Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“ Дослідження і програмна реалізація модуля е-реєстратури лікувально-діагностичного центру.”

КбПЗ_2024

Виконав здобувач вищої освіти
ІІ курсу, групи КН-23М
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ С.В. Беседа
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук
_____ О.П. Доренський
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12_____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерні науки"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 7 » серпня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Беседі Сергію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження і програмна реалізація модуля е-реєстратури лікувально-діагностичного центру*

2. Керівник роботи *Доренський Олександр Павлович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 7.08.2024 року

3. Строк подання студентом роботи до захисту *07.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація програмного модуля електронної реєстратури медичного закладу для ведення електронної картки пацієнта та створення попереднього запису на відвідування профільного медичного спеціаліста.*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування модуля.	*9. Висновки.*
5. Впровадження модуля в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема роботи додатку	*2 аркуша*
Маркетингове та економічне обгрунтування ІТ-проекту	*1 аркуш*

КОПІЯ-2024

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська Г.О.	21.11.2024	4.12.2024
Охорона праці	Марченко К.М.	21.11.2024	2.12.2024

7. Дата видачі завдання « 7 » серпня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	26.11.2024 р.	
9.	Оформлення ПЗ	27.11.2024 р.	
10.	Попередній захист роботи	07.12.2024 р.	

КПТВ_2024

Дата видачі завдання
« 7 » серпня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 7 » серпня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Беседа С.В. Дослідження і програмна реалізація модуля е-реєстратури лікувально-діагностичного центру. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

Випускна кваліфікаційна робота за другим (магістерським) рівнем вищої освіти присвячена дослідженню і програмній реалізації модуля е-реєстратури для лікувально-діагностичного центру. Модуль реєстратури є важливою складовою частиною сучасних медичних інформаційних систем і забезпечує автоматизацію процесів реєстрації пацієнтів, управління медичними картками та запису на прийом до лікарів.

Робота включає аналіз існуючих систем та підходів до автоматизації реєстратури в медичних закладах. Проведено огляд сучасних технологій та інструментів, які використовуються для створення подібних систем, зокрема JavaScript/React для розробки Frontend та Java з фреймворками Spring, JPA та Hibernate для Backend. Особлива увага приділяється використанню мікросервісної архітектури для забезпечення гнучкості та масштабованості системи.

У ході дослідження проведено аналіз потреб лікувально-діагностичного центру, де буде впроваджено даний модуль. На основі зібраних експериментальних даних визначено особливості роботи медичного закладу, такі як середня кількість пацієнтів на день, найбільш завантажені дні та години роботи, а також розподіл пацієнтів за віковими категоріями та статтю. Ці дані дозволили розробити оптимальну структуру модуля та визначити його основні функціональні можливості.

У роботі також детально розглянуто процеси проектування, розробки та тестування модуля е-реєстратури. Представлено прототип системи, який дозволяє перевірити основні функції та взаємодію між компонентами.

КОПЗ_2024

Проведено інтеграцію модуля з іншими системами медичного закладу, такими як лабораторний модуль та модуль стаціонарного лікування.

Результатом роботи є розроблений і протестований модуль е-реєстратури, який покращує ефективність роботи медичного закладу, забезпечує високу якість обслуговування пацієнтів та сприяє оптимізації робочих процесів. Впровадження цього модуля дозволяє значно зменшити навантаження на медичний персонал та підвищити рівень узгодженості даних.

КБПЗ_2024

ABSTRACT

Beseda S.V. Research and software implementation of the e-registry module of the medical and diagnostic center. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

The Graduation qualification work at the second (master's) level of higher education is dedicated to the research and software implementation of the e-registration module for a medical diagnostic center. The e-registration module is a crucial component of modern medical information systems and ensures the automation of patient registration processes, management of medical records, and appointment scheduling with doctors.

The work includes an analysis of existing systems and approaches to automating registration in medical institutions. It provides an overview of modern technologies and tools used to create such systems, including JavaScript/React for frontend development and Java with Spring, JPA, and Hibernate frameworks for the backend. Special attention is given to the use of microservice architecture to ensure the system's flexibility and scalability.

During the research, an analysis of the needs of the medical diagnostic center where the module will be implemented was conducted. Based on the collected experimental data, the specific characteristics of the medical institution's operation were identified, such as the average number of patients per day, the busiest days and working hours, and the distribution of patients by age and gender. These data helped to develop the optimal structure of the module and determine its main functional capabilities.

The work also details the processes of designing, developing, and testing the e-registration module. A prototype of the system is presented, allowing for verification of the main functions and interaction between components. Agile and Scrum methodologies were used for iterative implementation of functionality and

КДПЗ-2024

regular testing. Integration of the module with other medical institution systems, such as the laboratory module and inpatient treatment module, was carried out.

The result of the work is a developed and tested e-registration module that improves the efficiency of the medical institution, ensures high-quality patient care, and contributes to the optimization of work processes. The implementation of this module significantly reduces the workload on medical staff and data consistency.

КбПЗ_2024

ЗМІСТ

КОПЗ_2024

КбПЗ_2024

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення
JS	–	JavaScript
API	–	Application Programming Interface
SQL	–	Structured query language
REST	–	REpresentational State Transfer
JWT	–	JSON Web Token
ORM	–	Object-Relational Mapping
JSON	–	JavaScript Object Notation
CRUD	–	від англ. create, read, update, delete
JPA	–	Java Persistence API
ACID	–	від англ. atomicity, consistency, isolation, durability
HTTP(S)	–	Hypertext Transfer Protocol(Secure)
MUI	–	Material User interface
EMR	–	Electronic Medical Record
EHR	–	Electronic health record
RSA	–	криптографічний алгоритм Rivest, Shamir и Adleman
HIPPA	–	Health Insurance Portability and Accountability Act
HMAC	–	від англ. hash-based message authentication code
НСЗУ	–	Національна служба здоров'я України
РРО	–	Реєстратор розрахункових операцій

КбПЗ_2024

ВСТУП

У сучасному світі і в Україні медичні заклади стикаються зі зростаючим обсягом даних пацієнтів, які потребують ефективного управління та аналізу. Сучасні програмні технології відіграють ключову роль у забезпеченні належного функціонування середніх і великих медичних установ, де точність, швидкість і безпека обробки даних є критичними для надання високоякісних медичних послуг.

Одним із основних переваг використання сучасних програмних технологій є підвищення ефективності роботи медичного персоналу і його продуктивності. Автоматизація рутинних процесів, таких як реєстрація пацієнтів, облік аналізів та ведення медичних карток, яке дозволяє значно скоротити час, витрачений на адміністративні завдання. Це, своєю чергою, дає медичним працівникам можливість зосередитися на своїх основних обов'язках — лікуванні та консультуванні пацієнтів.

Крім того, сучасні програмні рішення забезпечують інтеграцію різних відділів та підрозділів лікарні. За допомогою мікросервісної архітектури та єдиної бази даних, медичні установи можуть створювати єдину інформаційну систему, де всі дані пацієнтів доступні у реальному часі. Це дозволяє лікарям отримувати повну картину про стан пацієнта, що сприяє прийняттю обґрунтованих рішень і покращує якість надання медичних послуг.

Також варто відзначити важливість безпеки даних у медичних закладах. Використання технологій аутентифікації та шифрування даних забезпечує захист конфіденційної інформації пацієнтів від несанкціонованого доступу. Це особливо актуально у світлі зростання кіберзагроз та необхідності дотримання міжнародних стандартів захисту даних.

Сучасні програмні рішення також сприяють покращенню взаємодії між пацієнтами та медичним закладом. Застосування електронних медичних карток, мобільних додатків та онлайн-платформ дозволяє пацієнтам зручно записуватися на прийом до лікаря, переглядати результати аналізів та додають можливість

КПІ – 2024

отримувати консультації дистанційно. Це не лише підвищує комфорт пацієнтів, але й зменшує навантаження на медичний персонал.

Загалом, впровадження сучасних програмних технологій у медичних центрах є необхідним кроком для підвищення ефективності, безпеки та якості надання медичних послуг. Це інвестиція у майбутнє, яка дозволяє медичним установам відповідати на виклики сучасного світу та задовольняти потреби пацієнтів на високому рівні обслуговування.

Актуальність теми дослідження обумовлена потребою сучасних медичних установ у вдосконаленні обліку, управління та обробки даних пацієнтів. Традиційні паперові та неавтоматизовані системи часто не здатні задовольнити вимоги швидкої, точної та надійної роботи з великим обсягом інформації. Це створює ризики помилок, збільшує час обслуговування пацієнтів та підвищує навантаження на персонал. З огляду на сучасні виклики, включаючи кіберзагрози, вимоги до захисту персональних даних та інтеграцію з державними електронними системами, застосування інноваційних програмних рішень є вкрай актуальним.

Проект спрямований на автоматизацію процесів реєстрації пацієнтів, покращення обміну інформацією між відділеннями та забезпечення конфіденційності даних, що сприяє підвищенню якості медичних послуг та ефективності роботи закладу.

Метою та завданням цього дослідження є створення та впровадження модуля "е-реєстратура", який забезпечує автоматизацію процесів реєстрації пацієнтів, управління записами та інтеграцію з іншими підсистемами медичного закладу. Основні завдання цієї праці такі:

1) аналіз потреб медичних закладів у цифровізації процесів реєстрації є вибір та обґрунтування програмних засобів і технологій для розробки модуля ;

2) розробка архітектури модуля з використанням сучасних підходів, таких як мікросервісна структура;

3) реалізація функціональних можливостей: реєстрація пацієнтів, пошук за унікальними ідентифікаторами, інтеграція з іншими медичними системами. -

КПЗ – 2024

забезпечення безпеки обміну даними через технології шифрування та аутентифікації;

4) оцінка ефективності впровадження через показники економічної вигоди та покращення якості обслуговування.

Наукова новизна результатів цієї праці полягає у:

1) розроблено модель компонент діджиталізованих бізнеспроцесів лікувально-діагностичного центру (медичного закладу) [51];

2) набув подальшого розвитку метод пошуку записів у базі даних пацієнтів з використанням біометрії, який на відміну від існуючих (традиційних) забезпечує підвищення швидкості цифрової ідентифікації пацієнтів лікувально-діагностичного центру, а також точності оброблення даних в інформаційній системі медичного закладу за рахунок зменшення ймовірності помилкового модифікування медичних даних.

Предметом дослідження є моделі та методи цифрової трансформації процесу реєстрації пацієнтів у лікувально-діагностичному центрі (в медичному закладі). Це охоплює підходи до проєктування, розробки, впровадження та оптимізації функціонального модуля "е-реєстратура", який забезпечує ефективний облік пацієнтів, інтеграцію з іншими медичними системами, захист персональних даних та відповідність сучасним вимогам до програмного забезпечення у медичній сфері.

Об’єктом дослідження є інформаційна система лікувально-діагностичного центру (медичного закладу), невід’ємною сладовою якої є модуль "е-реєстратура". Зокрема, вона включає процеси обробки даних пацієнтів, взаємодію медичного персоналу з програмним забезпеченням, функціональність реєстрації пацієнтів, інтеграцію між відділеннями та можливості застосування сучасних технологій (React, Spring Boot, PostgreSQL тощо) для створення ефективного та безпечного сервісу.

КБПЗ-2024

Методи дослідження, як такі мають бути використані:

- аналіз літературних та інтернет-джерел використовується для вивчення сучасних тенденцій у сфері медичних інформаційних систем, мікросервісної архітектури, інтерфейсів користувача та стандартів безпеки даних;

- метод моделювання передбачає побудову моделей процесів реєстрації пацієнтів, зокрема за допомогою діаграм процесів, структурних та функціональних схем, щоб виявити оптимальні рішення для автоматизації;

- емпіричні методи полягають у тестуванні прототипів модуля "е-реєстратура", оцінці його продуктивності, зручності для користувачів та відповідності вимогам медичних установ;

- метод експертних оцінок – це залучення медичного персоналу для оцінки практичності запропонованих рішень, зокрема функціоналу модуля, зручності інтерфейсу та відповідності потребам лікарів і пацієнтів;

- метод системного аналізу використовується для аналізу взаємодії модуля "е-реєстратура" з іншими компонентами медичної інформаційної системи, визначення критичних точок та їх оптимізації;

- метод прототипування передбачає створення функціонального прототипу модуля для тестування його працездатності, інтеграції з базою даних та перевірки взаємодії з іншими модулями системи;

- метод порівняльного аналізу полягає у порівнянні запропонованих рішень з існуючими аналогами на ринку, щоб визначити переваги та недоліки проєкту.

Застосування цих методів дозволяє отримати системне уявлення про процеси реєстрації пацієнтів, забезпечити якісну реалізацію програмного забезпечення та гарантувати його практичну користь для медичних закладів.

Практична цінність проєкту полягає у розробленні алгоритмів та програмного забезпечення модуля е-реєстратури лікувально-діагностичного центру, який може бути впроваджений у медичних установах різного масштабу для цифровізації реєстраційних процесів. Це дозволить значно скоротити час

КПЗ–2024

обслуговування пацієнтів, знизити навантаження на медичний персонал, уникнути помилок в обробці даних та підвищити загальну якість послуг.

Модуль "е-реєстратура" є універсальним інструментом для оптимізації роботи реєстратури, включаючи реєстрацію візитів, пошук інформації, управління чергами та планування ресурсів. Завдяки використанню відкритих стандартів і мікросервісної архітектури система може легко інтегруватися з іншими медичними підсистемами, такими як лабораторії, стаціонарні відділення чи система eHealth.

Результати впровадження включають:

- підвищення задоволеності пацієнтів за рахунок зменшення черг та забезпечення зручного доступу до послуг;

- скорочення витрат на паперову документацію;

- підвищення точності та швидкості обробки даних;

Основний результат цієї праці апробований на IX Міжнародній науково-практичній конференції «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» (м. Київ) [51].

Отже, ця кваліфікаційна робота сприяє підвищенню конкурентоспроможності медичного закладу на ринку та впровадженню сучасних стандартів роботи з даними медичної інформаційної системи.

КбПЗ_2024

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення модуля

Основною метою модуля «е-реєстратура» є забезпечення ефективного процесу управління реєстрацією пацієнтів у лікувально-діагностичному центрі. Система призначена для спрощення та автоматизації процесів запису пацієнтів на прийом до лікаря, зберігання інформації про пацієнтів, їхню медичну історію, облік прийомів та взаємодію з медичним персоналом. Оскільки сучасні медичні заклади обслуговують значну кількість пацієнтів щодня, необхідна платформа, яка б дозволила знизити навантаження на адміністративний персонал, підвищити точність обробки даних та зменшити ймовірність помилок під час реєстрації та запису хворих.

Модуль «е-реєстратура» призначений також для підтримки роботи як адміністративного, так і медичного персоналу. Медичні працівники, зокрема лікарі та медсестри, отримують доступ до електронних медичних записів пацієнтів, що полегшує діагностику, планування лікування та відстеження динаміки стану здоров’я хворих. Крім того, модуль дозволяє значно скоротити час на повторне введення даних пацієнта, оскільки інформація зберігається в електронному вигляді, доступна для всіх уповноважених осіб у реальному часі та може використовуватися з будь-якого комп’ютера, підключеного до системи.

Основні функції модуля «е-реєстратура»:

- реєстрація нових пацієнтів та ведення обліку наявних: Модуль дозволяє адміністраторам реєструвати нових пацієнтів, зберігаючи необхідну особисту інформацію, включаючи контактні дані, дату народження, медичні записи тощо. Всі дані хворих структуровано і надійно зберігаються в базі даних, що забезпечує швидкий доступ до інформації в разі необхідності;

КПЗ-2024

- запис на прийом: Модуль автоматизує процес запису пацієнтів на прийом до лікарів, дозволяючи хворим обирати зручний час, а також лікарів необхідної спеціалізації. Система інтегрується з розкладом лікарів, що допомагає уникнути дублювання записів та забезпечує ефективне управління часом;

- інформування пацієнтів: Модуль має функцію відправлення нагадувань пацієнтам про майбутні прийоми. Це може здійснюватися через швидкі повідомлення (Вайбер або Телеграм) або інші канали комунікації, що сприяє зменшенню кількості пропущених візитів та підвищує загальну ефективність роботи медичного закладу;

- зберігання медичних записів: Система дозволяє лікарям зберігати та переглядати електронні медичні записи пацієнтів, що включають інформацію про попередні обстеження, призначення, алергії, діагнози тощо. Завдяки цьому забезпечується швидкий доступ до історії хвороби, що, в свою чергу, підвищує якість медичних послуг;

- звітність та аналітика: Модуль «е-реєстратура» також генерує звіти та аналітику за визначеними критеріями, такими як кількість відвідувань, найпоширеніші діагнози, статистика записів тощо. Це надає керівництву клініки актуальну інформацію для прийняття рішень щодо оптимізації роботи медичного закладу;

- інтеграція з іншими медичними модулями: Система реєстратури може працювати в зв'язці з іншими модулями (наприклад, лабораторією, стаціонаром тощо), що дозволяє підтримувати комплексний підхід до лікування пацієнтів. Інформація зберігається централізовано і може передаватися між модулями без необхідності повторного введення;

Застосування системи «е-реєстратури» для управління процесами реєстрації має низку переваг у порівнянні з традиційними, паперовими методами:

- швидкість та ефективність: Завдяки автоматизації реєстратура може обробляти велику кількість пацієнтів за короткий час, що зменшує черги та підвищує комфорт для пацієнтів;

КПЗ-2024

- зниження людського фактору: Модуль мінімізує ризик помилок через ручне введення даних, що підвищує точність інформації;
- легкий доступ до даних: Вся інформація зберігається централізовано та доступна в будь-який момент, що полегшує роботу адміністративного та медичного персоналу;
- простота використання: Інтуїтивно зрозумілий інтерфейс для адміністраторів дозволяє швидко знаходити потрібну інформацію та виконувати необхідні операції, такі як реєстрація та запис на прийом;

Модуль «е-реєстратура» має важливу роль у підвищенні якості обслуговування хворих завдяки своєму функціоналу. Пацієнти отримують зручний доступ до медичних послуг і можуть ефективніше взаємодіяти з медичним закладом. Вони зможуть самостійно записуватися на прийом онлайн, отримувати інформацію про розклад лікарів і навіть нагадування про візити.

Завдяки електронній системі реєстратури пацієнтам не потрібно заповнювати повторно форму на кожному прийомі, що особливо важливо для хронічних хворих та людей з обмеженими можливостями. Це значно полегшує їх взаємодію з клінікою, знижує рівень стресу та зменшує витрати часу на непотрібні формальності.

Зберігання медичної інформації в електронному вигляді вимагає ретельного забезпечення безпеки даних пацієнтів. У модулі «е-реєстратура» реалізовані механізми захисту інформації, які відповідають вимогам законодавства і галузевих стандартів. Включення аутентифікації, шифрування даних, а також обмеження доступу до особистих даних тільки для авторизованих осіб допомагає запобігти несанкціонованому доступу до інформації.

Таким чином, призначення «е-реєстратура» — забезпечення ефективного управління процесами реєстрації пацієнтів і взаємодії між медичним та адміністративним персоналом, підвищення якості обслуговування, зниження навантаження на працівників реєстратури та медичних працівників, а також зменшення витрат часу пацієнтів на доступ до медичних послуг.

КПЗ - 2024

1.2 Область застосування

Система електронної реєстрації (е-реєстратура) є ключовою частиною інфраструктури сучасних медичних закладів, спрямованою на оптимізацію процесів обліку пацієнтів, адміністрування їхніх даних і полегшення доступу до медичних послуг. Область застосування е-реєстратури включає широкий спектр медичних установ та центрів, а також охоплює різні сценарії використання в межах кожного закладу. Впровадження такої системи в роботу медичних центрів забезпечує як ефективність, так і зручність для хворих і медичного персоналу, знижуючи потребу в паперовій документації та пришвидшуючи процеси обслуговування.

«Е-реєстратура» може бути використана в медичних установах різного профілю та рівня, включаючи:

- лікувально-діагностичні центри: Це одне з головних середовищ, для якого призначена система. Вони часто обслуговують сотні пацієнтів щодня, і система е-реєстратури дозволяє оперативно керувати записом хворих, вести облік їхніх даних і забезпечувати легкий доступ до необхідної інформації. Оскільки такі центри часто пропонують комплексні послуги (консультації фахівців, лабораторні дослідження, діагностика тощо), система е-реєстратури полегшує координацію між різними відділеннями і значно підвищує якість обслуговування;
- міські та районні поліклініки: У поліклініках модуль е-реєстратури дозволяє автоматизувати запис на прийом до лікарів і ведення електронних медичних карток пацієнтів. Це сприяє швидкому обслуговуванню хворих, адже лікарі отримують доступ до історії захворювань і результатів попередніх обстежень у кілька рухів; Враховуючи велику кількість відвідувачів поліклінік, е-реєстратура допомагає зменшити навантаження на відділ і сприяє оптимізації роботи персоналу;
- приватні клініки та медичні центри: Приватні медичні заклади, націлені на якісне обслуговування пацієнтів, також виграють від застосування е-

КОПІЯ 2024

реєстратури. У цих закладах особливо важлива точність і своєчасність надання послуг, що досягається через автоматизацію запису, збереження даних і управління інформацією про пацієнтів. Завдяки можливості інтеграції з іншими модулями, такими як лабораторія або стаціонар, система дозволяє приватним клінікам надавати комплексне медичне обслуговування;

- госпіталі і стаціонарні відділення: У госпіталях е-реєстратура допомагає зберігати інформацію про пацієнтів, записувати їх на консультації з профільними спеціалістами, призначати додаткові обстеження та керувати інформацією про перебування хворих у стаціонарі. Інтеграція з системою управління стаціонарним лікуванням дозволяє лікарям і медсестрам швидко отримувати доступ до історії хвороби, діагнозів та призначень, що сприяє підвищенню якості медичного обслуговування;

Система е-реєстратури використовується на різних етапах медичного обслуговування пацієнтів, від моменту їхнього першого звернення до клініки до моменту завершення лікування:

- реєстрація та первинний запис: Під час первинного звернення пацієнта до медичного закладу е-реєстратура дозволяє швидко зареєструвати його в системі, зберегти особисті дані, історію хвороби, контактну інформацію та інформацію про страхування. Це забезпечує точний облік пацієнтів і зберігання всієї необхідної інформації в єдиному місці;

- управління записами на прийом: е-реєстратура дозволяє записувати пацієнтів на прийом до лікарів різних спеціалізацій, а також призначати час відвідувань залежно від розкладу лікарів. Це полегшує процес взаємодії між пацієнтами та персоналом і допомагає уникнути черг;

- підтримка комунікації з пацієнтами: Система може бути використана для автоматичного надсилання пацієнтам нагадувань про майбутні прийоми, результатів обстежень чи запитів на повторне відвідування. Така комунікація забезпечує інформування пацієнтів, знижує ризик пропущених прийомів і сприяє кращій взаємодії з клінікою;

КПЗ-2024

- інтеграція з іншими системами: модуль «е-реєстратура» може бути інтегрована з лабораторними, аптеками та фінансовими системами, що дозволяє більш повноцінно управляти всією медичною інформацією пацієнта та забезпечувати доступ до неї в реальному часі. Завдяки цьому можна уникнути дублювання даних і підвищити загальну ефективність закладу;
- моніторинг та аналітика: «е-реєстратура» дозволяє створювати звіти за ключовими показниками роботи медичного закладу, зокрема кількістю відвідувань, статистикою записів до лікарів, найбільш популярними послугами тощо. Це забезпечує можливість керівництву клініки контролювати завантаженість відділень і планувати роботу персоналу на основі актуальних даних.

