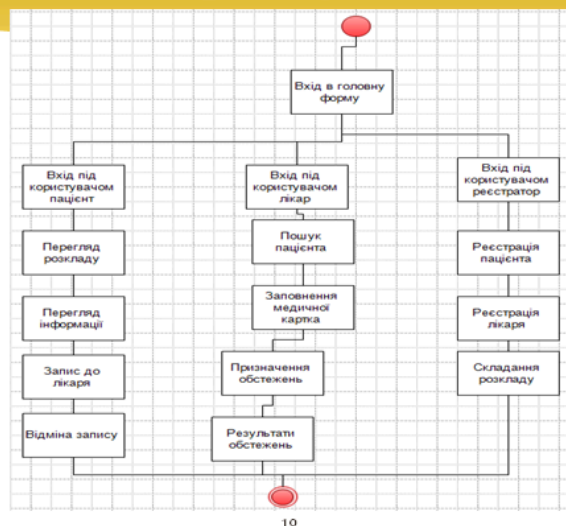
Схема функціонування програми



9

Рисунок В.9 – Слайд 9

Процес функціонування для різних типів користувачів



10

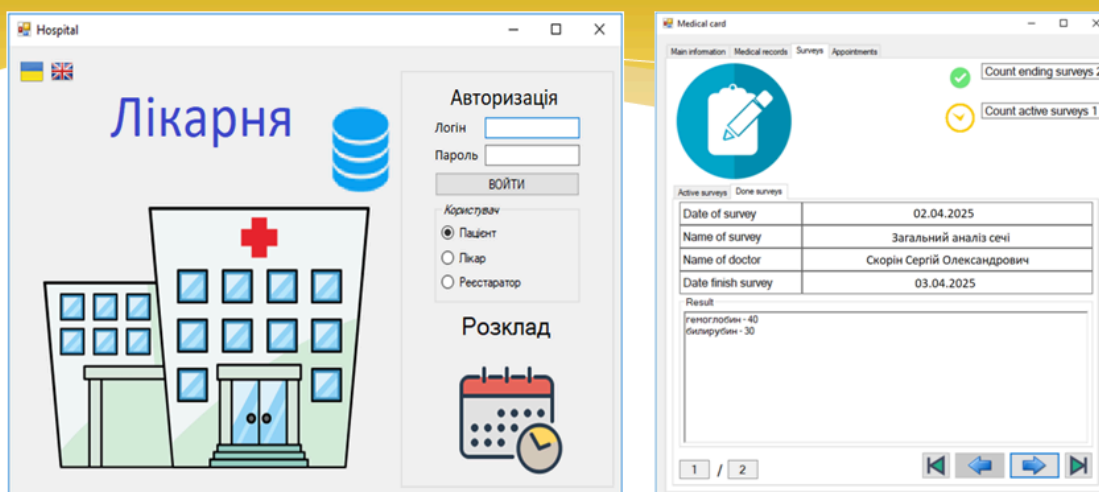
Рисунок В.10 – Слайд 10



11

Рисунок В.11 – Слайд 11

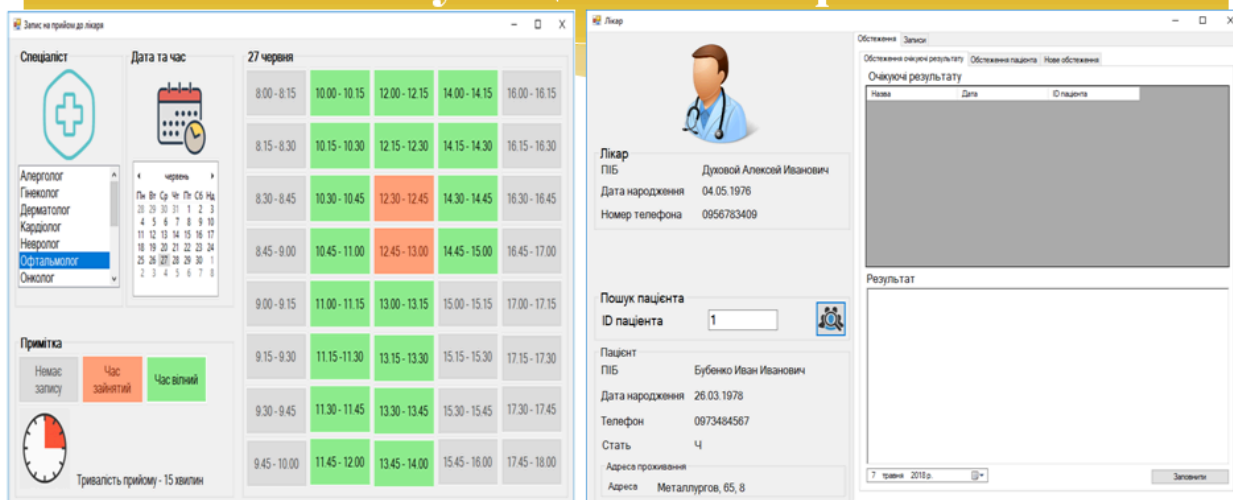
Робота пацієнта з програмою



12

Рисунок В.12 – Слайд 12

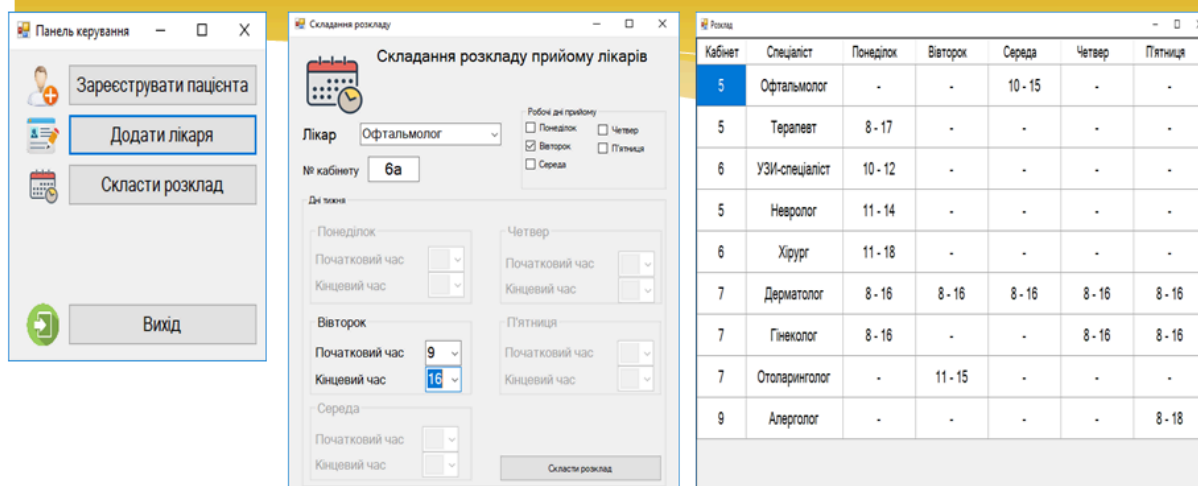
Робота з програмою. Запис на прийом. Функціонал лікаря



13

Рисунок В.13 – Слайд 13

Робота адміністратора з програмою



14

Рисунок В.14 – Слайд 14

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для підтримки функціонування медичних закладів.

Сформульовано функціональні вимоги до програмного забезпечення для підтримки роботи лікарні. Для реалізації програмного забезпечення для підтримки роботи лікарні обрано мову програмування C# та середовище розробки Visual Studio.

Запропоновано структуру програмного забезпечення для підтримки роботи лікарні. Визначено, що програмне забезпечення для підтримки роботи лікарні складається з кількох взаємопов'язаних модулів (модуль головної форми, модуль входу до кабінету користувача, модуль реєстрації нових пацієнтів, модуль роботи з даними про прийом пацієнта, модуль призначення процедур, модуль видачі інформації, модуль роботи з базою даних), кожен із яких виконує певні функції для забезпечення ефективної роботи медичного закладу.

В результаті виконання завдання було отримано програмну систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена програмна система є зручною для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.