Область застосування модуля «е-реєстратура» включає широке коло завдань та має такі переваги:

- підвищення продуктивності: Завдяки автоматизації рутинних процесів, таких як запис пацієнтів, зберігання даних та надсилання нагадувань, «е-реєстратура» дозволяє персоналу клініки більше зосереджуватися на наданні медичних послуг, а не на адміністративній роботі;
- поліпшення якості обслуговування: Пацієнти отримують зручний доступ до інформації, зокрема можливість запису на прийом онлайн, що значно полегшує взаємодію з медичним закладом і підвищує рівень обслуговування;
- швидкий доступ до інформації: Усі необхідні дані про пацієнтів доступні в реальному часі, що дає лікарям змогу швидко отримати доступ до історії хвороби та приймати обґрунтовані рішення, засновані на повній і актуальній інформації;
- ефективне використання ресурсів: Модуль дозволяє оптимально використовувати наявні ресурси, у тому числі час лікарів та медсестер, знижуючи кількість пацієнтів, які не з’являються на прийом, завдяки нагадуванням та автоматизації запису.

Модуль «е-реєстратура» має також задовольняти нормативні вимоги, зокрема до захисту персональних даних, які встановлені місцевим законодавством і стандартами галузі. Забезпечення конфіденційності та безпеки даних пацієнтів є

КПЗ-2024

обов’язковою умовою застосування таких систем. Це включає шифрування, багаторівневий доступ і регулярне резервне копіювання даних.

Завдяки широкій області застосування, модуль «е-реєстратура» виступає критичним інструментом для сучасних медичних закладів. Її функціональні можливості дозволяють організувати роботу з даними пацієнтів у лікувальних центрах.

КбПЗ_2024

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень

Дуже часто, в медичних закладах України облік пацієнтів і запис хворих до лікаря здійснюється або здійснювались вручну з використанням паперових документів та спеціальних реєстраційних журналів. Ось як це зазвичай відбувається:

1. Реєстрація та облік пацієнтів:

- паперові картки: Кожен пацієнт мав власну паперову медичну картку. Вона містила всю історію захворювань, проведених процедур, результати аналізів і рекомендації лікарів. Ці картки зберігалися в реєстратурі або архіві лікарні. Медичний персонал вручну додавав нові записи про кожен візит;
- реєстраційні журнали: Під час візиту пацієнта медсестра або реєстратор вручну записували в журнал основні дані про пацієнта (ім'я, вік, адреса, дата візиту) і мету візиту;
- нумерація карток: У деяких лікарнях використовувалася система нумерації медичних карток. Кожна картка мала унікальний номер, який допомагав швидко знаходити потрібну інформацію в архіві.

2. Запис пацієнтів до лікаря:

- журнал запису до лікаря: Для управління розкладом лікарів використовували спеціальні паперові журнали або блокноти, де реєстратори вручну записували час прийому пацієнта. Ці записи містили інформацію про час і дату візиту, а також про ім'я пацієнта;
- телефонний запис: Пацієнти могли телефонувати в лікарню, щоб домовитися про прийом. Реєстратор вручну вносив їх до журналу;
- фіксовані години прийому: У багатьох клініках існувала система "живої черги", де лікарі приймали пацієнтів у порядку їх прибуття;

КОПЗ_2024

- запис до лікаря міг не бути необхідним, але пацієнт приходив у певні години прийому, які були зафіксовані.

3. Управління чергами та розкладом лікарів:

– таблиці та розклади на папері: Для управління розкладом лікарів використовували паперові таблиці або спеціальні стенди, де вказували час прийомів лікарів. Реєстратори або медперсонал оновлювали ці таблиці вручну;

– чергування лікарів: Лікарі часто мали графіки чергувань, які відображалися в спеціальних журналах або на інформаційних дошках. Це допомагало пацієнтам знати, коли лікар буде на місці.

4. Зберігання інформації:

– архіви: Паперові картки пацієнтів зберігалися в архівах лікарень, зазвичай у великих шафах або картотечних системах. Картки впорядковували за алфавітом або номерами для зручності пошуку. Іноді такі архіви займали значну частину простору в лікарні;

– проблеми зі зберіганням: Через великий обсяг паперових даних виникали проблеми з пошуком інформації, особливо якщо картка була втрачена або переміщена. Також виникали труднощі із зберіганням великих архівів, які вимагали постійної організації.

5. Виклики старої системи:

– повільність і неефективність: Обробка інформації вручну займала багато часу і вимагала зусиль від медичного персоналу. Часто могли виникати помилки через людський фактор;

– обмежена доступність даних: Якщо пацієнт відвідував кілька лікарень, його медична історія могла бути розкидана по різних установах, що ускладнювало лікарям отримання повної картини стану здоров'я;

– збереження: Паперові картки могли пошкодитися, загубитися або постраждати через пожежу або інші фізичні пошкодження;

КПЗ-2024

Паперові системи управління пацієнтами мали ряд обмежень і викликів. Вони були схильні до помилок, втрати даних, а також вимагали великих людських і матеріальних ресурсів для підтримки. Введення електронних систем медичних карток (EMR/EHR) стало революційним кроком у підвищенні ефективності та якості медичних послуг, дозволяючи значно швидше управляти інформацією, забезпечуючи її збереження та доступність.

Сучасні технології суттєво змінюють роботу в медичній сфері, дозволяючи лікарям і персоналу більше часу приділяти пацієнтам та приймати обґрунтовані рішення на основі швидкого доступу до даних. Автоматизація процесів, цифрові медичні картки та електронні системи дозволяють знизити кількість помилок і значно підвищити якість медичних послуг.

У медичних закладах використовуються різноманітні програмні системи для обліку пацієнтів, управління клінічними процесами та забезпечення ефективної роботи медичного персоналу.

Ось декілька найбільш поширених систем обліку пацієнтів у медичних закладах в Україні:

Health24 - Хмарна медична інформаційна система для управління процесами медичної установи будь-якого формату та форми власності: від мережевих клінік до приватних медичних кабінетів. Містить весь потрібний функціонал для роботи з eHealth.[22]

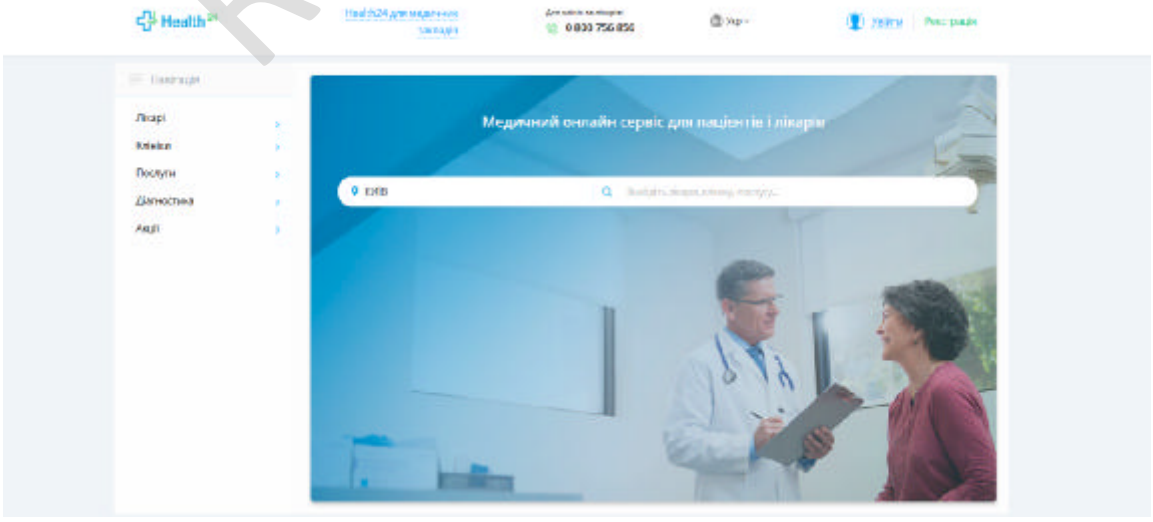


Рисунок 2.1- Інформаційний комплекс Health24 [22]

КбПЗ-2024

Імед - Програма для медичного центру, яка автоматизує усі бізнес-процеси: від реєстрації пацієнтів до фінансових звітів власнику. Керівництво медичної установи може швидко проаналізувати поточний стан справ завдяки постійному моніторингу показників діяльності. Інтегрується з іншими інформаційними системами (у тому числі лабораторій), сервісами doc.ua, likarni.com, IP телефонією, календарем Google, порталами Бітрікс24.[23]

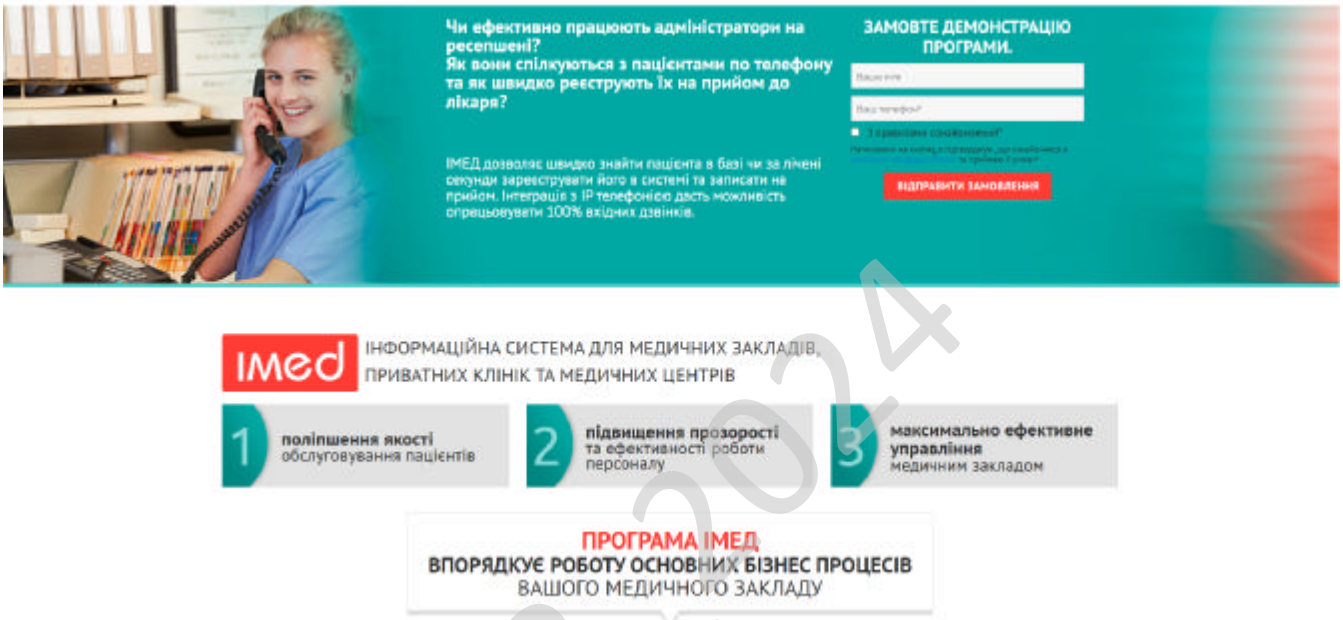


Рисунок 2.2- Медична система Імед[23]

Clinica Web - MIC підключено до системи eHealth. Інтеграція з Binotel, CheckBox, eHealth лабораторії та інші сервіси. Зручний інтерфейс запису на прийом. Нагадування пацієнтам про запис SMS/IM.[24]

КПІЗ 2024

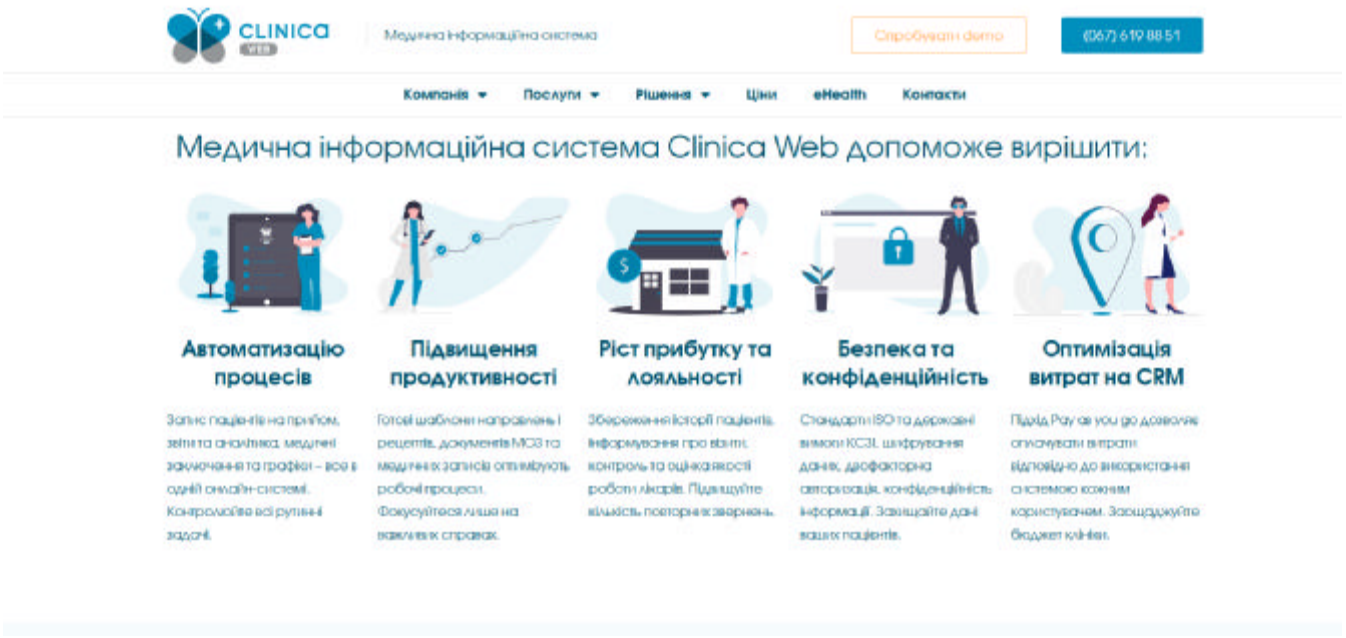


Рисунок 2.3- Медичний модуль Clinica Web[24]

Доктор Элекс - Комплексне рішення, що дозволяє оптимізувати процес лікування, змінюючи уявлення лікарів та пацієнтів про якість медичного обслуговування. Адаптивність до різних бізнес-процесів та умов використання. Забезпечить гармонійну співпрацю всього персоналу. Впровадить контроль за процесом лікування.[25]

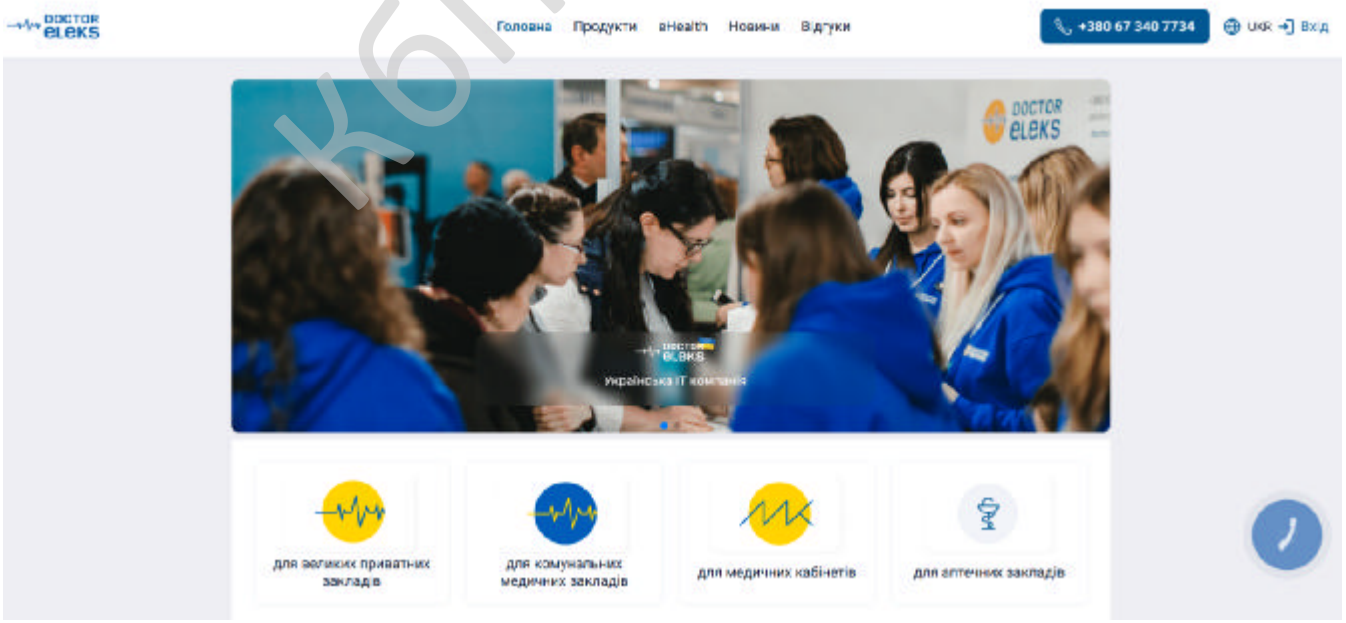


Рисунок 2.4- Медичний програмний комплекс Доктор Элекс[25]

КбПЗ 2024

EMCImed - Медична інформаційна система для автоматизації повного циклу надання медичних послуг: від запису пацієнта на прийом до подання звіту про надані послуги у НСЗУ. Модульна архітектура системи дозволяє налаштувати та адаптувати її під особливості та потреби конкретного медичного закладу.[26]

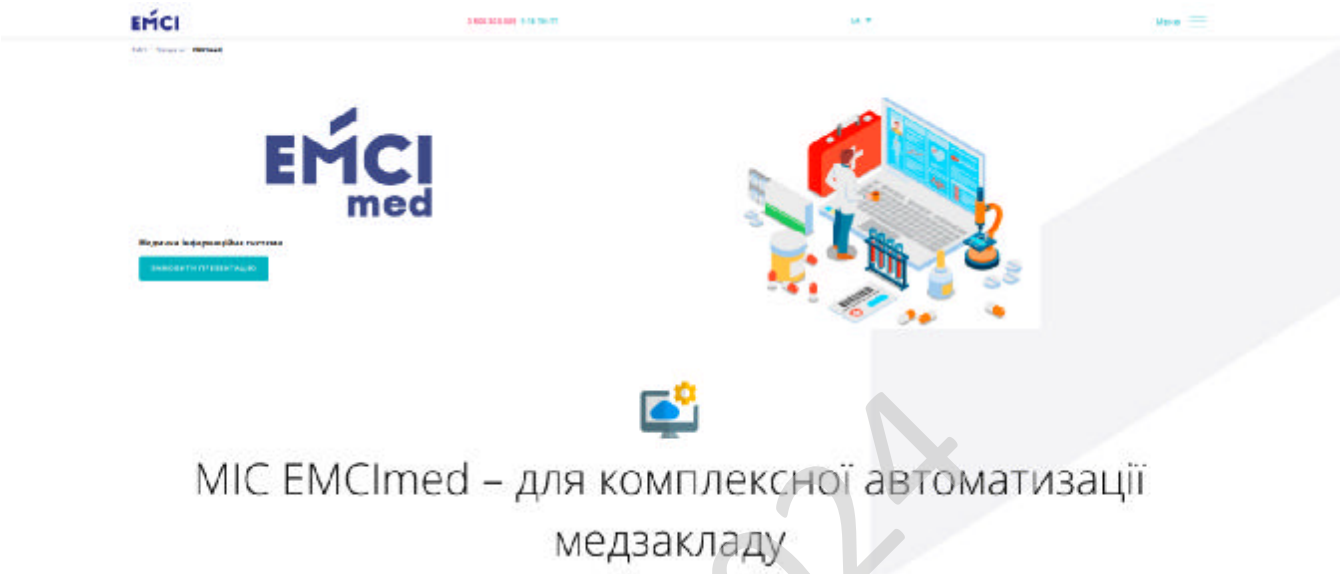


Рисунок 2.5- Лікувально-інформаційна програма EMCImed [26]

DocDream - Медична інформаційна система містить всі необхідні функції для роботи з пацієнтами, електронні медичні карти, контакт-центр, онлайн запис на прийом, документообіг, складський облік, управління персоналом, звітність та аналітика [27].

WebClinic - Універсальна програма для медичних центрів Є локальна та хмарна версія. Обмін даними з 1С: Бухгалтерія, Medoc. Інтеграція з eHealth. Дистанційний запис пацієнта самостійно. Підключення касового апарату та модуль програмного РРО [28].

Medexpert - Система управління медичною клінікою - призначена для малих та великих медичних клінік для автоматизації медичних та бізнес-процесів, аналізі ефективності роботи клініки, обліку та аналізу роботи клініки, а також впровадження сучасних інформаційних та цифрових технологій у практику специфічних медичних процесів [29].

КОПЗ-2024

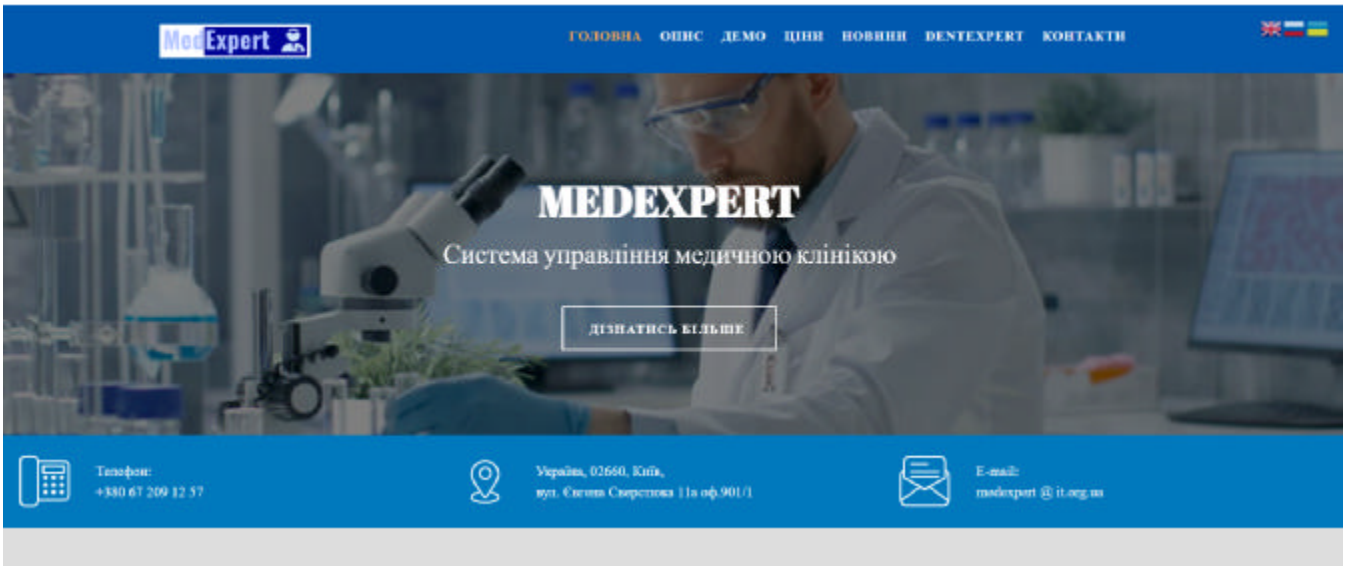


Рисунок 2.6- Програмний комплекс Medexpert [29]

EvoMIS - EvoMIS дозволяє медустановам первинної, вторинної та третинної ланки підключитися до eHealth та комфортно працювати в новій моделі фінансування [30].

Binomed - ПЗ для медичних установ. Функціонал включає: ведення електронних карток пацієнтів, довідники лікарів та пацієнтів спільні з програмою «Електронна реєстратура», можливість створювати та редагувати свій список послуг, що надаються пацієнту [31].

Усі ці системи та ще багато інших , допомагають медичним закладам ефективно управляти пацієнтськими записами, планувати прийоми та забезпечувати високий рівень обслуговування.

Описані програмні системи допомагають не лише з медичною частиною, але й з управлінськими та адміністративними задачами, дозволяючи медичним закладам працювати ефективніше.

У сучасних клініках для побудови програмних систем управління медичними картками та історіями хвороби використовуються різноманітні технології.

Ось кілька основних:

– електронні медичні картки і історії хвороб (EMR/EHR): EMR (Електронна історія хвороби) і EHR (Електронна медична картка) є основними системами для

КОПІЯ 2024

збору, зберігання та обробки медичних даних пацієнтів. EHR містять більше даних про пацієнтів, включаючи інформацію про попередні лікарські призначення, результати лабораторних досліджень, медичну історію, алергії тощо;

– хмарні технології: Багато клінік переходять на хмарні рішення для зберігання даних, що дозволяє забезпечити доступ до інформації з різних пристроїв і локацій, зменшуючи витрати на IT-інфраструктуру;

– стандарти даних: Використання стандартів, таких як HL7 та FHIR (Fast Healthcare Interoperability Resources), для обміну даними між різними системами та забезпечення сумісності;

– аналіз даних та штучний інтелект: Технології аналізу даних і штучного інтелекту використовуються для виявлення закономірностей у медичних даних, що допомагає в діагностиці та персоналізації лікування;

– мобільні додатки: Розробка мобільних додатків для лікарів і пацієнтів, які забезпечують зручний доступ до медичних карток, нагадування про прийом ліків та можливість запису на прийом;

– блокчейн: Досліджуються можливості використання блокчейну для забезпечення безпеки та конфіденційності медичних даних, а також для полегшення обміну інформацією між постачальниками медичних послуг;

– інтерфейси програмування (API): Використання API для інтеграції різних медичних систем, що дозволяє обмінюватися даними між ними;

– кібербезпека: Зважаючи на чутливість медичних даних, важливо впроваджувати заходи з кібербезпеки, такі як шифрування даних, контроль доступу та моніторинг безпеки.

Ці технології забезпечують ефективніше управління медичними даними, покращують якість обслуговування пацієнтів і підвищують загальну ефективність роботи клінік.

КОПЗ – 2024

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Під час розробки модуля електронної реєстратури для лікувально-діагностичного центру важливо обрати технології, які забезпечать продуктивність, надійність і масштабованість системи. Враховуючи необхідність інтеграції з іншими модулями, наявність значного обсягу даних, вимоги до безпеки та вимогу ефективної роботи в умовах високих навантажень, доцільно орієнтуватися на сучасні програмні засоби та фреймворки, які можуть задовольнити такі вимоги. Обрані засоби мають забезпечити стабільну і надійну роботу всіх функціональних модулів системи, зокрема модулів реєстрації, управління даними пацієнтів, автоматизації запису та взаємодії з іншими компонентами.

Для програмування серверної частини було обрано мову Java — одну з найпоширеніших мов програмування для розробки корпоративних застосунків, у тому числі в галузі охорони здоров'я. Ця мова поєднує у собі високу продуктивність, безпеку та великий набір бібліотек і фреймворків, що прискорюють процес розробки та полегшують подальшу підтримку системи.

Основні причини, чому Java є оптимальним вибором для цього проекту, включають:

– масштабованість: Java пропонує можливості для створення масштабованих систем завдяки об'єктно-орієнтованому підходу. Це дозволяє легко розширювати функціонал системи, додаючи нові модулі або інтегруючи нові сервіси;

– безпека: У медичних системах безпека є критично важливою, адже йдеться про конфіденційні дані пацієнтів. Java забезпечує високий рівень безпеки через використання вбудованих механізмів шифрування, контролю доступу та надійного керування пам’яттю. Ці можливості роблять її одним із лідерів для створення безпечних додатків, які відповідають вимогам охорони здоров'я;

КПЗ_2024

– незалежність від платформи: Завдяки Java Virtual Machine (JVM) Java-додатки можна запускати на різних платформах. Це важливо для гнучкого розгортання системи як у внутрішніх мережах медичного закладу, так і в хмарних середовищах. Додаток може працювати на різних операційних системах, таких як Windows, Linux, або MacOS, що забезпечує гнучкість розгортання;

– великий набір бібліотек та фреймворків: Java має великий вибір бібліотек і фреймворків, що підтримують стандарти REST API, роботу з базами даних і інтеграцію з іншими сервісами. Зокрема, для проекту використовується Spring Boot, який полегшує створення RESTful-сервісів, керування даними, а також обробку запитів до бази даних.

Spring Boot є фреймворком, що надає потужний набір інструментів для розробки серверної частини корпоративних додатків. Він забезпечує швидкий старт розробки, знижуючи необхідність писати базові конфігурації вручну, що прискорює розгортання і спрощує підтримку системи. Основні переваги використання Spring Boot включають:

– модульність і легкість інтеграції: Завдяки Spring Boot можна легко розділити систему на окремі модулі, кожен з яких відповідає за свою частину функціоналу. Наприклад, модулі для роботи з базою даних, обробки запитів від фронтенду та забезпечення безпеки можуть працювати як окремі компоненти, що забезпечує високий рівень ізоляції;

– підтримка REST API: Spring Boot забезпечує зручні інструменти для створення RESTful-сервісів, що дозволяє розробити гнучкий інтерфейс для зв’язку фронтенду з бекендом. Це спрощує інтеграцію з іншими модулями медичної системи або навіть з зовнішніми системами;

– безпека і управління доступом: Spring Security з JWT — це потужний інструмент для аутентифікації та авторизації, який використовується в медичних додатках для обмеження доступу до конфіденційної інформації пацієнтів. Завдяки Spring Boot безпека стає невід’ємною частиною архітектури програми;

КПЗ – 2024

– підтримка обробки транзакцій: У медичному програмному забезпеченні важливо, щоб усі дії з базами даних були транзакційними, зокрема операції створення, читання, оновлення та видалення інформації. Spring Boot пропонує потужну підтримку транзакційності через Spring Data та JPA, що забезпечує цілісність даних у разі одночасних запитів.

Реляційна база даних PostgreSQL[32] є популярною системою управління базами даних, яка підтримує реляційну модель та забезпечує високу продуктивність при обробці великої кількості записів. PostgreSQL було обрано для модуля реєстратури через такі причини:

– Надійність і безпека: PostgreSQL підтримує ACID-транзакції, що забезпечує цілісність і захист даних. Завдяки цьому база даних ефективно підтримує медичні записи, які вимагають високої надійності.

– Можливість розширення: У PostgreSQL можна використовувати різні типи даних та індекси, що дозволяє оптимізувати базу даних для потреб конкретного проекту. Це особливо важливо для зберігання структурованих даних пацієнтів, таких як історія захворювань, медичні записи і результати аналізів.

– Підтримка складних запитів: PostgreSQL може обробляти складні SQL-запити, що дозволяє виконувати аналіз і витягування даних для створення звітів та інших аналітичних завдань.

Для реалізації інтерфейсу користувача модулю реєстратури обрано фреймворк React у поєднанні з Material-UI (MUI), оскільки ці інструменти забезпечують динамічний та зручний для користувачів інтерфейс.

Фреймворк React дозволяє створювати сучасний інтерфейс з компонентами, що оновлюються в реальному часі, що є важливим для інтерактивних додатків. Використання React дозволяє легко створювати багатофункціональні елементи, забезпечуючи зручний інтерфейс для пошуку пацієнтів, редагування записів, перегляду історії відвідувань тощо.

Фреймворк Material-UI (MUI) забезпечує готові компоненти користувацького інтерфейсу, які відповідають сучасним стандартам дизайну та є

КБПЗ-2024

інтуїтивно зрозумілими для користувачів. Використання MUI допомагає підтримувати єдиний стиль дизайну, що підвищує зручність використання для медичного персоналу.

Взаємодія між компонентами системи відбувається за допомогою REST API, який забезпечує зв'язок між фронтендом і бекендом. Бекенд у свою чергу підключається до бази даних, що дозволяє ефективно обробляти запити від користувачів. Інтеграція компонентів відбувається через стандартні протоколи, що спрощує масштабування і підтримку системи, а також дозволяє додавати нові функції.

Вибрані технології забезпечують швидку розробку, безпечне зберігання даних і зручний для користувачів інтерфейс, що є ключовими факторами для створення ефективної системи електронної реєстратури.

КПЗ - 2024

2.3 Розгорнута постановка завдання

В умовах сучасної охорони здоров’я ефективне управління потоком пацієнтів та оптимізація внутрішніх процесів у клініці відіграють важливу роль для забезпечення високої якості медичних послуг. Програмний модуль "е-реєстратура" покликаний вирішити низку завдань, пов'язаних із реєстрацією, обліком та управлінням даними пацієнтів. Метою цього проєкту є створення функціонального, гнучкого та безпечного модуля, який би інтегрувався з іншими інформаційними системами клініки та забезпечував доступ до медичної інформації в режимі реального часу.

"е-реєстратура" має на меті замінити або значно спростити традиційні процеси реєстрації, автоматизувати рутинні завдання та мінімізувати людський фактор, що часто стає причиною помилок або втрати даних. Розробка цього модуля передбачає врахування різноманітних сценаріїв роботи та функціональних вимог для забезпечення високої зручності та простоти використання системи як для працівників реєстратури, так і для пацієнтів.

Ефективне функціонування е-реєстратури має базуватися на певних вимогах, серед яких:

– Автоматизація процесів реєстрації пацієнтів: Модуль має підтримувати зручний інтерфейс для швидкої реєстрації нових пацієнтів та коригування даних наявних пацієнтів, що зменшує тривалість обробки кожного запиту та підвищує точність введених даних.

– Управління записами на прийом: Система повинна мати можливість запису пацієнтів на прийом до лікаря, відслідковувати доступні часи та керувати розкладом лікарів. Це передбачає наявність календаря з можливістю інтеграції розкладу з іншими модулями, такими як модуль "поліклініка".

– Обробка історії відвідувань: Історія відвідувань пацієнтів має зберігатися в системі для забезпечення послідовного лікування та доступності попередніх записів, включаючи історії захворювань, проведені консультації, лабораторні тести та інші записи, що можуть знадобитися лікарям.

– Інтеграція з іншими модулями: Оскільки е-реєстратура є лише одним із компонентів комплексної інформаційної системи клініки, вона повинна мати можливість інтегруватися з іншими модулями, такими як стаціонарне лікування, лабораторія, поліклініка, для обміну даними та надання повної картини стану пацієнта.

Функціональність модуля включає:

– Реєстрація та управління обліковими записами пацієнтів - це функціональність створення профілів нових пацієнтів з основними даними (ПІБ, дата народження, контактна інформація) та можливість редагування наявних профілів.

– Робота з базою даних пацієнтів спрямовується для зручного та швидкого доступу до інформації про пацієнтів модуль повинен підтримувати потужний пошук та фільтрацію даних. Наприклад, швидке знаходження пацієнта за ПІБ, номером документа або датою останнього візиту.

КПЗ-2024

– Керування чергами та розкладом лікарів повинна мати функцію запису на прийом до лікаря з можливістю автоматичного налаштування часу прийому, а також підтримувати функціональність відстеження та управління чергами у реальному часі. Це дозволить оптимізувати потік пацієнтів, скорочуючи час очікування та підвищуючи задоволеність пацієнтів.

– Безпека і доступ до даних повинні забезпечувати захищений доступ до даних пацієнтів, дотримуючись вимог стандартів безпеки (наприклад, HIPAA для західних країн). Для цього передбачено використання механізмів аутентифікації користувачів, обмеження доступу до певних даних та шифрування.

– Повідомлення та нагадування для пацієнтів слугує для покращення взаємодії з пацієнтами модуль повинен підтримувати функціональність відправлення нагадувань про майбутні візити, а також інформування про зміни в розкладі прийомів.

Також існують нефункціональні вимоги до модуля:

– Продуктивність та масштабованість: Модуль повинен підтримувати одночасний доступ значної кількості користувачів (пацієнтів та працівників клініки) і забезпечувати швидке виконання операцій без затримок. Для цього обрана технологія мікросервісної архітектури з використанням Java та Spring Boot, що забезпечує можливість масштабування.

– Доступність та надійність: З урахуванням значної важливості інформації про пацієнтів та розкладів, система повинна мати високу доступність і забезпечувати мінімальні ризики втрати даних. Система повинна функціонувати цілодобово з мінімальним часом простою.

– Простота у використанні та ергономічність: Інтерфейс модуля повинен бути інтуїтивно зрозумілим та ергономічним для користувачів (адміністраторів, працівників реєстратури), щоб звести до мінімуму час на навчання персоналу та забезпечити високу швидкість роботи.

Для реалізації модуля е-реєстратури буде використано набір сучасних технологій:

КПЗ–2024

– Backend: Java та Spring Boot для розробки серверної частини забезпечують стабільну роботу та високу продуктивність при обробці великої кількості запитів, а також підтримують RESTful API для зручної інтеграції з іншими модулями та зручної передачі даних.

– Frontend: Інтерфейс буде розроблений на базі JavaScript і React, що забезпечить зручність користування, можливість оновлення компонентів в реальному часі, а також адаптивність для роботи на різних пристроях.

– База даних: PostgreSQL буде використовуватися для зберігання даних про пацієнтів і записів, забезпечуючи високу надійність і гнучкість роботи з даними.

Процес створення модуля передбачає кілька етапів:

– Аналіз вимог і проектування: На цьому етапі аналізуються функціональні і нефункціональні вимоги, формується загальна архітектура, розподіляються завдання і визначаються засоби для реалізації.

– Розробка бази даних: Створення бази даних, яка зберігатиме інформацію про пацієнтів, записи на прийоми, черги та інші важливі дані.

– Розробка серверної частини: Реалізація функціоналу бекенду з використанням Spring Boot, побудова REST API для зв’язку між сервером і клієнтською частиною.

– Розробка інтерфейсу користувача: Створення фронтенду з використанням React, що забезпечить зручний доступ до всіх функцій модуля для співробітників реєстратури.

– Інтеграція і тестування: Інтеграція з іншими модулями та тестування на наявність помилок, перевірка функціональності та надійності.

– Розгортання та підтримка: Розгортання системи в умовах реального середовища, а також забезпечення постійної підтримки та оновлення системи.

Очікується, що після впровадження модуля "е-реєстратура" клініка зможе значно спростити процес реєстрації та управління даними пацієнтів, підвищити продуктивність співробітників реєстратури та загальну якість обслуговування пацієнтів. Система забезпечить зручність використання для адміністративного

КПЗ – 2024

персоналу, лікарів та пацієнтів, що позитивно вплине на ефективність роботи клініки в цілому.

КбПЗ_2024

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Модуль реєстрації пацієнтів, або е-реєстратура, є центральним компонентом системи електронного обліку, який забезпечує автоматизацію процесів прийому, обліку та управління інформацією про пацієнтів у медичному закладі. Його основна функція полягає в тому, щоб оптимізувати роботу реєстратури, знижуючи навантаження на персонал і спрощуючи пацієнтам доступ до медичних послуг. Даний модуль поєднує в собі кілька ключових функціональних можливостей, які забезпечують повний цикл обробки даних, починаючи з первинної реєстрації пацієнта і закінчуючи записом на консультації, процедури або лікування.

Першим і одним із найбільш важливих етапів функціонування модуля є реєстрація нових пацієнтів. Під час першого звернення пацієнта до медичного закладу, медичний персонал за допомогою е-реєстратури створює електронну картку пацієнта, яка містить базову інформацію: прізвище, ім'я, по батькові, дату народження, контактні дані, інформацію про місце проживання, та, за потреби, медичну інформацію (групу крові, відомості про алергії, хронічні захворювання тощо). Цей процес також включає унікальну ідентифікацію пацієнта у системі, що забезпечує точність та унеможливлює дублювання записів. Модуль дозволяє створити унікальний ID або QR-код, який надалі може бути використаний для ідентифікації пацієнта під час його відвідувань, обстежень та інших процедур.

Завдяки цьому підходу, створення нової картки стає швидким і точним процесом, який виключає ризики помилок і дозволяє отримувати доступ до всієї історії пацієнта в кілька переходів.

Модуль реєстрації забезпечує ефективні інструменти пошуку пацієнтів у базі даних, що є важливою частиною роботи е-реєстратури. Система дозволяє шукати пацієнта за різними параметрами, такими як прізвище, ім'я, дата

КОПЗ_2024

народження, унікальний ідентифікатор або номер телефону. Це особливо корисно, коли йдеться про роботу з великими обсягами даних і необхідність швидко знаходити потрібну інформацію. Сучасні алгоритми пошуку можуть також використовувати частковий збіг даних, що дозволяє знаходити пацієнтів навіть за неповною інформацією.

Управління даними включає в себе редагування інформації про пацієнта, оновлення контактних даних або записів про медичні процедури. Кожна зміна запису автоматично відображається в історії, що дозволяє лікарям та медичному персоналу отримати доступ до актуальних даних, а також відстежувати всі зміни, які вносяться до картки пацієнта.

Е-реєстратура значно спрощує процес запису пацієнта на прийом до лікаря, забезпечуючи швидкий та інтуїтивний інтерфейс для вибору дати, часу і спеціаліста. Пацієнти можуть записатися на прийом як у відділенні медичного закладу, так і онлайн через веб-портал або мобільний додаток, інтегрований із системою реєстрації. Модуль дозволяє переглядати графіки лікарів у реальному часі, що знижує кількість помилок у записах та виключає дублювання.

Ця функція підвищує ефективність роботи персоналу та зменшує час очікування для пацієнтів, створюючи більш зручні умови для їх обслуговування. Крім того, можливість попереднього запису онлайн розвантажує медичний персонал, дозволяючи пацієнтам самостійно обирати зручний час для візиту, що в свою чергу позитивно впливає на їхній досвід користування медичними послугами.

Модуль реєстрації також дозволяє зберігати історію відвідувань пацієнта та його медичних процедур. Кожен візит до лікаря, проведене обстеження чи призначення фіксується у відповідному записі. Це дозволяє лікарям отримати доступ до детальної історії хвороби пацієнта і, таким чином, приймати обґрунтовані рішення щодо лікування.

Історія прийомів також дозволяє легко відстежувати частоту звернень пацієнта, його перебування на стаціонарному лікуванні та інші аспекти медичної

КПЗ-2024

допомоги, що особливо корисно при лікуванні хронічних захворювань. Окрім того, цей підхід полегшує аналітику і формування звітності, що є цінним ресурсом для оптимізації роботи медичного закладу.

Функціонування модуля е-реєстратури передбачає тісну інтеграцію з іншими компонентами медичної інформаційної системи, такими як лабораторний модуль, стаціонарне відділення, їдальня, відділ обліку медикаментів тощо. Наприклад, якщо пацієнт направлений на лабораторне дослідження, е-реєстратура автоматично передає необхідну інформацію в лабораторний модуль, де фіксуються результати аналізів, доступні для лікаря у відповідний момент. Інтеграція з їдальнею дозволяє автоматично враховувати пацієнтів, що перебувають на стаціонарному лікуванні, та фіксувати їхні потреби у харчуванні.

Ця інтеграція дозволяє створити єдину інформаційну базу, яка містить усі дані про пацієнта, його обстеження, призначення та лікування, забезпечуючи повну взаємодію між усіма підрозділами медичного закладу. Це не тільки підвищує ефективність обслуговування, але й мінімізує ризик втрати інформації.

Ще однією важливою функцією модуля реєстрації є автоматизація сповіщень і нагадувань для пацієнтів. Система може надсилати повідомлення про наближення дати прийому або про необхідність пройти певне обстеження. Це суттєво покращує комунікацію між медичним закладом і пацієнтами, підвищуючи їхню відповідальність за своє здоров'я та знижуючи ризик пропуску прийомів.

Система сповіщень дозволяє також лікарям отримувати повідомлення про зміни у графіку роботи або нагадування про планові прийоми пацієнтів. Це дає змогу більш гнучко керувати часом, а також підвищити ефективність робочих процесів у медичному закладі.

Модуль е-реєстратури забезпечує формування звітів та аналітичних даних, які допомагають оцінювати роботу закладу, відслідковувати кількість прийомів, виявляти затримки чи невідповідності у роботі персоналу, визначати популярність певних спеціалістів або обстежень. Такі звіти є цінним джерелом

КОПЗ-2024

інформації для управлінського персоналу, дозволяючи оптимізувати робочі процеси і планувати ресурси медичного закладу.

Завдяки використанню електронної реєстратури можна отримувати точні дані у реальному часі, що сприяє більш ефективному управлінню закладом і поліпшенню якості медичного обслуговування.

Таким чином, модуль е-реєстратури є незамінним інструментом для автоматизації та оптимізації роботи медичних закладів. Його функціональні можливості дозволяють значно підвищити точність і швидкість обробки даних, забезпечуючи якісне та зручне обслуговування пацієнтів, а також покращуючи умови роботи медичного персоналу.

Фронтенд модуля "е-реєстратура" є першим рівнем, з яким взаємодіють користувачі — реєстратори, пацієнти та медичний персонал. Важливими складовими цього рівня є:

– Форма реєстрації пацієнта: Забезпечує інтерфейс для введення особистих і контактних даних пацієнта, таких як ім’я, дата народження, адреса, контактний номер телефону, історія захворювань (при потребі) тощо.

– Пошук пацієнта: Дозволяє реєстратору або медичному персоналу швидко знайти потрібного пацієнта за декількома критеріями, такими як ім’я, дата народження, номер картки чи ID.

– Календар запису на прийом: Ця частина інтерфейсу дозволяє обирати дату, час і спеціаліста для запису пацієнта на консультацію. Вона інтегрована з графіком роботи лікарів, щоб уникнути дублювання записів.

– Система сповіщень: Забезпечує користувачів (як медичний персонал, так і пацієнтів) повідомленнями про майбутні записи, зміну графіку лікарів чи необхідність оновлення особистих даних.

Фронтенд побудований із використанням технологій JavaScript і React, що забезпечує інтерактивність, швидкість реакції на дії користувача та адаптивність до різних типів пристроїв.

КПЗ-2024

API (Application Programming Interface) виступає посередником між фронтендом і бекендом. Він забезпечує обробку запитів від інтерфейсу користувача та передачу даних на сервер для подальшої обробки. API також дозволяє обмежити доступ до інформації залежно від ролі користувача, забезпечуючи захист чутливих даних. Основні функції API:

– обробка запитів на створення або оновлення картки пацієнта,

– обробка запитів на запис до лікаря,

– передача запитів на пошук пацієнтів або оновлення інформації,

– забезпечення захисту даних через токени доступу та шифрування,

– API розробляється із використанням RESTful технологій для ефективного обміну даними.

Бекенд є ядром модуля "е-реєстратура", де зберігається та обробляється вся інформація. Це комплекс, який включає такі основні компоненти:

– Модуль управління пацієнтами: Відповідає за створення, збереження, оновлення та видалення записів пацієнтів. Це дозволяє медичному персоналу створювати нові записи та забезпечує безпомилковий облік.

– Модуль управління записами: Зберігає інформацію про записи пацієнтів на прийом до лікарів, зокрема час, дату та спеціаліста. Система також відстежує доступність лікарів і запобігає можливим конфліктам записів.

– Модуль історії прийомів та медичних процедур: Важливий компонент, який зберігає всю інформацію про історію відвідувань і медичних процедур пацієнта. Він забезпечує повний доступ лікарям до медичних записів, що допомагає приймати обґрунтовані рішення.

– Система безпеки та управління ролями: Впроваджує обмеження доступу до даних залежно від ролі користувача (лікар, реєстратор, адміністратор). Включає функції аутентифікації, авторизації та шифрування.

Бекенд реалізований на мові програмування Java із застосуванням фреймворку Spring Boot, що забезпечує надійність, високу продуктивність та можливість масштабування.

КПЗ-2024

База даних — це ключовий елемент, що відповідає за зберігання всіх даних модуля "е-реєстратура". Вона включає такі основні таблиці:

– Таблиця пацієнтів: Містить особисту, контактну та медичну інформацію про кожного пацієнта.

– Таблиця записів: Зберігає дані про майбутні та минулі прийоми пацієнтів, включаючи дату, час та лікаря, до якого записаний пацієнт.

– Таблиця історії прийомів: Містить інформацію про всі обстеження, аналізи та лікування, які були проведені.

– Таблиця лікарів та спеціалістів: Включає контактну інформацію про лікарів, спеціалізацію та графік роботи.

– Журнал аудиту(логгер): Записує всі зміни, внесені до бази даних, що дозволяє відстежувати дії користувачів та захищати інформацію.

– PostgreSQL, що використовується як СУБД, забезпечує надійне зберігання даних із можливістю масштабування.

Е-реєстратура часто інтегрується з іншими модулями, такими як лабораторний модуль, модуль стаціонарного лікування, система обліку їдальні та фармацевтичний модуль. Інтеграція здійснюється через внутрішні API, що дозволяє автоматично передавати інформацію про пацієнта до інших систем без необхідності дублювання даних. Це значно підвищує ефективність та точність роботи закладу.

Система сповіщень дозволяє автоматично надсилати повідомлення пацієнтам про наближення дати прийому, заплановані обстеження або зміну графіку прийому лікаря. Це підвищує зручність для пацієнтів, знижуючи ризик пропуску прийому.

Взаємодія компонентів у структурній схемі:

– Реєстрація нового пацієнта: Користувач (реєстратор) через фронтенд вводить дані нового пацієнта. API передає ці дані на бекенд, де вони обробляються та записуються в базу даних. Унікальний ідентифікатор (ID) пацієнта генерується автоматично.

КПЗ-2024

– Запис пацієнта на прийом: Реєстратор або пацієнт обирає дату та час для запису на прийом через фронтенд. Інтерфейс запису пов’язаний із календарем лікарів, що дозволяє уникнути дублювань. Запис через API передається на сервер, де перевіряється доступність часу і лікаря, після чого дані фіксуються в базі даних.

– Пошук пацієнта та перегляд історії: Реєстратор або лікар може здійснити пошук пацієнта за ім’ям, ID або датою народження. Запит надходить на API, який передає його на сервер, де відбувається пошук у базі даних. Відповідна інформація повертається та відображається на інтерфейсі.

– Передача даних до інших модулів: У разі потреби інформація з модуля "е-реєстратура" автоматично передається в лабораторію або стаціонарне відділення. Це здійснюється через внутрішні API, які дозволяють синхронізувати дані між модулями.

– Обробка сповіщень: Система сповіщень надсилає повідомлення пацієнтам і лікарям про наближення прийомів або про важливі події. Інформація для сповіщень отримується з бази даних на бекенді та передається через API на сторонні сервіси, які надсилають короткі повідомлення або email.

Розроблена структурна схема модуля "е-реєстратура" дозволяє забезпечити високий рівень автоматизації процесів реєстрації та запису пацієнтів. Вона враховує всі необхідні аспекти захисту даних і зручність використання для медичного персоналу та пацієнтів.

КПЗ – 2024

3.2 Структурна схема модуля

Модуль реєстратури є ключовим компонентом електронної системи управління лікувальним закладом, який забезпечує реєстрацію, збереження та обробку інформації про пацієнтів. Його структурна схема є відображенням компонентів і зв’язків між ними, що забезпечують коректне функціонування модуля.

Основні компоненти модуля реєстратури:

Інтерфейс користувача забезпечує взаємодію реєстратора або адміністратора з системою. Основні функції:

-введення даних про пацієнтів;

-пошук у базі даних за заданими параметрами;

-редагування інформації;

-запис на прийом до лікаря.

Інтерфейс розробляється з використанням React і бібліотеки компонентів MUI для забезпечення сучасного і зручного дизайну.

Сервер обробляє запити від інтерфейсу користувача, забезпечує логіку бізнес-процесів та взаємодіє з базою даних. Його функції:

– обробка запитів CRUD (створення, читання, оновлення, видалення даних);

– перевірка вхідних даних;

– забезпечення безпеки (аутентифікація та авторизація користувачів);

– обробка помилок.

Сервер реалізовано за допомогою Spring Boot, з використанням ORM-бібліотеки Hibernate для доступу до бази даних.

База даних використовується для зберігання всієї інформації про пацієнтів, лікарів, запис на прийом та історії взаємодій. Для реалізації обрано реляційну базу даних PostgreSQL, яка забезпечує високу продуктивність і підтримку складних запитів.

Структура бази даних включає такі основні таблиці:

– patients: зберігає дані про пацієнтів (ПІБ, дата народження, контактна інформація);

– appointments: дані про записи до лікаря;

– users: інформація про користувачів системи (логіни, паролі, ролі);

– doctors: перелік лікарів і їх спеціалізацій.

КОПЗ_2024

API забезпечує комунікацію між фронтендом та бекендом. Основні ендпоінти включають:

/patients: створення, редагування, видалення та перегляд пацієнтів;

/appointments: створення запису до лікаря;

/doctors: отримання списку лікарів і їх розкладів;

/auth: аутентифікація користувачів.

Використовується REST-архітектура, що забезпечує легкість інтеграції з іншими модулями системи.

Реалізовано за допомогою JWT (JSON Web Token). Цей модуль забезпечує:

– перевірку прав доступу до функціоналу;

– шифрування і захист даних користувача.

JWT-токени зберігаються на клієнтській стороні для зменшення навантаження на сервер.

Цей компонент відповідає за збереження записів про події у системі: успішні або невдалі спроби авторизації, зміни даних, створення нових записів. Інструменти моніторингу, такі як Spring Actuator або зовнішні сервіси, наприклад Prometheus або Grafana, використовуються для відстеження стану системи.

Клієнтська частина (React) відправляє HTTP-запити на сервер (Spring Boot) через REST API. Наприклад, під час реєстрації нового пацієнта інтерфейс передає дані, які сервер перевіряє та записує в базу.

Бекенд обробляє SQL-запити до PostgreSQL за допомогою Hibernate. Це забезпечує абстракцію над запитами та мінімізує ризик помилок у роботі з базою даних.

Під час авторизації сервер перевіряє дійсність JWT-токенів, які клієнт надсилає у заголовках запитів. Якщо токен є недійсним, доступ до системи блокується.

Модуль реєстратури може взаємодіяти з іншими сервісами (наприклад, лабораторією або поліклінікою) через API, що дозволяє передавати дані про пацієнтів або записи.

КОПЗ-2024

Опис функціонування:

– Реєстрація нового пацієнта: Реєстратор вводить дані пацієнта в інтерфейс. Дані відправляються на сервер, де вони перевіряються на валідність. Якщо дані коректні, сервер зберігає їх у базу даних та повертає повідомлення про успіх.

– Пошук пацієнта:Реєстратор вводить параметри пошуку (наприклад, прізвище, дату народження). Сервер виконує SQL-запит до бази даних і повертає список результатів. Результати відображаються у вигляді таблиці в інтерфейсі користувача.

– Запис пацієнта до лікаря: Реєстратор вибирає лікаря та вільний час у його графіку. Сервер перевіряє, чи цей час доступний, та створює запис у базі даних. Пацієнту видається підтвердження про запис.

– Авторизація користувачів: При вході в систему користувач вводить логін і пароль. Сервер генерує JWT-токен, який клієнт зберігає та надсилає при кожному запиті. Система перевіряє права доступу користувача на виконання певних дій.

КОПЗ – 2024

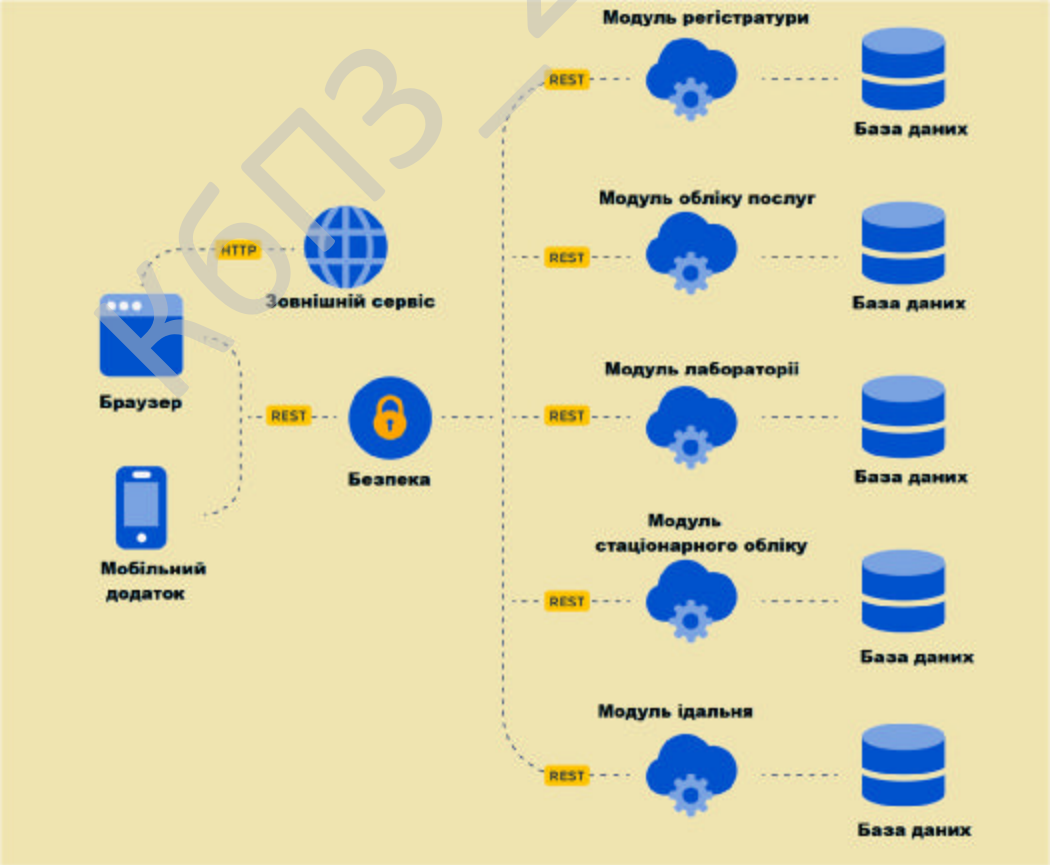


Рисунок 3.1 - Загальна архітектура ІС ЛДЦ

Переваги розробленої структури:

– Гнучкість: Завдяки використанню модульної архітектури легко оновлювати окремі компоненти без впливу на інші частини системи,

– Безпека: Використання JWT гарантує надійний рівень захисту даних користувачів,

– Масштабованість: Використання PostgreSQL забезпечує обробку великої кількості даних, а React дозволяє створювати інтерфейси з високою продуктивністю,

– Легкість інтеграції: REST API забезпечує зручність взаємодії між модулями.

Структурну схему модуля, який розробляється, представлено на рисунку 3.2.

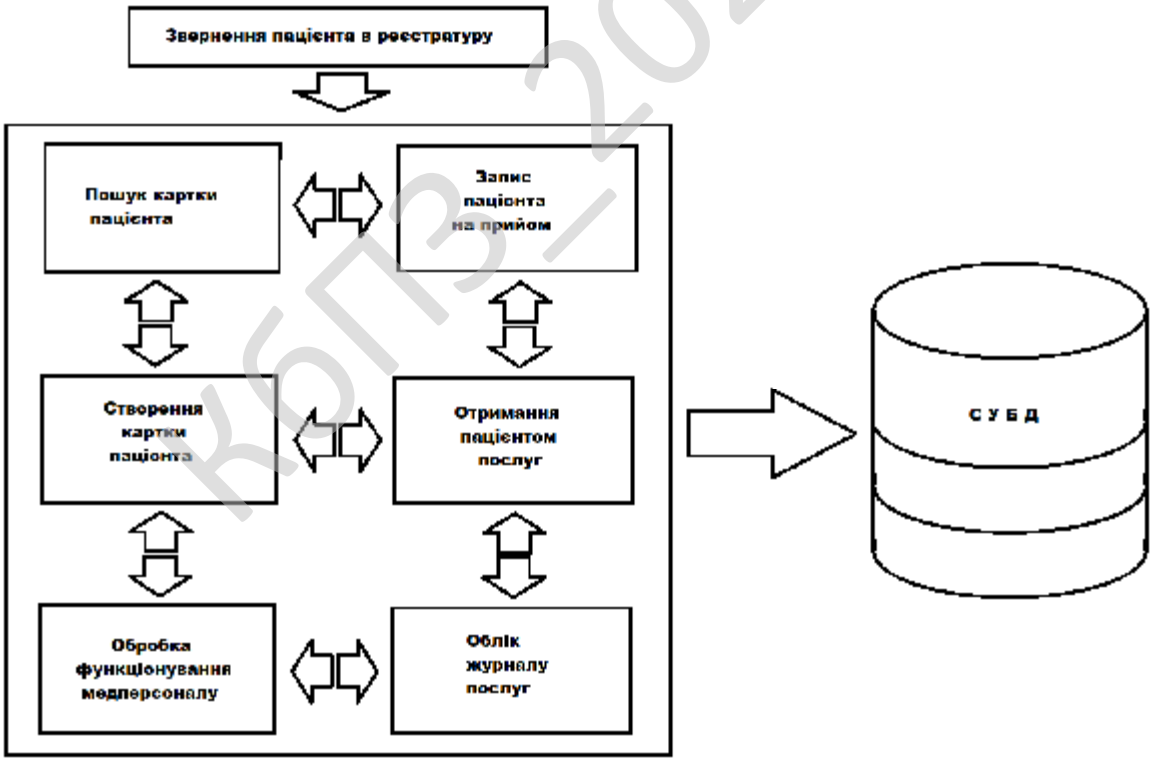


Рисунок 3.2- Структурна схема модуля

Такий підхід дозволяє створити надійну, сучасну та зручну систему реєстрації пацієнтів у медичних установах.

КПЗ-2024

3.3 Розробка функціональної схеми

Функціональна схема модуля "е-реєстратура" у медичному закладі орієнтована на автоматизацію процесів реєстрації, обробки та зберігання даних пацієнтів. Її мета — забезпечити швидкий та зручний доступ до реєстрації пацієнтів для медичного персоналу, а також для пацієнтів, які бажають записатися на прийом до лікаря онлайн. Даний модуль будується на мікросервісній архітектурі, з відокремленим frontend-ом і backend-ом, що дозволяє масштабувати та оновлювати окремі компоненти без значного впливу на загальну функціональність системи.

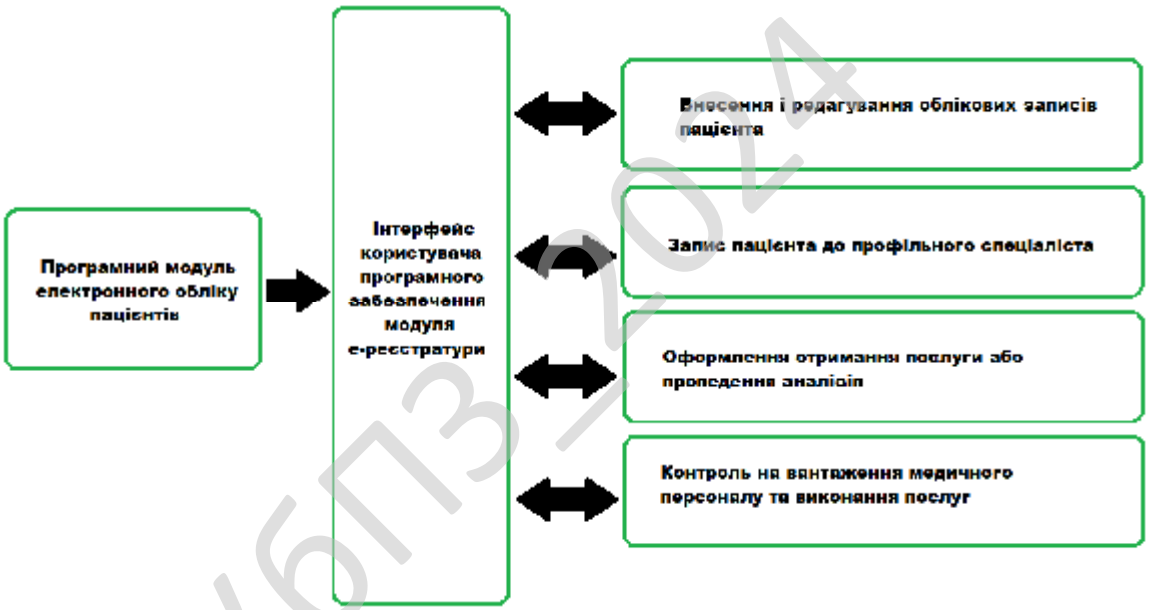


Рисунок 3.3 - Функціональна схема системи

Розглянемо основні компоненти та їх функції. Перший основний компонент це Фронтенд , який базується на технологіях React, MUI (Material-UI); функції: Фронтенд компонент забезпечує взаємодію користувачів з модулем "е-реєстратура". Використання React дозволяє створювати інтерактивний інтерфейс, який реагує на дії користувача в режимі реального часу без перезавантаження сторінки. MUI додає стилізацію компонентів, роблячи інтерфейс інтуїтивно зрозумілим та візуально привабливим.

КПЗ 2024

Основні компоненти інтерфейсу:

– Форма реєстрації пацієнтів: Збір даних пацієнтів (ПІБ, дата народження, адреса, контактна інформація). Ця форма містить валідацію полів та інтеграцію з backend-ом для перевірки наявності пацієнта в базі.

– Форма запису на прийом до лікаря: Пацієнти можуть вибрати лікаря, час і тип прийому. Інтерфейс відображає доступні дати та часи, інтегровані з розкладом лікарів.

– Система авторизації та аутентифікації: Дозволяє персоналу входити до системи, щоб мати доступ до даних пацієнтів та можливостей редагування записів. Також можливий доступ для пацієнтів для перегляду записів.

– Сповіщення: Компоненти для відображення сповіщень про успішне або невдале завершення дій (наприклад, успішна реєстрація або помилка через відсутність місць на прийом).

Другий важливий компонент - це Backend . Використовує технології Java, Spring Boot, JPA, Hibernate.

Функції: Backend забезпечує логіку бізнес-процесів, управління даними та обробку запитів від frontend-а. Spring Boot дозволяє будувати RESTful API, через які frontend взаємодіє з backend-ом, а також підтримує інші модулі та забезпечує інтеграцію з базою даних.

КбПЗ-2024

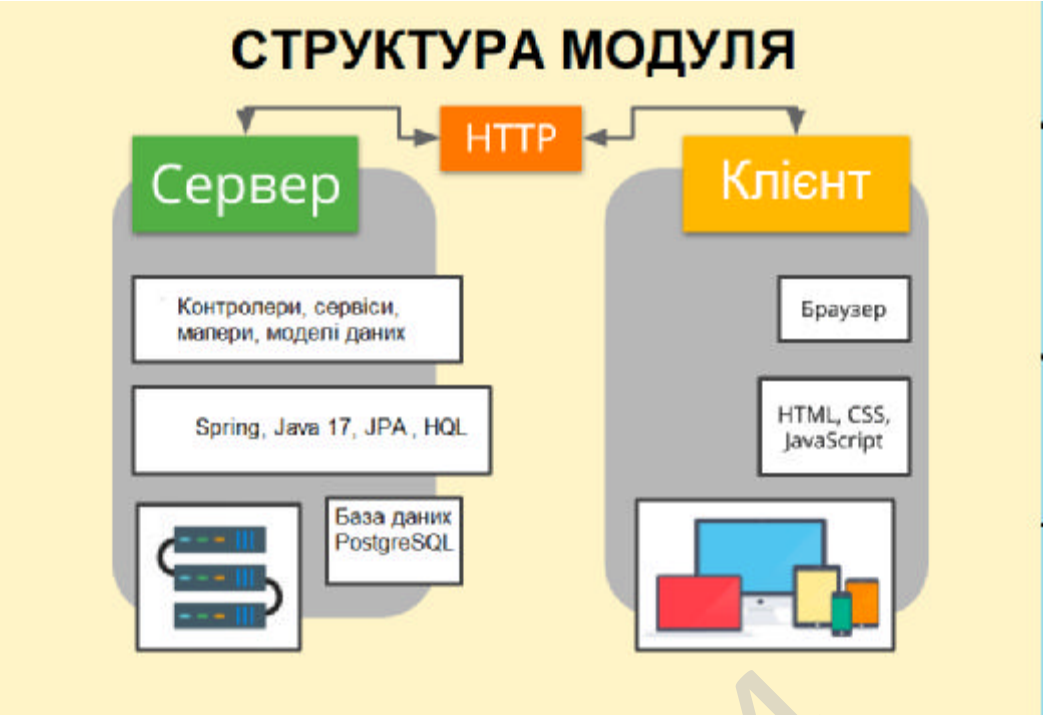


Рисунок 3.4 – Схема модуля

КОПЗ_2024

Основні функціональні сервіси:

– Сервіс реєстрації пацієнтів: Приймає дані від frontend-а, перевіряє їх на коректність і зберігає у базі даних. Використання JPA і Hibernate спрощує взаємодію з базою даних, абстрагуючи SQL-запити і забезпечуючи роботу з об'єктами.

– Сервіс запису на прийом: Перевіряє наявність лікаря на обраний час, а також перевіряє кількість пацієнтів для уникнення перевантаження. За успішної перевірки запис зберігається у базі.

– Сервіс аутентифікації: Spring Security забезпечує захист API від несанкціонованого доступу, а також реалізує логіку авторизації користувачів.

– Сервіс для обробки запитів пацієнтів: Обробляє запити на перегляд записів або зміну інформації, і передає результати на frontend.

– Система сповіщень: Генерує сповіщення на основі дій пацієнта чи персоналу, такі як підтвердження запису або нагадування про прийом. Сповіщення можуть надсилатися через інтеграцію з поштовими сервісами або SMS API.

Наступним важливим компонентом є СУБД. Тутобрано популярну систему PostgreSQL.

Функції: Зберігає усі дані, пов’язані з пацієнтами, розкладами прийомів, історією відвідувань, а також дані авторизації користувачів. PostgreSQL вибрано за його надійність, відповідність стандартам SQL, а також підтримку розширених функцій безпеки та обробки великих обсягів даних.

Інтеграція Frontend та Backend буде здійснюватись через REST API.

Функції: Взаємодія між frontend-ом і backend-ом відбувається за допомогою REST API. Запити відправляються у форматі JSON, а backend відповідає JSON-об’єктами, що містять дані для відображення на frontend-і. Наприклад, коли пацієнт реєструється на прийом, його запит спершу перевіряється на backend-і, а потім API повертає підтвердження, що відображається у інтерфейсі.

Основні API-методи:

– POST /внесення даних,

– PUT /зміна даних,

– GET /отримання даних,

– DELETE /видалення даних.

Безпека є критично важливою для медичних систем, тому застосовуються кілька рівнів захисту, що забезпечують конфіденційність даних пацієнтів:

– Spring Security забезпечує контроль доступу до API на backend-і, обмежуючи доступ до медичних даних тільки для авторизованих користувачів,

– Шифрування даних: Використання HTTPS для передачі даних забезпечує захист від перехоплення,

– Захист паролів: Паролі користувачів хешуються та зберігаються у захищеному вигляді в базі даних,

Функціональна схема модуля "е-реєстратура" створює повноцінну інфраструктуру для автоматизації процесів реєстрації пацієнтів, запису на прийом та ведення записів. З використанням таких технологій, як Java, React, Spring Boot,

КПЗ_2024

PostgreSQL, JPA та Hibernate, система стає гнучкою, масштабованою і здатною до інтеграції з іншими медичними сервісами.

3.4 Розробка діаграми процесів

Діаграма процесів для модуля "е-реєстратура" в медичному закладі дає загальне уявлення про взаємодію між ключовими учасниками процесу та етапи, через які проходять дані пацієнтів від моменту реєстрації до запису на прийом. Нижче наведено детальний опис процесів та етапів, які відображатиме така діаграма.

Перш ніж перейти до опису самої діаграми процесів, розглянемо основні ролі, які беруть участь у функціонуванні модуля реєстратури:

– Пацієнт: особа, яка звертається до медичного закладу для отримання консультації або лікування.

– Адміністратор реєстратури: медичний персонал, який виконує функції реєстрації пацієнтів, запису на прийом, а також обробки базових даних пацієнта.

– Лікар: медичний фахівець, який надає консультації, призначає лікування та проводить обстеження.

– Система е-реєстратури: програмний модуль, який зберігає дані про пацієнтів, обробляє запити на запис до лікаря, веде журнал прийомів та історію звернень.

Основні процеси в модулі "е-реєстратура":

Процес 1: Реєстрація нового пацієнта.

Старт процесу – Адміністратор реєстратури отримує від пацієнта персональну інформацію: ПІБ, дату народження, контактні дані та інформацію про документи.

Перевірка наявності пацієнта – Система перевіряє базу даних, щоб визначити, чи раніше пацієнт вже реєструвався у клініці. Це необхідно, щоб уникнути дублювання записів.

КПЗ–2024

Занесення інформації у базу – Якщо пацієнта немає в системі, адміністратор вводить всі дані у форму на frontend-і (React, MUI), яка передає інформацію на backend (Java, Spring Boot) для збереження.

Генерація ідентифікатора – Система генерує унікальний ідентифікатор для пацієнта, який зберігається у базі даних (PostgreSQL) і надалі використовуватиметься для ідентифікації пацієнта.

Виведення підтвердження – Адміністратор отримує сповіщення про успішну реєстрацію, а також дані пацієнта для видачі йому або її особистого ID та інших документів (якщо потрібно).

Завершення процесу – Процес завершується, і пацієнт може перейти до наступних етапів, наприклад, запису на прийом.

Процес 2: Запис на прийом до лікаря.

Старт процесу – Адміністратор або пацієнт самостійно ініціює процес запису на прийом до лікаря.

Вибір лікаря та часу прийому – Система відображає розклад доступних лікарів та їхні спеціалізації, щоб пацієнт міг вибрати зручний час. Інформація формується з бази даних, що дозволяє уникнути накладення записів.

Перевірка доступності часу – Після вибору часу система перевіряє доступність обраного лікаря на цей час і місце у розкладі.

Реєстрація запису – У разі підтвердження доступності, інформація про запис пацієнта (ID пацієнта, час, лікар) зберігається у базі даних.

Сповіщення про підтвердження запису – Система генерує сповіщення для пацієнта про успішний запис на прийом.

Процес 3: Авторизація та аутентифікація користувачів.

Старт процесу – Користувач (адміністратор чи лікар) вводить свої облікові дані для доступу до модуля.

Перевірка даних – Система аутентифікації перевіряє логін та пароль користувача.

КПЗ-2024

Надання доступу до функціоналу – У разі успішної перевірки користувач отримує доступ до свого профілю і необхідного функціоналу в залежності від своєї ролі.

Процес 4: Управління та обробка даних пацієнтів.

Доступ до історії пацієнта – Авторизовані користувачі, такі як лікарі та адміністрація, можуть переглядати історію звернень пацієнта, його медичні записи.

Оновлення даних пацієнта – У разі змін в даних (наприклад, зміна номера телефону чи адреси), адміністратор вносить оновлення в базу.

Закриття або архівування записів – При завершенні обслуговування дані можуть бути позначені як архівні для подальшого зберігання відповідно до політики закладу.

Процес 5: Журнал послуг.

Формування журналу послуг – Система дозволяє адміністраторам і лікарям мониторити послуги пацієнтів, розклад та інші параметри, що зберігаються у базі даних.

Аналітичні функції – Система надає можливості для аналізу даних про потік пацієнтів, завантаженість лікарів та інші аспекти, що допомагають керівництву приймати стратегічні рішення.

3. Опис компонентів функціональної схеми

Графічний інтерфейс (Frontend).

Забезпечує зручний доступ до реєстрації та запису на прийом.

Реалізується на базі React з використанням Material-UI для стилізації.

Включає в себе форми для збору інформації, розклад лікарів, валідацію даних, сповіщення та інші компоненти для інтерактивної взаємодії.

Серверна частина (Backend).

Реалізована на базі Java, Spring Boot.

Обробляє запити від frontend-а, керує логікою перевірки, зберігає дані та взаємодіє з базою даних.

КОПЗ-2024

Рисунок 3.5 - Діаграма взаємодії процесів

Виконує аутентифікацію користувачів, управління розкладом та реєстрацією, обробку запитів пацієнтів та інших учасників.

База даних (PostgreSQL).

Зберігає всі дані про пацієнтів, розклад, записи на прийом та історію звернень.

Використовує JPA та Hibernate для зручної взаємодії з даними на рівні об'єктів.

Реалізує зв’язки між таблицями (пацієнти, лікарі, розклад), які забезпечують швидкий та надійний доступ до інформації; API для взаємодії між компонентами, REST API - для комунікації між frontend-ом і backend-ом.

Включає основні ендпоінти для реєстрації, запису, отримання даних пацієнта, отримання розкладу та аутентифікації.

КОПЗ-2024

Функціональна схема модуля "е-реєстратура" структуровано об’єднує всі компоненти для забезпечення ефективної та зручної роботи. Кожен процес має чітку послідовність, що дозволяє користувачам легко взаємодіяти із системою, а також забезпечує повний контроль за записами, доступом до даних та їх безпекою. Ця діаграма процесів підкреслює взаємодію різних компонентів, технологій та етапів, що обумовлюють стабільну і гнучку роботу "е-реєстратури" для медичних закладів.

КбПЗ_2024

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок–схем та опис алгоритмів функціонування модуля

Розглянемо для модуля "е-реєстратура" ключові етапи роботи: реєстрація нового пацієнта, пошук і редагування інформації пацієнта, запис до лікаря, інтеграція з іншими медичними модулями та обробка результатів. Опис алгоритмів забезпечить чітке розуміння взаємодії між компонентами модуля і його користувачами.

Блок-схема модуля "е-реєстратура":

1. Початок процесу

- Вхідні точки: реєстратор, пацієнт, лікар або адміністративний персонал.

2. Перевірка існування пацієнта в системі

- Пошук пацієнта : введення унікального ідентифікатора (номер картки пацієнта), імені, прізвища, дати народження чи інших даних. У разі збігу даних відбувається завантаження картки.

- Новий пацієнт : якщо дані пацієнта не знайдені, система пропонує створити новий запис.

КПЗ-2024

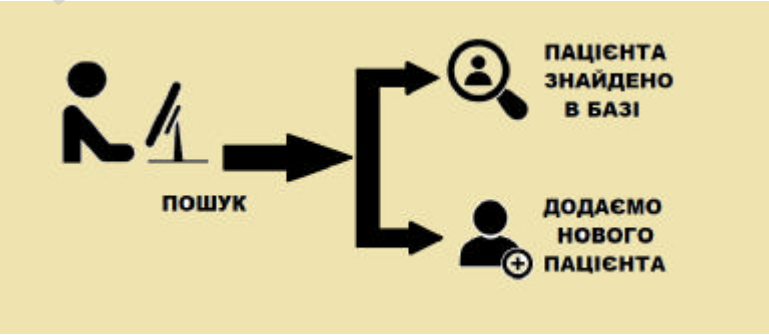


Рисунок 4.1 - Алгоритм роботи з пацієнтом

3. Додавання або оновлення інформації пацієнта :

- Новий запис : введення персональних даних (ім’я, прізвище, контактна інформація, дата народження тощо).

- Оновлення : внесення змін в існуючі записи (адреса, контактні дані тощо), за необхідності.

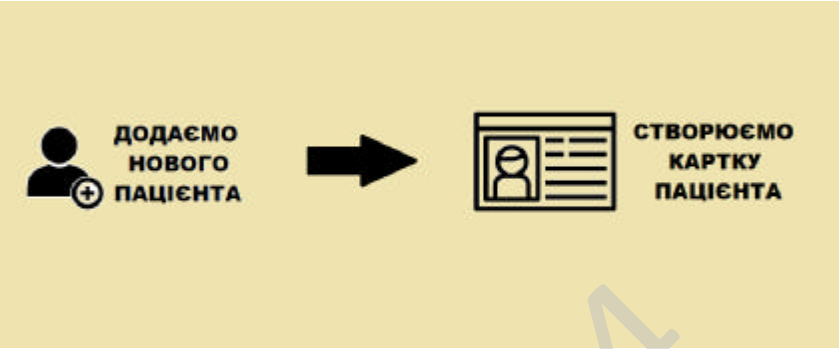


Рисунок 4.2 - Створення картки пацієнта

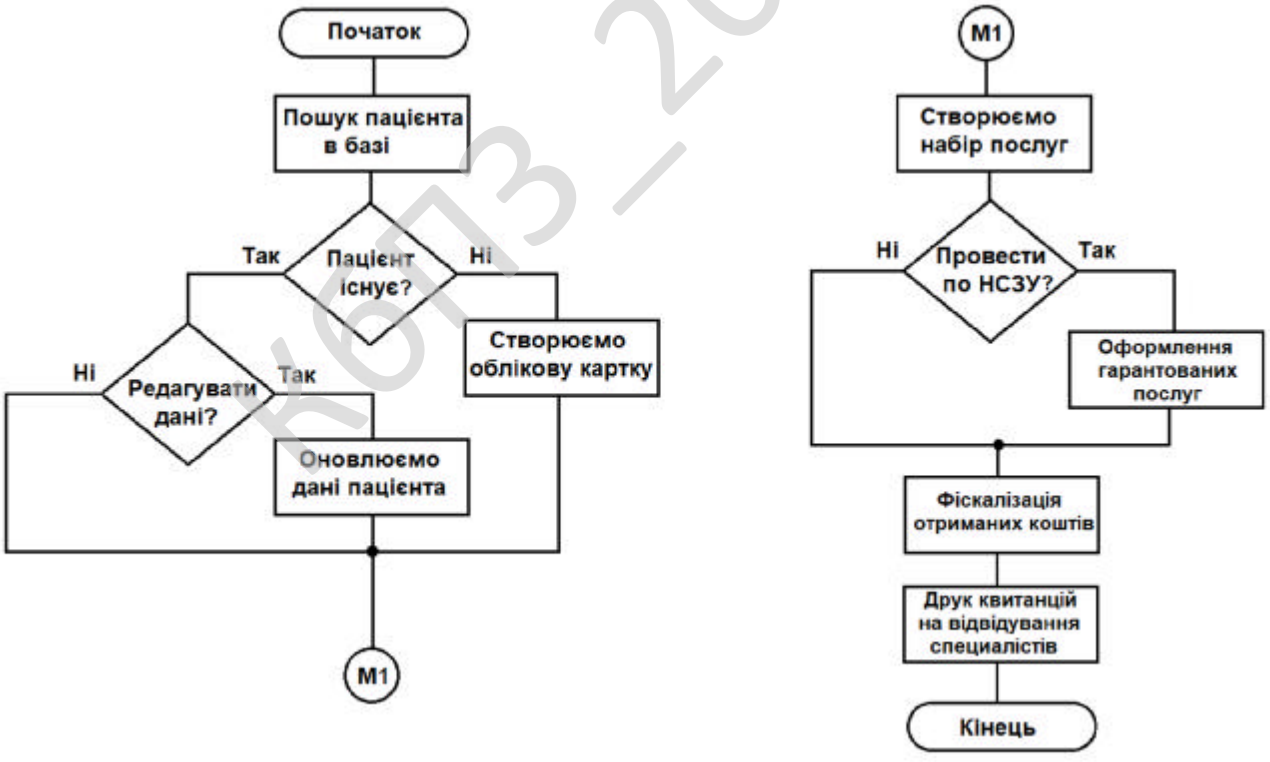


Рисунок 4.3-Блок-схема основної програми

КОПЗ_2024

4. Запис до лікаря

- Вибір лікаря та послуги: відображення списку доступних лікарів, спеціальностей і часу.

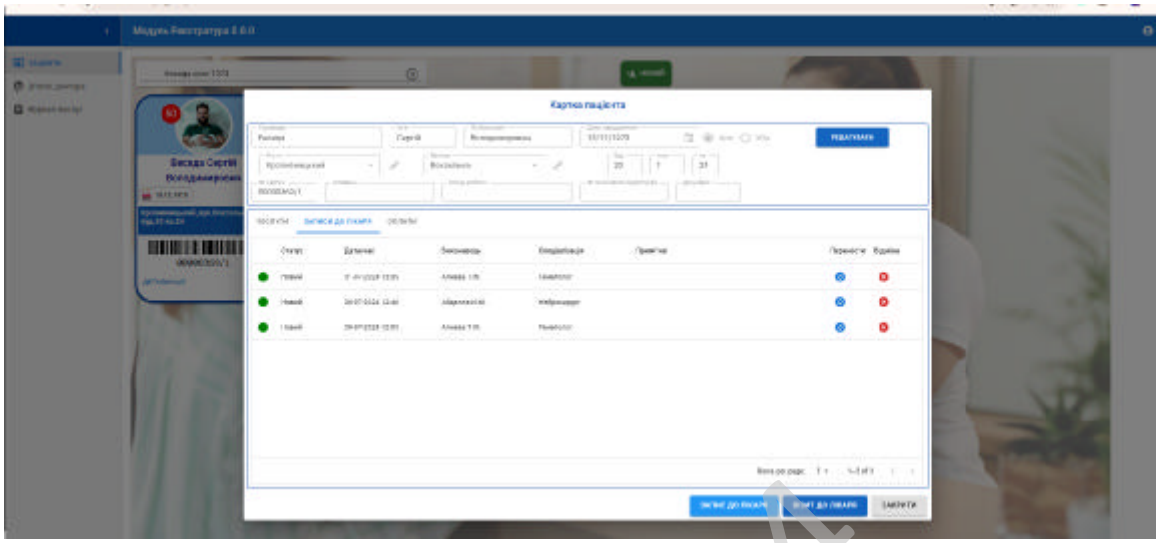


Рисунок 4.4-Запис на прийом до спеціаліста

- Бронювання часу : після вибору лікаря та часу запис створюється, інформація зберігається, а пацієнту видається роздрукований талон (за бажанням).

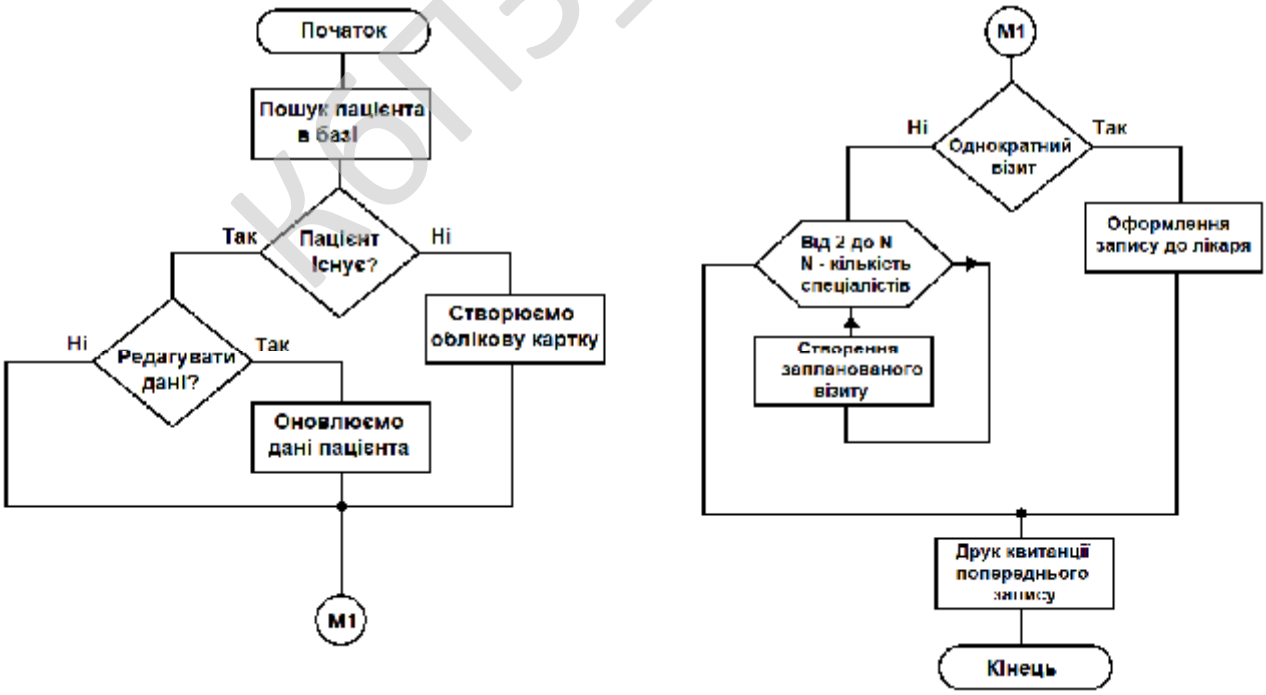


Рисунок 4.5-Блок-схема роботи підпрограми

КПІЗ_2024

5. Інтеграція з іншими модулями: Передача даних з модуля реєстратури в інші модулі (наприклад, лабораторії чи відділення стаціонарного лікування) для полегшення взаємодії між відділами.

6. Завершення процесу та збереження даних :

– Перевірка правильності внесених даних і їх збереження в базі даних;

– Опис алгоритмів функціонування модуля "е-реєстратура";

Алгоритм пошуку пацієнта:

1. Вхідні дані : ім’я, прізвище або дата народження.

2. Процес :

- Система виконує SQL-запит до бази даних для пошуку даних пацієнта.

- Якщо знайдено відповідність, інформація пацієнта завантажується.

3. Вихід : Відображення детальної інформації або повідомлення про відсутність даних.

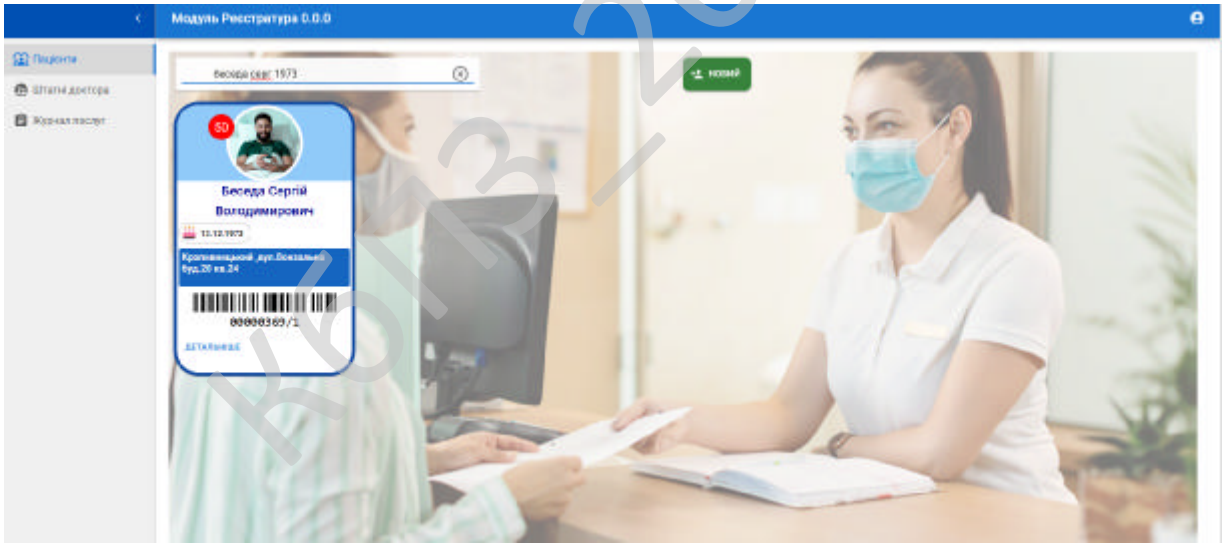


Рисунок 4.6-Пошук облікової картки пацієнта

Використання фреймворку Hibernate та JPA забезпечує зручну роботу з базою даних, дозволяючи ефективно використовувати запити до PostgreSQL.

Алгоритм реєстрації нового пацієнта:

1. Вхідні дані : персональні дані (ім’я, прізвище, дата народження, контактна інформація тощо).

КОПІЯ 2024

2. Процес :

- Реєстратор заповнює поля з інформацією про пацієнта у формі.

- Дані форми надсилаються на backend через REST API, де Spring обробляє запит.

- Система перевіряє, чи дані пацієнта вже існують (дублікати).

- Після перевірки пацієнт додається до бази даних.

3. Вихід : Повідомлення про успішну реєстрацію нового пацієнта можливістю роздрукувати картку.

Алгоритм оновлення інформації пацієнта:

1. Вхідні дані : редаговані дані пацієнта.

2. Процес :

- Вибирається пацієнт із бази даних, завантажується його інформація.

- Реєстратор вносить зміни в потрібні поля, наприклад, оновлює контактні дані.

- Після збереження інформація відправляється на сервер.

3. Вихід : оновлені дані зберігаються в базі даних.

Алгоритм запису до лікаря:

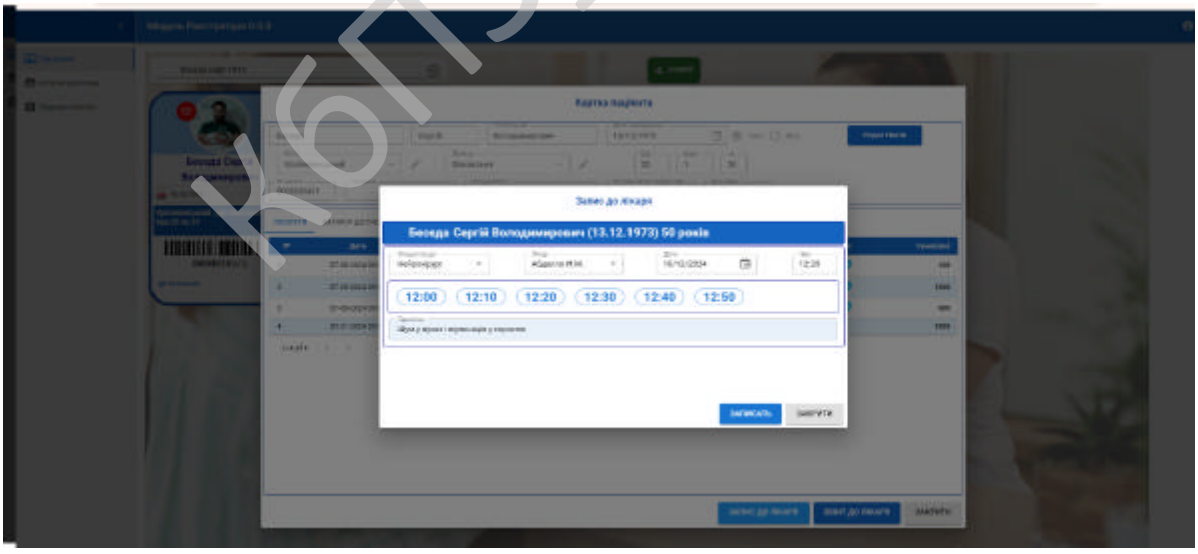


Рисунок 4.7- Форма запису пацієнта до лікаря

1. Вхідні дані : вибір лікаря, спеціальності та зручного часу.

2. Процес :

КБПЗ_2024

- Фронтенд у React взаємодіє з календарем MUI, де відображається доступний час для лікарів.

- Після вибору системою перевіряється доступність часу.

- Інформація запису надсилається на backend, зберігається у базі даних і резервується.

3. Вихід : підтвердження запису за допомогою нотифікації.

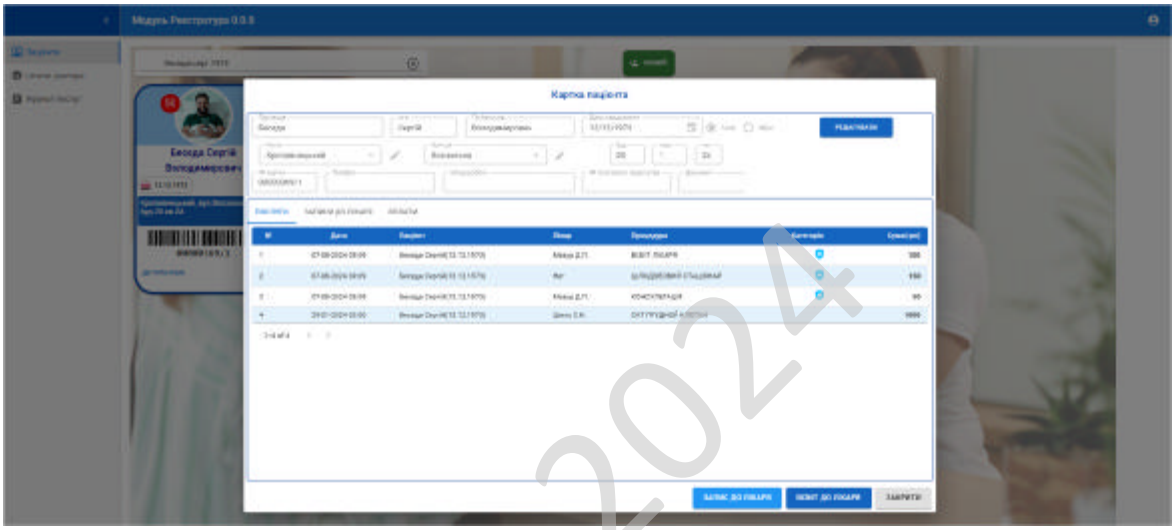


Рисунок 4.8-Історія отриманних послуг пацієнтом

КПЗ_2024

Алгоритм інтеграції з іншими модулями:

1. Вхідні дані : інформація про пацієнта для передачі в інші модулі (наприклад, лабораторія).

2. Процес :

- Через API дані передаються в інші сервіси, де вони використовуються для подальших процедур.

3. Вихід : інформація про пацієнта передана у потрібний модуль для обробки.

Усі дії завершуються підтвердженням, дані зберігаються у базі PostgreSQL, і модуль переходить у готовність для наступного запиту.

4.2 Захисту доступу до ПЗ та обміну даними за допомогою бібліотеки

JWT (JSON Web Token) [12] є відкритим стандартом (RFC 7519), який використовується для обміну інформацією між сторонами у вигляді токенів. Ці токени є компактними, безпечними та самодостатніми для передачі даних про автентифікацію чи авторизацію. JWT часто застосовується для захисту веб-додатків і API, забезпечуючи механізм перевірки користувачів і передачі безпечної інформації.

JWT складається з трьох частин[12]:

Header: містить тип токена (JWT) і алгоритм підпису (наприклад, HMAC або RSA).

Payload: містить дані (claims), які можуть бути зареєстрованими (наприклад, iss, sub, exp), публічними чи приватними.

Signature: використовується для перевірки цілісності даних і гарантує, що інформація не була змінена після створення.

JWT має наступний формат:

header.payload.signature, де кожна частина закодована у форматі Base64URL.

Принцип роботи JWT[12].

Користувач автентифікується:

При вході користувач надає свої облікові дані (логін і пароль), які перевіряються сервером.

Сервер генерує JWT:

Якщо автентифікація успішна, сервер створює токен, що містить інформацію про користувача та термін дії. Наприклад:

{ "sub": "1234567890",

"name": "user_kasa",

"admin": true,

"iat": 1516239022,

КОПЗ - 2024

"exp": 1516242622 }

Токен підписується секретним ключем (симетричний алгоритм) або приватним ключем (асиметричний алгоритм).

Клієнт отримує токен:

Згенерований токен передається клієнту, зазвичай через HTTP-відповідь. Клієнт зберігає його (наприклад, у localStorage, sessionStorage або як HTTP-заголовок).

Передача запитів із токеном:

Клієнт використовує токен для доступу до захищених ресурсів. Токен додається в заголовок кожного запиту:

Authorization: Bearer <JWT>

Валідація на сервері:

Сервер отримує запит із токеном, перевіряє його підпис і термін дії. Якщо токен валідний, сервер обробляє запит.

Захищений обмін:

Усі наступні запити відбуваються із JWT, забезпечуючи безперервний доступ до ресурсів без повторної автентифікації.

Схема роботи JWT

Користувач -> Сервер (Запит на автентифікацію):

Користувач вводить свої облікові дані.

Сервер перевіряє їх і генерує JWT.

Сервер -> Користувач (Відповідь із JWT):

Сервер передає JWT клієнту.

Користувач -> Сервер (Запит із JWT):

Користувач надсилає запит із JWT у заголовку.

Сервер перевіряє JWT:

Сервер перевіряє підпис і термін дії токена. Якщо валідний, надає доступ до запитуваних ресурсів.

КБПЗ-2024

Рисунок 4.9- Структура JWT[12]

Створення токена:

Header:

{ "alg": "HS256",

"typ": "JWT" }

Payload:

Містить дані про користувача та метадані. Наприклад:

{ "sub": "1234567890",

"name": "user_kasir",

"admin": true,

"iat": 1516239022,

"exp": 1516242622 }

Signature :

Генерується шляхом шифрування заголовка та тіла токена:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

КбПЗ_2024

Перевірка токена:

Сервер отримує JWT і розділяє його на три частини: header, payload, signature.

Перевіряється, чи відповідає підпис розрахованій сигнатурі.

Якщо токен прострочений (exp), сервер відхиляє запит.

Безпека обміну:

JWT ніколи не повинен передаватися без HTTPS, щоб уникнути перехоплення токена.

Зберігання токена повинно бути безпечним, наприклад, із використанням HTTP для запобігання XSS-атак.

Переваги використання JWT.

Автономність токена:

JWT містить всю необхідну інформацію, тому серверу не потрібно зберігати стан сеансу.

Гнучкість:

JWT можна використовувати для автентифікації користувачів, авторизації API-запитів та передачі метаданих.

Масштабованість:

Відсутність стану (stateless) на сервері робить систему масштабованою, оскільки кожен сервер може перевіряти токени незалежно.

Простота інтеграції:

JWT легко інтегрувати з існуючими системами.

Виклики використання JWT.

Безпека секретного ключа:

У разі компрометації секретного ключа весь механізм безпеки стає вразливим.

Розмір токена:

Через додавання даних токен може бути великим, що впливає на продуктивність у системах із великою кількістю запитів.

КБПЗ_2024

Відкликання токена:

JWT не підтримує механізм скасування (revoke).

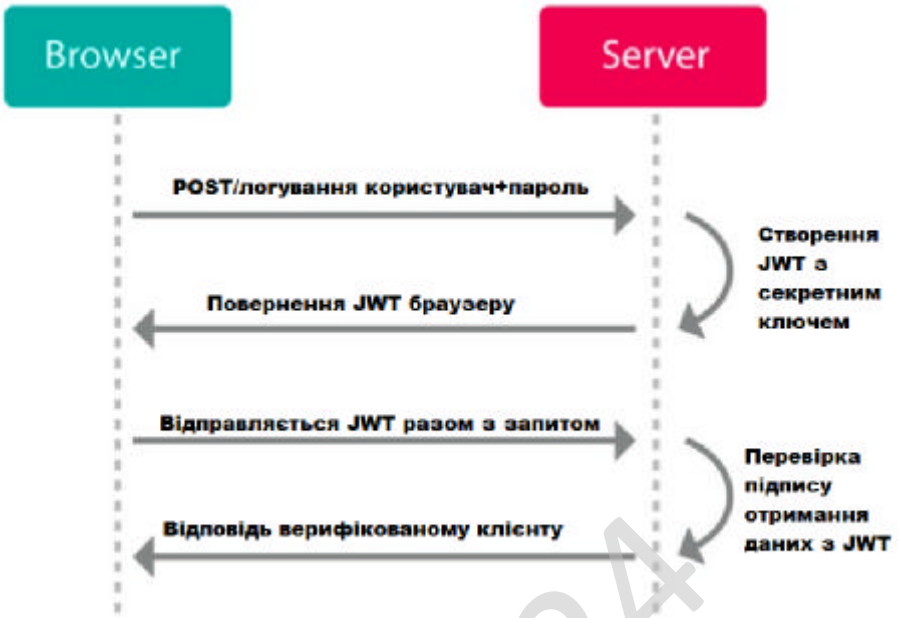


Рисунок 4.10- Схема роботи JWT (графічно)

Кроки:

Логін: Клієнт вводить облікові дані → запит до сервера → сервер генерує JWT → відповідь із токеном.

Запити з токеном: Клієнт додає JWT до заголовка → запит до сервера → сервер перевіряє токен → повертає ресурс.

Оновлення токена: Якщо токен скоро прострочиться, сервер видає новий токен на основі рефреш-токена.

КОПЗ - 2024

Рисунок 4.11- Реєстрація користувача у системі

Використання JWT у реальних сценаріях:

1. Мобільні додатки: Безпечна автентифікація користувачів і обмін даними.
2. SPA (Single Page Applications):Реалізація автентифікації без необхідності зберігати стан на сервері.
3. API-захист: Використання токенів для перевірки доступу до REST API.
4. Мікросервіси: JWT дозволяє передавати дані автентифікації між мікросервісами.

КбПЗ_2024

5 ВПРОВАДЖЕННЯ В ЕКСПЛУАТАЦІЮ

У цьому розділі запропоновано основні етапи, необхідні для належного запуску та інтеграції моду ля е-реєстратури в існуючий робочий процес медичної установи.

Перший етап це підготовка до впровадження.

Перед запуском модуля е-реєстратури необхідно провести кілька ключових підготовчих заходів:

– Аналіз поточних процесів : Документування існуючих процесів реєстрації пацієнтів, визначення найважливіших етапів і слабких місць у поточній системі. Це допоможе оцінити, які саме процеси можна вдосконалити за допомогою е-реєстратури.

– Визначення технічних вимог : Підтвердження апаратних і програмних вимог для безперебійного функціонування модуля, включаючи сервери, бази даних, мережеві налаштування та безпеку.

– Планування інфраструктури : Підготовка серверів для бекенду (Spring Boot з Java), налаштування бази даних PostgreSQL і серверних компонентів для фронтенду (React з MUI). Важливою частиною є також інтеграція з іншими медичними системами, якщо вони вже впроваджені в установі.

– Розробка плану впровадження : Визначення ключових етапів впровадження, розробка покрокового плану з визначенням термінів, відповідальних осіб і ресурсів.

КОПЗ-2024

Рисунок 5.1- Суцільний комплекс модулів лікувально-діагностичного центру

Наступний етап - тестування модуля е-реєстратури.

– Функціональне тестування : Перевірка роботи ключових функцій, таких як реєстрація нових пацієнтів, запис на прийом до лікаря, внесення змін до інформації про пацієнтів, пошук пацієнтів за кількома параметрами (ПІБ, дата народження, ID), а також забезпечення швидкого та коректного доступу до даних у базі.

– Інтеграційне тестування : Якщо е-реєстратура інтегрується з іншими модулями (наприклад, лабораторією, поліклінікою), слід перевірити коректну роботу взаємодії між ними. Інтеграційні тести гарантують, що дані з модуля реєстратури коректно передаються та отримуються іншими модулями.

– Безпека даних : Тестування на предмет захисту конфіденційної інформації пацієнтів. Перевірка безпеки автентифікації та авторизації користувачів, шифрування даних у процесі передачі й зберігання, а також захист від несанкціонованого доступу.

– Тестування навантаження : Симуляція різного навантаження на систему для оцінки її стійкості при одночасному доступі великої кількості користувачів. Це допоможе виявити потенційні вузькі місця та оптимізувати роботу бази даних PostgreSQL і REST API на Spring Boot.

КПЗ_2024

– Тестування продуктивності : Оцінка швидкості роботи ключових функцій і часу відповіді на запити. Низька затримка та висока продуктивність важливі, особливо в години пік, коли кількість запитів зростає.

Необхідною складовою процеса має бути навчання персоналу.

Навчання користувачів є важливим етапом для успішного впровадження модуля е-реєстратури:

– Підготовка інструкцій : Документація, яка містить покрокові інструкції для виконання всіх основних функцій, включаючи реєстрацію, пошук пацієнтів, запис на прийом тощо.

– Навчальні сесії : Проведення практичних навчань для всіх користувачів системи — адміністративного персоналу, медичних реєстраторів і лікарів. Під час навчання персонал отримує практичні навички роботи з системою, а також може задати питання.

– Підтримка та наставництво : Забезпечення можливості консультування протягом перших кількох тижнів після запуску, щоб персонал міг отримати оперативну допомогу в разі виникнення труднощів.

Перехід до експлуатації є важливим і відповідальним кроком до впровадження модуля в роботу. І він має пройти наступні етапи:

– Пілотний запуск : Рекомендовано провести пілотний запуск системи, почавши з одного або декількох відділень. Це дозволить виявити потенційні проблеми перед повним впровадженням. На цьому етапі персонал може відпрацювати всі необхідні процедури й надати зворотний зв’язок.

– Збір відгуків : Після пілотного запуску слід зібрати відгуки від персоналу та внести необхідні зміни, які допоможуть покращити роботу модуля. Це може стосуватися як інтерфейсу, так і функціональних аспектів.

– Оцінка готовності до повного запуску : Після успішного завершення пілотного запуску та внесення виправлень проводиться остаточна оцінка готовності до впровадження в усіх відділеннях.

КОПЗ-2024

– Повний запуск : Після затвердження модуль запускається у всіх відповідних підрозділах медичної установи, де він буде використовуватися на щоденній основі.

Завершальним і довготривалим етапом проекту є підтримка після запуску. Складові цього процесу доволі важливі:

– Моніторинг роботи : Налагодження системи моніторингу для відстеження ключових показників, таких як швидкість відповіді, помилки, якість доступу до даних і навантаження на систему. Це допоможе забезпечити стабільність роботи.

– Резервне копіювання даних : Регулярне резервне копіювання бази даних є критично важливим для забезпечення безперервності роботи. Це гарантує можливість відновлення даних у разі збою.

– Технічна підтримка : Забезпечення наявності технічної підтримки для допомоги користувачам у разі виникнення проблем і для оперативного усунення будь-яких неполадок у системі.

– Оновлення та вдосконалення : Регулярне оновлення модуля, що включає виправлення помилок, оновлення бібліотек, які використовуються, та впровадження нових функцій. Це дозволить модулю е-реєстратури залишатися актуальним і відповідати потребам установи.

Впровадження модуля е-реєстратури є важливим етапом у цифровій трансформації медичної установи. При успішному виконанні всіх етапів, від планування до підтримки після запуску, медична клініка зможе автоматизувати значну частину процесів реєстрації пацієнтів, покращити обслуговування пацієнтів і підвищити ефективність роботи персоналу. Це впровадження також підготує клініку до подальшої інтеграції з іншими модулями та розвитком єдиної медичної системи на основі мікросервісної архітектури.

КОПЗ-2024

6 НАУКОВА НОВИЗНА

Наукова новизна результатів цієї праці полягає в:

1) розроблено модель компонент діджиталізованих бізнеспроцесів лікувально-діагностичного центру (медичного закладу) [51];

2) набув подальшого розвитку метод пошуку записів у базі даних пацієнтів з використанням біометрії, який на відміну від існуючих (традиційних) забезпечує підвищення швидкості цифрової ідентифікації пацієнтів лікувально-діагностичного центру, а також точності оброблення даних в інформаційній системі медичного закладу за рахунок зменшення ймовірності помилкового модифікування медичних даних.

Запровадження електронної реєстрації пацієнтів у медичних закладах з використанням інноваційних технологій, таких як ідентифікація за штрих-кодом або відбитком пальця, має значний потенціал для покращення управління медичними даними та обслуговування пацієнтів. Сучасна електронна реєстратура може вирішувати одразу кілька важливих завдань: автоматизацію ідентифікації відвідувачів, прискорення доступу до даних, підвищення безпеки медичних записів та мінімізацію помилок. Застосування біометричних методів та штрих-кодів у системах реєстрації та пошуку надає можливість отримати нові наукові результати, важливі для розвитку інформатизації охорони здоров'я.

Автоматизація процесів ідентифікації та реєстрації

Традиційні системи реєстрації пацієнтів вимагають введення даних вручну, що може призводити до помилок, втрат часу та знижувати ефективність. Використання сучасних технологій для ідентифікації пацієнтів значно підвищує точність і швидкість обробки даних. Наприклад, штрих-коди дозволяють зберігати унікальний ідентифікаційний номер пацієнта, який може бути миттєво відсканований на прийомі, в реєстратурі або в будь-якому іншому підрозділі медичного закладу. Аналогічно, використання відбитків пальців як засобу

КБПЗ_2024

ідентифікації забезпечує швидкий і безпомилковий доступ до медичних записів, автоматизуючи процедури пошуку та реєстрації пацієнта.

Дослідження показують, що автоматизація ідентифікації та реєстрації, зокрема із використанням біометричних технологій, може зменшити час, витрачений на реєстрацію, на 30-40%, підвищити точність даних та мінімізувати ризик неправильної ідентифікації. Це наукове досягнення уможливлює швидкий облік, економію ресурсів та покращує управління даними в реальному часі, що є важливим для ефективного обслуговування пацієнтів.

Підвищення рівня безпеки та захисту персональних даних

Традиційні методи зберігання даних на папері або навіть електронні бази, що потребують звичайної аутентифікації (логін і пароль), стають недостатніми для забезпечення належного рівня захисту інформації. Впровадження електронної реєстратури з використанням біометричних даних, як-от відбитків пальців, підвищує рівень захисту персональних даних. Біометрія гарантує, що доступ до даних можуть отримати лише авторизовані особи, оскільки особисті характеристики (наприклад, відбиток пальця) не можуть бути легко відтворені або передані. Це також підвищує безпеку даних, оскільки біометричні параметри забезпечують захист від крадіжок та маніпуляцій.

Штрих-коди також можуть забезпечити рівень безпеки, що покращує традиційні методи доступу. Це знижує ризик несанкціонованого доступу і робить систему більш надійною у питаннях збереження конфіденційності.

Прискорення процесу доступу до медичної інформації

Застосування електронної реєстратури з технологіями штрих-кодів і біометрії дозволяє лікарям, медсестрам та іншому персоналу медичних закладів швидше отримувати доступ до медичних записів пацієнтів. Наприклад, пацієнт, який приходить на прийом, може ідентифікувати себе за допомогою відбитка пальця або штрих-коду на картці, що автоматично відкриває відповідні дані про його медичну історію, останні результати обстежень, поточне лікування тощо. Це

КПЗ-2024

виключає необхідність пошуку записів вручну та значно скорочує час, який пацієнт витрачає на очікування.

Це є науковим досягненням, оскільки автоматизація доступу до медичних даних забезпечує швидший прийом, підвищуючи ефективність роботи персоналу, і знижує ризик помилок, пов'язаних з ручним введенням інформації. Оперативний доступ до даних є особливо важливим у випадках надання невідкладної допомоги, коли кожна секунда має значення.

Підвищення точності та зменшення ризиків помилок

Одним із основних ризиків при роботі з пацієнтами є людський фактор, який може призводити до помилок у записах, пошуку або ідентифікації. Використання штрих-кодів та біометрії мінімізує ці ризики, забезпечуючи точну ідентифікацію пацієнта без необхідності ручного введення даних. Наприклад, штрих-код можна роздруковувати на медичних документах, аналізах та призначеннях, що дозволяє точно ідентифікувати їх для кожного пацієнта та уникнути помилок, що виникають у процесі лікування або лабораторних досліджень. Ця точність стає важливим фактором підвищення безпеки пацієнтів, адже правильне лікування залежить від точності даних. Знижуючи ризик помилок, медичні заклади можуть не лише забезпечити якіснішу допомогу, але й підвищити довіру пацієнтів до системи.

Розширення можливостей аналітики та прогнозування

Електронна реєстратура з інтегрованими штрих-кодами та біометричними даними надає унікальні можливості для збору й аналізу медичних даних. Зокрема, кожен запис, відвідування, призначення може бути зібрано та систематизовано, що дає можливість проводити аналітику для покращення роботи клініки. Наприклад, можна відстежувати статистику відвідувань за період, виявляти найпоширеніші захворювання, аналізувати ефективність лікування тощо. Це дозволяє медичному закладу ухвалювати обґрунтовані рішення щодо покращення якості послуг, підбору оптимальних методів лікування та покращення управління ресурсами.

КПЗ-2024

Біометричні дані та штрих-коди дозволяють збирати анонімізовану інформацію для досліджень і розробок, що може бути корисним для розробки нових методів лікування, прогнозування тенденцій здоров'я населення тощо. Цей аспект надає електронній реєстратурі не лише значення адміністративного інструменту, а й наукової платформи для подальших досліджень у галузі медицини.

Інтеграція з мобільними пристроями та віддаленим доступом

Сучасні технології ідентифікації, такі як штрих-коди у вигляді QR-кодів або інтеграція біометричних систем з мобільними додатками, дозволяють пацієнтам і лікарям зручно використовувати мобільні пристрої для доступу до медичної інформації. Наприклад, пацієнти можуть за допомогою мобільного додатку записатися на прийом, відсканувати свій штрих-код для перевірки інформації або навіть використовувати відбиток пальця для доступу до своєї електронної медичної картки. Це забезпечує зручність, спрощує доступ і підвищує комфорт використання системи для пацієнтів, що є важливим аспектом сучасного медичного обслуговування.

Таким чином, запровадження електронної реєстратури з використанням штрих-кодів та біометрії розширює функціональні можливості медичних ІС, покращує якість обслуговування, підвищує безпеку даних і знижує ризик помилок.

КбПЗ – 2024

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЕКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Електронний модуль реєстратури для медичних закладів є важливим компонентом сучасних інформаційних систем, який сприяє ефективній роботі закладу та поліпшенню обслуговування пацієнтів. Оскільки система розроблена з урахуванням специфічних потреб медичних установ, її цільова аудиторія включає як працівників медичних закладів, так і пацієнтів, що користуються медичними послугами. Визначення цієї аудиторії допоможе не лише оптимізувати роботу системи, але й підвищити її корисність і прийнятність для користувачів.

Медичні реєстратори є основними користувачами модуля електронної реєстратури. Їхня робота полягає у здійсненні первинної реєстрації пацієнтів, оформленні записів на прийом, веденні графіку візитів, відстеженні історії відвідувань. Їхнє завдання – швидке і точне введення інформації про пацієнта, щоб забезпечити лікарів необхідними даними для надання якісної медичної допомоги. Модуль електронної реєстратури дозволяє реєстраторам:

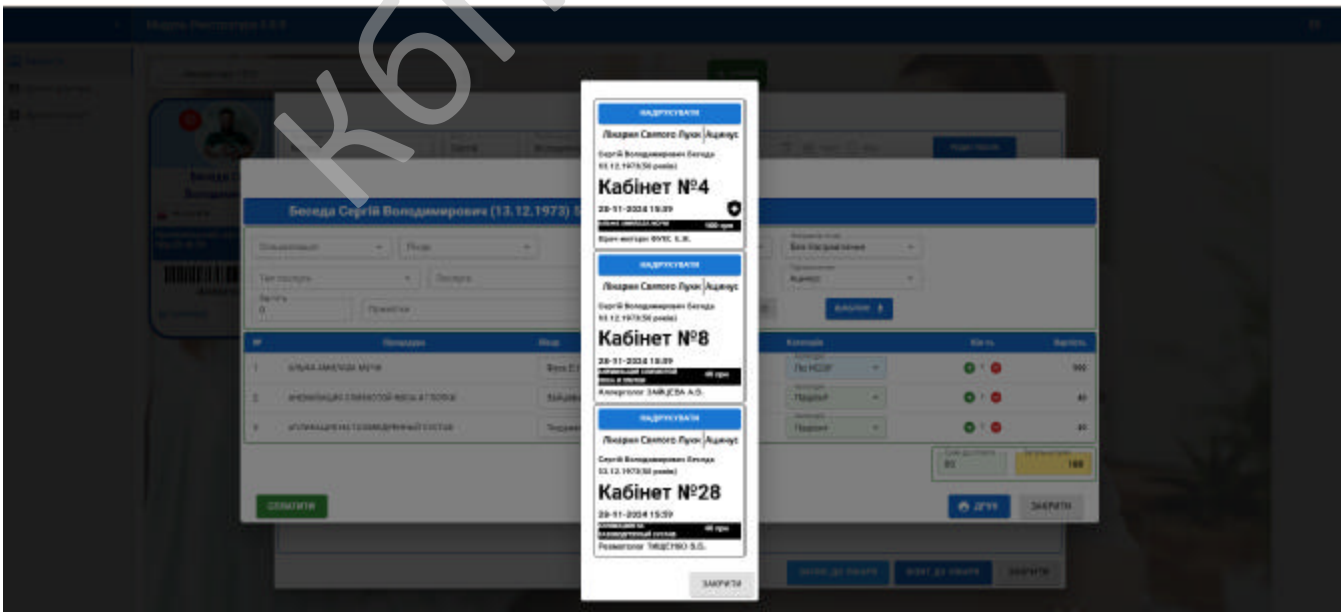


Рисунок 7.1- Талони на відвідування спеціаліста або отримання послуги

КПЗ – 2024

Зменшити час на обробку даних і запис пацієнтів.

Уникнути помилок, пов’язаних із ручним введенням даних або роботою з паперовими документами.

Забезпечити оперативне інформування пацієнтів про час прийому та зміни в розкладі.

Впровадити єдину електронну картотеку, яка полегшує доступ до потрібної інформації.

Другий за значимістю сегмент цільової аудиторії – це **лікарі та інші медичні працівники** (медсестри, лаборанти, фахівці з діагностики тощо), які використовують інформацію із системи для надання послуг пацієнтам. Модуль електронної реєстратури дозволяє їм:

1. Мати доступ до повної інформації про пацієнта, включаючи історію хвороби, проведені обстеження, аналізи, призначення лікарів.

2. Швидко переглядати графік прийому пацієнтів та координувати свої дії з медичними реєстраторами.

3. Зменшити витрати часу на отримання необхідної інформації, оскільки вся вона зберігається в цифровому вигляді.

4. Впроваджувати ефективну комунікацію між лікарями та іншими медичними працівниками.

Адміністративні співробітники або керівництво медичного закладу є ще однією важливою аудиторією модуля електронної реєстратури, оскільки вони відповідають за координацію роботи закладу, організацію та оптимізацію робочих процесів, забезпечення ресурсами і управління фінансами. Завдяки модулю електронної реєстратури вони можуть:

Оцінювати ефективність роботи реєстраторів, лікарів та інших працівників медичного закладу.

Планувати кількість і розподіл пацієнтів, що дозволяє оптимізувати потоки клієнтів у закладі.

КПЗ_2024

Відстежувати показники, такі як кількість прийнятих пацієнтів, частота звернень, розподіл пацієнтів за різними лікарями та відділеннями.

Збирати і аналізувати дані для стратегічного планування та прийняття рішень.

Пацієнти є кінцевими користувачами послуг, наданих за допомогою електронної реєстратури. Хоча вони безпосередньо не взаємодіють із системою так, як медичні працівники, вони отримують суттєві переваги від її використання. Основні потреби пацієнтів, які задовольняє система, включають:

Можливість швидкого запису на прийом через інтернет або мобільний додаток.

Отримання нагадувань про майбутній прийом або обстеження, що допомагає пацієнтам планувати візити.

Прозорість процесу запису, доступ до інформації про наявність місць і доступні години прийому.

Зменшення часу очікування у черзі завдяки більш точному плануванню роботи медичного персоналу.

У великих медичних закладах IT-фахівці часто відповідають за налаштування та підтримку систем, таких як електронна реєстратура. Для них система є важливим інструментом, оскільки вони забезпечують її безперебійне функціонування, налаштування та інтеграцію з іншими системами. IT-відділ є аудиторією, що забезпечує:

1. Встановлення та налаштування сервісів для інтеграції з іншими системами, такими як системи управління обліком чи лабораторії.

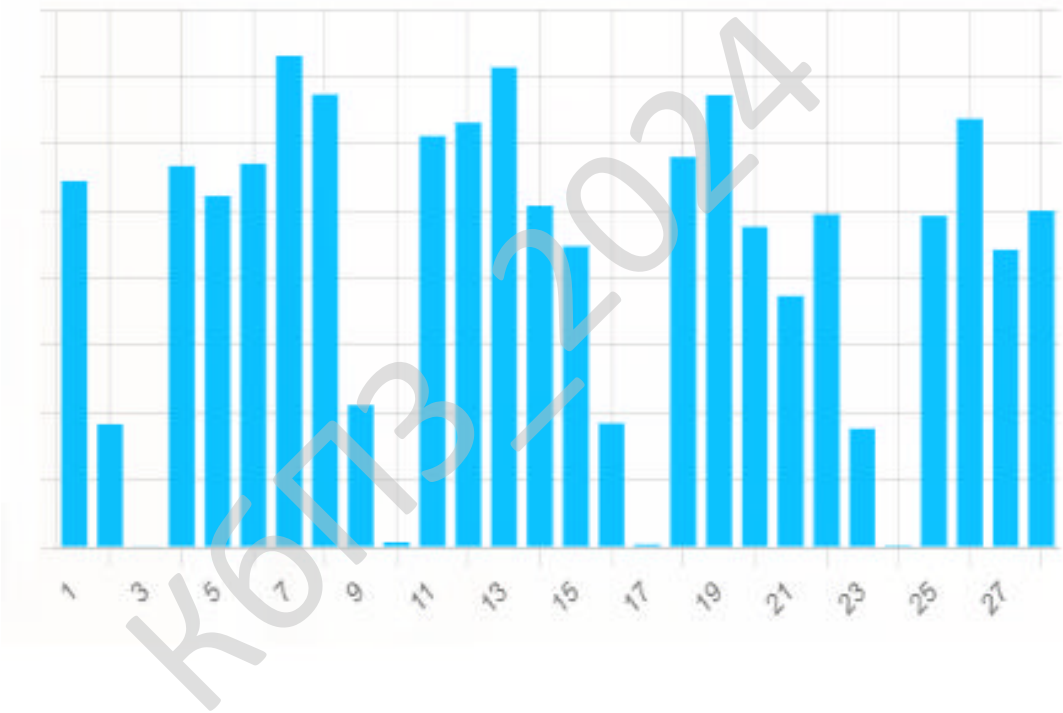
2. Здійснення технічного обслуговування та підтримки системи, гарантію її стабільності та надійності.

3. Впровадження заходів щодо забезпечення безпеки даних, таких як шифрування, багаторівнева автентифікація та резервне копіювання.

КПЗ_2024

Міністерства охорони здоров'я та регуляторні органи в багатьох країнах також можуть виступати як непряма аудиторія. Вони встановлюють вимоги до ведення обліку пацієнтів та захисту їхніх персональних даних. Система е-реєстратури повинна відповідати цим стандартам та регуляторним вимогам, що дозволить:

1. легше відповідати вимогам законодавства щодо захисту даних.
2. забезпечити прозорість та ефективність процесів у медичних закладах.
3. спрощувати подачу звітності медичними установами, що часто вимагається органами охорони здоров'я.
4. взаємодія між цільовими групами.



КбПЗ 2024

Рисунок 7.2- Кількість виконаних послуг медзакладу по дням

Важливо також врахувати взаємодію між цими цільовими групами, оскільки ефективне використання системи передбачає гармонійний зв'язок між ними.

Наприклад:

1. медичні реєстратори взаємодіють з пацієнтами для запису на прийом та отримання інформації, а також передають дані лікарям.

2. лікарі та реєстратори можуть координувати графіки, щоб пацієнти були записані на зручний час без накладок.

3. IT-відділ підтримує безперебійну роботу системи для всіх інших груп, що дозволяє зберегти дані, запобігти збоям та захистити конфіденційність особливо під час воєнного стану.

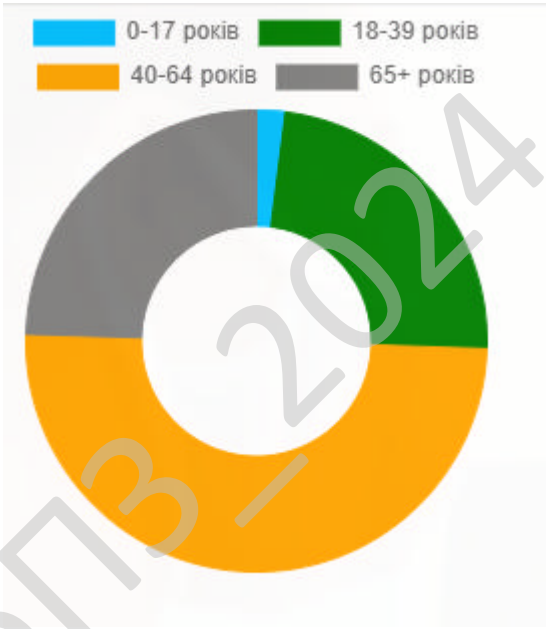


Рисунок 7.3- Розподіл пацієнтів по віковим групам

КОПЗ 2024

Роль технологічних компонентів у забезпеченні зручності для цільової аудиторії.

Сучасні технології, такі як інтеграція з базами даних, система онлайн-запису, віддалений доступ до записів, дозволяють реалізувати важливі функції для кожної з груп користувачів:

1. Медичні реєстратори використовують інтерфейс із можливістю швидкого пошуку, додавання та редагування записів про пацієнтів.

2. Лікарі отримують доступ до історії хвороби, що дозволяє їм швидко знайомитися з інформацією про пацієнтів та забезпечувати вищу якість медичних послуг.

3. Адміністрація може моніторити активність у системі, створювати звіти, а також планувати ресурси на основі аналітичних даних.

Отже, електронна реєстратура для медичних закладів охоплює широку аудиторію користувачів, кожна з яких має свої специфічні потреби. Розуміння цих потреб та належне врахування особливостей кожної групи дозволяє забезпечити оптимальне функціонування системи та підвищити її корисність.

7.2 Оцінка привабливості проекту шляхом застосування методів експертних оцінок

Успішність впровадження інноваційних ІТ-проектів, а також і таких як модуль е-реєстратури, залежить від їхньої привабливості для користувачів, керівників медичних закладів. Для більш об’єктивного визначення доцільності проекту ймовірність його успішної реалізації аналізується за допомогою методів експертних оцінок. Цей підхід дозволяє оцінити ефективність та значущість проекту на основі думок фахівців із відповідної сфери, таких як ІТ-розробники, медичні адміністратори, лікарі та економісти.

Використання методів експертних оцінок має певні переваги.

Експертні оцінки є ефективним інструментом для аналізу й оцінювання проектів, особливо коли мова йде про інноваційні розробки, де відсутні стандартизовані процедури оцінювання. Серед переваг методів експертних оцінок можна виділити:

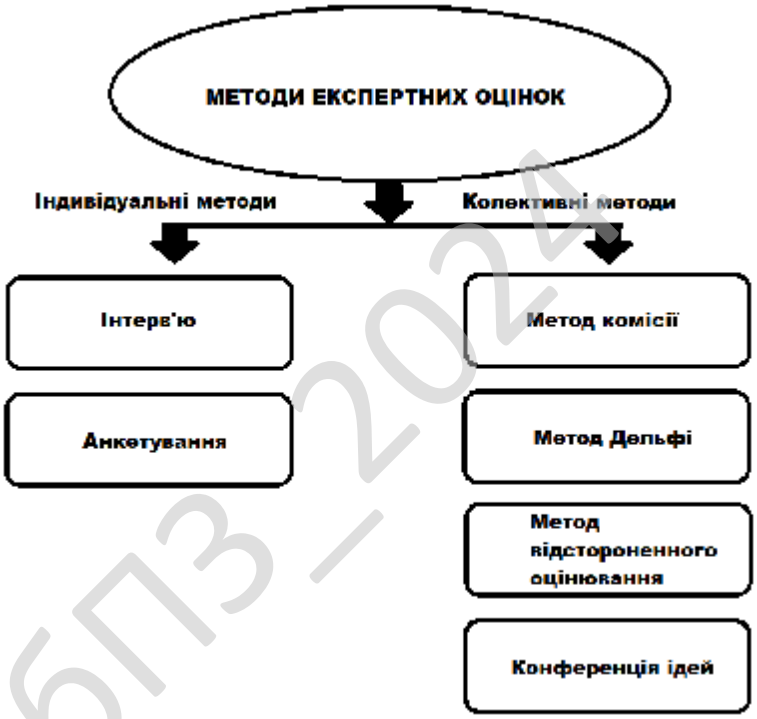
– гнучкість та адаптивність: експертні методи дозволяють адаптувати критерії оцінювання під конкретний проект.

КПЗ-2024

– урахування різних точок зору: залучення фахівців із різних сфер допомагає оцінити проект із точки зору технологічної реалізації, медичних стандартів та економічної ефективності.

– Швидкість отримання результатів: на відміну від тривалих аналітичних досліджень, метод експертних оцінок дозволяє оперативно визначити ключові характеристики та ризики проекту .

Розрізняють наступні методики збору експертних оцінок.



КБПЗ_2024

Рисунок 7.4- Основні види методів експертних оцінок

Методи збору експертних думок можуть різнитися в залежності від цілей і структури проекту . У випадку оцінки привабливості модуля е-реєстратури можна використовувати кілька основних підходів:

– колективні експертні методи – це методи, які забезпечують формування єдиної спільної думки в результаті взаємодії залучених фахівців-експертів.

– метод комісії полягає у вироблені експертами кращого варіанта досягнення поставленої мети з урахуванням усіх висловлених на нараді пропозицій, ідей.

– анкетування: Експерти відповідають на низку запитань, що стосуються різних аспектів проекту — технологічного, функціонального, фінансового. На основі відповідей розробляється середній бал оцінки.

– метод Дельфі: Цей метод передбачає декілька етапів опитувань з анонімними відповідями та поступовим корегуванням результатів, що дозволяє досягнути консенсусу серед експертів.

– конференція ідей або інтерв'ю з експертами: Дозволяє глибше вивчити думку експертів, обговорити деталі та варіанти розвитку проекту , а також виявити приховані ризики.

– метод парних порівнянь: Використовується для оцінки пріоритетності окремих характеристик проекту , таких як функціональні можливості, безпека, зручність використання та вартість впровадження.

Визначають ключові критерії оцінки проекту.

При оцінці привабливості модуля е-реєстратури експерти орієнтуються на такі основні критерії:

– технічна реалізація: Враховується рівень складності реалізації проекту , необхідність додаткових ресурсів, а також сумісність із сучасними IT-стандартами та системами.

– безпека даних: Одним із критично важливих аспектів для будь-якої медичної системи є захист даних пацієнтів, тому проект оцінюється за рівнем безпеки та конфіденційності.

– користувацька зручність: Проект оцінюється з точки зору зручності для різних користувачів (лікарі, пацієнти, адміністративний персонал), а також можливості швидкого навчання для нових користувачів.

КПЗ-2024

– економічна ефективність: Важливо оцінити очікуваний рівень економії ресурсів, скорочення витрат на обробку та зберігання паперової документації, а також можливість залучення додаткового фінансування

– відповідність законодавчим вимогам: Експерти перевіряють, чи відповідає розробка нормативним стандартам, наприклад, щодо зберігання персональних даних.

На основі отриманих оцінок формується звіт, який містить узагальнені висновки щодо привабливості проекту та його перспектив. Експертний аналіз також дозволяє корегувати деталі проекту, визначити слабкі місця, які потребують додаткового опрацювання.

7.3 Оцінка вартості розробки програмного забезпечення

Ефективна оцінка вартості розробки програмного забезпечення є ключовим етапом для розуміння економічної доцільності проекту. Вартість створення модуля "е-реєстратура" залежить від низки чинників, зокрема технічних вимог, кількості функціональних модулів, потреб у масштабуванні та сумісності з іншими системами. Для цього вибір методу оцінки вартості стає критично важливим рішенням, оскільки він допоможе уникнути непередбачуваних витрат та забезпечити реалізацію проекту в межах бюджету.

Сучасна практика програмної інженерії використовує кілька основних підходів до оцінки вартості розробки програмного забезпечення, кожен із яких має свої переваги та недоліки.

Метод оцінки за робочими годинами

Цей метод передбачає розрахунок вартості на основі оцінки робочих годин, необхідних для виконання завдань, і погодинної ставки розробників. Для модуля "е-реєстратура" цей підхід дозволяє детально врахувати час, витрачений на проектування, розробку, тестування та документування.

КПЗ-2024

Переваги: Метод забезпечує гнучкість у випадку, коли вимоги до проекту можуть змінюватися, або коли важко оцінити обсяг робіт на початку.

Недоліки: Може бути складно прогнозувати загальну вартість заздалегідь, оскільки вона залежить від реальних витрат часу, що робить метод не завжди зручним для фіксованих бюджетів.

Метод оцінки на основі функціональних точок

Цей метод враховує кількість функціональних компонентів, таких як введення даних, запити та генерація звітів, для оцінки вартості проекту. Застосування методу функціональних точок дозволяє врахувати рівень складності та обсяг функціональності модуля "е-реєстратура".

Переваги: Метод дозволяє обчислити приблизну вартість, незалежно від технологічних особливостей реалізації, завдяки чому є універсальним для різних типів програм.

Недоліки: Не враховує специфічні технічні ризики проекту або обмеження, такі як потреба в додатковій інтеграції з іншими системами.

Метод оцінки COCOMO (COnstructive COst MOdel)

COCOMO базується на емпіричних моделях і дозволяє оцінювати вартість проекту на основі попереднього аналізу його розмірів та складності. Цей підхід розраховує вартість, базуючись на кількості рядків коду або «доставлених функцій», і поділяє проект на три рівні: базовий, проміжний та детальний.

Переваги: Забезпечує об'єктивний та структурований підхід до розрахунку вартості, враховуючи різні сценарії масштабів розробки.

Недоліки: Потребує великої кількості даних для точної оцінки, а також знань щодо складності проекту.

Метод аналогій

Метод аналогій передбачає використання історичних даних подібних проектів для прогнозування вартості. У випадку модуля "е-реєстратура" можна оцінити його вартість, використовуючи дані про аналогічні модулі, реалізовані в медичній сфері, зокрема проекти реєстрації пацієнтів або ведення медичних записів.

КПЗ-2024

Переваги: Простота у використанні, особливо якщо є доступ до історичних даних проектів подібної складності.

Недоліки: Висока залежність від наявних даних; якщо їх недостатньо або проекти суттєво відрізняються, оцінка може бути неточною.

Метод оцінки на основі ризиків

Цей підхід враховує потенційні ризики проекту, наприклад, технічні труднощі, можливість затримок або проблеми з інтеграцією. Оцінка вартості здійснюється шляхом додавання резерву на покриття ризиків до базової вартості.

Переваги: Враховує потенційні непередбачувані витрати, що робить оцінку надійнішою.

Недоліки: Метод може збільшити загальний бюджет проекту, навіть якщо ризики не реалізуються.

З урахуванням складності та вимог до модуля "е-реєстратура", оптимальним вибором може бути комбінування методу за робочими годинами та функціональних точок. Поєднання цих двох методів дозволить оцінити як вартість реалізації конкретних функцій, так і загальну трудомісткість проекту.

Визначення функціональних вимог: Ідентифікувати всі функції модуля, включно з реєстрацією пацієнтів, записом до лікаря, перевіркою статусу пацієнта, інтеграцією з базою даних та налаштуванням прав доступу.

Розрахунок кількості функціональних точок: Визначити кількість точок для кожної функції на основі її складності (висока, середня, низька).

Оцінка робочих годин: Для кожної функціональної точки розрахувати орієнтовний обсяг робочих годин для розробки, тестування та інтеграції.

Розрахунок загальної вартості: Визначити загальну вартість на основі погодинної ставки розробників, множачи її на сумарну кількість робочих годин.

Аналіз ризиків та резерв: Додати резервний бюджет на покриття можливих непередбачуваних витрат, що можуть виникнути в процесі розробки.

КПЗ-2024

Обрані методи оцінки дозволять об’єктивно визначити загальну вартість проекту, а також створити резерв для покриття ризиків, які можуть вплинути на процес розробки.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ

Економічна ефективність будь-якого програмного рішення є одним з головних показників його привабливості для інвесторів та кінцевих користувачів. Модуль електронної реєстратури (е-реєстратура), призначений для оптимізації процесу обслуговування пацієнтів, підвищення якості обліку і взаємодії медичних підрозділів, може істотно скоротити затрати ресурсів та підвищити загальну продуктивність закладу. У цьому розділі розглянемо фактори економічної ефективності впровадження модуля е-реєстратури в медичному закладі, а також методи та показники, що дозволяють оцінити економічну віддачу.

Основні фактори економічної ефективності впровадження модуля е-реєстратури

Зниження затрат на адміністрування: Автоматизація значної частини процесів реєстрації пацієнтів зменшує кількість необхідних адміністративних ресурсів, скорочуючи робоче навантаження реєстраторів. Зменшення кількості операцій вручну призводить до економії трудових ресурсів та потенційного скорочення штату, що дозволяє зменшити витрати на заробітні плати та супутні адміністративні витрати.

Підвищення точності даних та зниження рівня помилок: Використання електронної системи знижує ризик помилок під час введення та обробки даних пацієнтів. Це дозволяє уникати витрат, пов'язаних із виправленням неточностей, а також підвищує якість обслуговування, оскільки точність інформації є вирішальним фактором у медичній сфері.

КПЗ_2024

Покращення доступності інформації: Впровадження електронної реєстратури дозволяє створити централізовану базу пацієнтів з миттєвим доступом до даних з будь-якого підрозділу. Це прискорює комунікацію між відділеннями та підвищує оперативність обслуговування пацієнтів, що позитивно впливає на фінансові показники.

Оптимізація використання ресурсів закладу: За рахунок швидшого обслуговування пацієнтів, автоматизації запису до лікарів і управління чергою, медичний заклад зможе більш ефективно розподіляти робочий час лікарів та іншого персоналу. У довгостроковій перспективі це підвищує продуктивність роботи, зменшує простої та покращує використання доступних медичних ресурсів.

Покращення рівня задоволеності пацієнтів: Електронна реєстратура забезпечує швидке та зручне обслуговування, що значно підвищує задоволеність пацієнтів. Це може позитивно позначитись на репутації закладу, а також сприяти залученню більшої кількості пацієнтів, що, в свою чергу, збільшить прибуток клініки.

Показники та методи оцінки економічної ефективності

Метод рентабельності інвестицій (ROI): Один з найбільш поширених показників економічної ефективності, який дозволяє оцінити прибуток від інвестицій у модуль е-реєстратури. ROI визначається як відношення чистого прибутку до інвестиційних витрат і розраховується за формулою:

$$ROI = \frac{\text{Чистий прибуток}}{\text{Вартість інвестицій}} * 100\%$$

У випадку модуля е-реєстратури чистий прибуток включає в себе зекономлені адміністративні витрати, скорочення затрат на обслуговування пацієнтів, а також потенційний приріст доходів від збільшення кількості пацієнтів.

КБПЗ – 2024

Метод розрахунку загальної вартості володіння (TCO): Показник, що включає всі витрати, пов'язані з впровадженням, підтримкою і експлуатацією модуля е-реєстратури. TCO дозволяє оцінити довгострокові витрати на систему, враховуючи як початкові капітальні інвестиції, так і операційні витрати на підтримку.

Метод чистої теперішньої вартості (NPV): Показник, який враховує майбутні грошові потоки, зважені на час, для визначення їх теперішньої вартості. Це особливо корисний метод, коли економічні вигоди від впровадження е-реєстратури очікуються у віддаленій перспективі. NPV розраховується за формулою:

$$NPV = \sum \frac{\text{Прогнозовані грошові потоки}}{(1 + \text{Дисконтна ставка})^t} - \text{Інвестиційні витрати}$$

Даний показник враховує інфляцію та дозволяє визначити, чи проект є економічно вигідним у порівнянні з альтернативними інвестиціями.

Очікувані результати від впровадження

Зниження витрат: Очікується, що модуль е-реєстратура зменшить операційні витрати на 20–30%, особливо завдяки оптимізації робочого часу персоналу, зниженню помилок у даних та покращенню загальної продуктивності. Це особливо актуально для клінік з високим потоком пацієнтів.

Збільшення доходів: Поліпшення обслуговування, зниження черг, ефективне використання ресурсів та зручний доступ до даних про пацієнтів сприяють зростанню задоволеності клієнтів, що може призвести до залучення нових пацієнтів, підвищуючи прибуток клініки на 10–15%.

Поліпшення продуктивності персоналу: Впровадження електронної реєстратури дозволяє медичному персоналу та адміністраторам зосередитись на якісному наданні послуг, замість виконання рутинних процесів, що додатково підвищує ефективність і задоволеність працівників.

КОПЗ_2024

Покращення якості обслуговування: Модуль забезпечує оперативний доступ до важливих даних, що дозволяє персоналу швидко знаходити необхідну інформацію та знижує час очікування пацієнтів. Це підвищує рівень довіри пацієнтів та загальне враження від закладу.

Економічна ефективність від впровадження модуля е-реєстратура реалізується завдяки зменшенню витрат на адміністрацію, підвищенню продуктивності персоналу, оптимізації використання ресурсів та покращенню якості обслуговування пацієнтів. Додавання таких функцій, як пошук за штрих-кодом або ідентифікація за біометричними даними, може додатково підвищити рівень автоматизації та уникнути помилок при ідентифікації пацієнтів, забезпечуючи довгострокові переваги. Загальна ефективність від впровадження модуля е-реєстратура надає вагомі аргументи для його впровадження та є фактором підвищення конкурентоспроможності медичного закладу.

7.5 Алгоритм просування проекту розробки модуля

Алгоритм просування проекту розробки модуля "е-реєстратура" має кілька ключових етапів, що допоможуть ефективно інтегрувати продукт на ринку та привернути увагу медичних закладів, інвесторів і користувачів. Ось детальний план просування:

1. Дослідження потреб медичних установ: Вивчення проблем і запитів медичних установ, зокрема клінік, поліклінік, діагностичних центрів. Це дозволить зрозуміти, які функції модуля є найбільш затребуваними, щоб підкреслити їх у просуванні.

2. Вивчення конкурентів: Аналіз існуючих рішень на ринку електронної реєстрації пацієнтів. Це допоможе сформувати унікальну торгову пропозицію (USP) і виокремити переваги над іншими програмними рішеннями.

3. Сегментування цільової аудиторії: Визначення основних категорій користувачів, таких як адміністративний персонал медичних закладів, лікарі, IT-

КбПЗ-2024

спеціалісти, менеджери, щоб адаптувати маркетингове повідомлення до потреб кожної групи.

4. Фокус на ефективності та зручності: Оскільки модуль "е-реєстратура" сприяє оптимізації реєстрації пацієнтів, зниженню навантаження на персонал і покращенню доступу до даних, це стане ключовою пропозицією для залучення уваги.

5. Переваги в захисті даних: Наголос на використанні сучасних технологій захисту інформації (шифрування, доступ за рівнями) стане вагомим фактором для медичних установ, оскільки безпека даних є критично важливою.

6. Інноваційні функції: Наприклад, підтримка технологій біометрії чи штрих-кодування для ідентифікації пацієнтів, що зменшить час на пошук картки пацієнта та мінімізує помилки.

7. Створення сайту та лендінг-сторінки: Лендінг-сторінка з ключовими перевагами модуля, демонстраційними відео, можливістю запису на презентацію або вебінар. Важливо передбачити форму зворотного зв’язку для швидкого зв’язку із зацікавленими клієнтами.

8. Цільовий контент-маркетинг: Публікація блогів, статей та новин на медичну тематику, де підкреслюються переваги автоматизації реєстраційних процесів. Просування через соціальні мережі: Використання LinkedIn, Facebook та інших платформ для таргетованої реклами на медичний персонал, керівників і ІТ-спеціалістів у медичній сфері. Також важливо публікувати корисний контент, інфографіку та демонстрації інтерфейсу.

9. Створення демонстраційної версії модуля: Демо-версія з обмеженим функціоналом дає можливість медичним установам ознайомитися з основними можливостями системи. Це дозволяє потенційним клієнтам відчути зручність та корисність програми.

10. Проведення вебінарів і презентацій: Запуск серії онлайн-зустрічей для демонстрації продукту. Такі заходи допоможуть зібрати відгуки від аудиторії та скоригувати маркетингову стратегію.

КОПЗ_2024

11. Співпраця з постачальниками медичного обладнання та IT-рішень: Партнерство з компаніями, які вже обслуговують медичні установи, може відкрити доступ до ширшої аудиторії.

12. Пошук державних або грантових програм: Участь у програмах підтримки інновацій в охороні здоров’я. Це може допомогти отримати фінансування для подальшого розвитку продукту або підтримки на початкових етапах впровадження.

13. Тестування продукту в кількох медичних установах: Проведення пілотних запусків для оцінки практичного використання модуля, виявлення можливих проблем і вдосконалення на основі зворотного зв’язку.

14. Внесення змін на основі відгуків: Після пілотного тестування потрібно провести оновлення, оптимізацію та вдосконалення функцій відповідно до реальних потреб установ.

15. Розробка інструкцій і навчальних матеріалів: Доступні навчальні матеріали для персоналу, включаючи відеоуроки, документацію та FAQ.

16. Технічна підтримка: Наявність гарячої лінії та підтримки для клієнтів є критичною. Це підвищує довіру та лояльність до продукту.

17. Постійний моніторинг та оновлення: Регулярне оновлення модуля на основі нових потреб користувачів і вдосконалень у галузі охорони здоров’я.

18. Аналіз використання функцій: Регулярне отримання зворотного зв'язку від користувачів модуля для моніторингу найбільш корисних і затребуваних функцій.

19. Оцінка економічного ефекту: Проведення досліджень для визначення ефективності модуля з точки зору економії часу і ресурсів, зменшення паперової роботи та підвищення швидкості обслуговування пацієнтів.

20. Побудова аналітичної звітності для управлінців: Формування звітів щодо основних метрик використання системи, які дозволяють визначити рівень економічної ефективності та допомогти в подальшому розвитку проекту.

КБПЗ_2024

21. Оптимізація процесу впровадження для масштабування: Налагодження процесів підтримки, адаптації та навчання користувачів, щоб зробити модуль зручним для масштабного впровадження.

22. План розширення: Вивчення можливостей інтеграції з іншими системами медичного закладу (лабораторія, відділення, харчування) та інших клінік у мережі.

23. Оцінка можливостей виходу на міжнародний ринок: У разі успішного впровадження в національних медичних закладах, розробка стратегій для просування продукту на міжнародних ринках.

Просування проекту модуля "е-реєстратура" потребує стратегічного підходу та чіткої структури. Кожен етап алгоритму орієнтований на забезпечення того, щоб продукт задовольняв потреби цільової аудиторії та вирішував реальні проблеми, з якими стикаються медичні установи. Такий підхід сприятиме успішному запуску та закріпленню модуля на ринку.

7.6 Визначення ключових факторів успіху проекту

Успішне впровадження модуля е-реєстратури в медичному закладі залежить від низки ключових факторів, які впливають на якість реалізації проекту, ефективність його функціонування та задоволення потреб користувачів. Ці фактори стосуються технологічних, організаційних і людських аспектів проекту.

Основою успіху є правильне розуміння потреб медичного персоналу, пацієнтів і адміністрації. Важливо врахувати специфіку закладу, поточні процеси та особливості роботи, щоб е-реєстратура дійсно стала корисним інструментом.

Регулярне залучення кінцевих користувачів на всіх етапах розробки сприятиме створенню інтуїтивно зрозумілого та зручного інтерфейсу.

Використання надійних і сучасних технологій, таких як Java, React, Spring Boot, PostgreSQL, гарантує стабільну роботу системи.

Реалізація високого рівня безпеки даних, наприклад, захист через JWT (JSON Web Token) та шифрування, знижує ризик витоку інформації.

КБПЗ_2024

Забезпечення гнучкості архітектури дозволяє легко масштабувати систему у разі розширення функціональності або інтеграції з іншими модулями.

Інтуїтивний інтерфейс, створений за допомогою сучасних бібліотек UI, таких як MUI, робить систему простою для всіх категорій користувачів.

Автоматизація рутинних процесів, наприклад, пошук пацієнта або запис до лікаря, значно зменшує навантаження на персонал.

Система повинна легко інтегруватися з іншими модулями закладу, такими як лабораторія, стаціонарне лікування, поліклініка.

Інтеграція з державними реєстрами або страховими компаніями, якщо це потрібно, значно підвищує функціональність системи.

Е-реєстратура має забезпечувати безперебійне обслуговування навіть при високому навантаженні, наприклад, у годину пік.

Здатність адаптуватися до зростання кількості пацієнтів або розширення закладу є критично важливою.

Важливим фактором є проведення навчання для співробітників, які працюватимуть із системою. Це допоможе зменшити опір змінам і підвищить ефективність впровадження.

Регулярні оновлення, технічна підтримка та зворотний зв’язок з користувачами гарантують тривалу функціональність і задоволення потреб.

Розробка повинна бути фінансово обґрунтованою. Зниження витрат на ручну обробку документів, оптимізація робочих процесів і підвищення продуктивності персоналу сприяють економічній доцільності проекту.

Система повинна відповідати законодавчим нормам щодо обробки персональних даних, таких як Закон України «Про захист персональних даних».

Постійний моніторинг продуктивності системи після впровадження допомагає виявляти проблеми та швидко їх вирішувати.

КПЗ_2024

8 ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ

8.1 Вступ

Охорона праці та забезпечення техніки безпеки є важливими складовими функціонування будь-якого підприємства, незалежно від його спеціалізації. Для програмно-аналітичного відділу, який займається розробкою програмного забезпечення, аналізом даних і технічною підтримкою інформаційних систем, питання охорони праці та безпеки також залишаються актуальними. Особливості діяльності такого відділу включають переважно офісну роботу з використанням комп’ютерної техніки та іншого обладнання, однак це не зменшує необхідності дотримання відповідних вимог до умов праці.

Відповідно до законодавства України, основними нормативними документами у сфері охорони праці є Закон України "Про охорону праці"[9], Кодекс законів про працю України (КЗпП), а також низка стандартів і положень, що регламентують забезпечення безпечних умов праці для працівників. Зокрема, ст. 153 КЗпП передбачає обов'язок роботодавця забезпечувати безпечні та здорові умови праці, а також відповідний контроль за їх дотриманням.

Робота програмістів та аналітиків пов'язана з довготривалим перебуванням за комп'ютером, що може призводити до фізичного, психоемоційного навантаження, а також впливу шкідливих факторів, таких як статична поза, напруження зору, електромагнітне випромінювання та недостатня фізична активність. Окрім цього, програмно-аналітична діяльність вимагає організації умов, які мінімізують ризики аварійних ситуацій у разі роботи з інформаційними системами, забезпечують збереження здоров'я працівників і високу продуктивність.

КПІ-2024

У цьому розділі буде детально розглянуто заходи, спрямовані на дотримання принципів охорони праці та техніки безпеки для програмно-аналітичного відділу підприємства. Зокрема, увага приділятиметься:

- аналізу нормативної бази України у сфері охорони праці;
- організації безпечного робочого місця працівників;
- рекомендаціям щодо профілактики професійних захворювань;
- технічним та організаційним заходам із забезпечення безпеки праці;
- відповідальності роботодавця та працівників за дотримання норм охорони праці.

Впровадження сучасних стандартів охорони праці є не лише обов'язковим відповідно до чинного законодавства, але й вагомим кроком у забезпеченні комфортних умов праці, підвищенні ефективності діяльності персоналу та зміцненні репутації підприємства як соціально відповідального.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Організація комфортних та безпечних умов праці є важливим фактором забезпечення ефективності роботи програмістів. У цьому розділі проаналізуємо санітарно-гігієнічні умови в приміщенні площею 60 м², де працює 9 співробітників.

1. Площа приміщення та розміщення робочих місць

Згідно з державними будівельними нормами України (ДБН)[16], на кожного працівника має припадати не менше 4,5 м² вільного простору. У приміщенні на 60 м² це забезпечує комфортну роботу для 9 співробітників, враховуючи простір для обладнання, меблів та проходів.

Робочі місця розташовані з урахуванням:

– мінімальної відстані між столами програмістів (не менше 1 м);

– безперешкодного доступу до аварійних виходів і зон евакуації;

КОПЗ_2024

– ергономічного облаштування робочих місць для зниження втоми працівників.

2. Освітлення

Освітлення в приміщенні складається з природного та штучного.

Природне світло надходить через вікна площею 15% від загальної площі приміщення. Вікна обладнані жалюзі для регулювання яскравості світла.

Штучне освітлення має відповідати нормативам: лампи з нейтральним білим світлом (4000-5000 К) забезпечувати рівномірну яскравість. Освітленість має становить 500-750 люкс на рівні робочих поверхонь.

Рекомендовано уникати бликів на екранах моніторів шляхом правильного розташування столів та світильників.

3. Мікроклімат

Відповідно до ДСанПіН 2.2.4-171-10, температура в офісі має бути в межах 22-24°С при вологості повітря 40-60%. У приміщенні має бути встановлено:

– кондиціонер для регулювання температури в літній період;
– централізоване опалення взимку;
– зволожувачі для підтримання оптимальної вологості.

Системи вентиляції мають забезпечувати обмін повітря на рівні 60 м³/год на одного працівника, що відповідає нормам.

4. Шумове навантаження

Рівень шуму на робочому місці програмістів не повинен перевищувати 50 дБ (згідно з Санітарними нормами мікроклімату виробничих приміщень ДСН 3.3.6.042-99). У приміщенні має бути використано шумоізоляційні матеріали:

– килимові покриття для підлоги;
– акустичні панелі на стінах і стелі.

Обладнання, як то маршрутизатори та сервери, мають бути встановлені у віддаленому від робочих місць куті, щоб мінімізувати шумове навантаження.

КПЗ_2024

5. Організація робочих місць

Кожне робоче місце обладнане ергономічними меблями:

– столи з регульованою висотою;
– ортопедичні крісла з підтримкою спини;
– з необхідності підставки для ніг.

Монітори розташовані на відстані 50-70 см від очей, їх висота налаштована так, щоб верхній край екрану був трохи нижчим за рівень очей.

6. Електромагнітне випромінювання

Для зменшення впливу електромагнітного випромінювання має бути передбачено:

– заземлення електрообладнання;
– використання якісних екранів із низьким рівнем мерехтіння;
– дотримання вимоги безпечної відстані між пристроями (не менше 1 м).

7. Психоемоційні фактори

У приміщенні передбачено окрему зону для відпочинку, де працівники можуть відновлювати сили. Її площа приблизно має становить 12 м² і вона має бути обладнана зручними меблями та кавоваркою.

Створення дружнього середовища мінімізує стресові фактори. Високий рівень природного освітлення та доступ до кондиціонованого свіжого повітря будуть сприяти підвищенню продуктивності праці і комфорту.

Додатковий рівень комфорту потребує:

– установити автоматичні системи контролю температури та вологості.
– організувати періодичні перерви для виконання вправ на розминку.
– забезпечити регулярне провітрювання приміщення.
– запровадити програму моніторингу здоров’я співробітників.

КБПЗ - 2024

Таким чином, аналіз санітарно-гігієнічних умов праці свідчить, що за умови виконання рекомендацій можна створити максимально комфортне, безпечне та продуктивне робоче середовище для програмістів.

8.3 Розробка заходів з поліпшення стану охорони праці

Ефективна організація охорони праці є невід'ємною складовою забезпечення безпеки та комфорту на робочих місцях. Для покращення комфорту і стану охорони праці потрібно провести наступні заходи.

1. Провести оцінку ризиків і контроль небезпечних факторів.

Для поліпшення стану охорони праці важливо регулярно проводити оцінку ризиків і впроваджувати відповідні заходи контролю. Основні кроки:

Виявити та провести аналіз потенційних небезпек (недостатнє освітлення, надмірний шум, високий рівень електромагнітного випромінювання).

Розробити карти ризиків для приміщення, що охоплює робочі зони, розташування електрообладнання та шляхи евакуації.

Забезпечити проведення тренінгів для працівників із запобігання професійним небезпекам.

2. Оптимізувати мікроклімат та вентиляцію у приміщенні.

Робота в умовах комфортного мікроклімату знижує втомлюваність і ризик захворювань. Рекомендовані для цього заходи:

- Установлення сучасної вентиляційної системи, що забезпечує обмін повітря 60 м³/год на працівника.
- Контроль температурного режиму на рівні 22-24°C у холодний сезон та 24-26°C у теплий сезон.
- Використання зволожувачів повітря для підтримання вологості на рівні 40-60%.
- Регулярне технічне обслуговування систем опалення, кондиціонування та вентиляції.

КПЗ - 2024

– Забезпечити правильне і комфортне освітлення

Правильне освітлення сприяє збереженню здоров'я зору та підвищенню продуктивності. Заходи з поліпшення:

– Встановлення додаткових світильників з нейтральним білим світлом (4000-5000 К) для забезпечення освітленості 500-750 люкс.
– Обладнати робочі місця моніторами для запобігання відблискам від природного світла.
– Регулярно обслуговувати світильники і чистити вікна для підтримання максимальної пропускної здатності світла.

4. Шумозахист

Зменшення шумового навантаження позитивно впливає на нервову систему працівників. Потрібно провести наступні заходи:

– Використовувати звукопоглинаючі матеріали на стінах та стелі.
– Розташувати маршрутеризатори та сервери у спеціально відведеній зоні.
– Забезпечити обмеження рівня шуму техніки до допустимих 50 дБ.

5. Ергономіка робочих місць

Забезпечення ергономіки робочого середовища є ключовим фактором зниження фізичної втоми.

Старатись використовувати ергономічні меблі: регульовані столи, ортопедичних крісла із підтримкою спини та регульованою висотою сидіння.

Використовувати спеціальні підставки для моніторів і ніг.

Зонувати робочі місця для зменшення психологічної напруженості в колективі.

6. Навчання та інструктажі

Систематично проводити навчання працівників, що сприяє підвищенню обізнаності та зниженню ризиків травматизму. Рекомендації:

– Щоквартальне проведення інструктажів з охорони праці.
– Навчання правил надання першої домедичної допомоги.
– Практичні тренінги з евакуації в надзвичайних ситуаціях.

КПЗ_2024

7. Організація зон відпочинку

Забезпечити створення працівникам зон відпочинку, що буде сприяти відновленню працездатності:

– Виділити частину приміщення (не менше 12 м²) для облаштування зони відпочинку із зручними меблями та кулером для води.
– Розмістити в зоні відпочинку кімнатні рослини для створення комфортної атмосфери.

8. Запровадити заходи з профілактики захворювань

Здоров’я працівників залежить від профілактичних заходів:

– Організація щорічних медичних оглядів.
– Забезпечення доступу до дезінфікуючих засобів для рук та регулярної дезінфекції приміщень.
– Моніторинг якості повітря та температурного режиму.

9. Контроль електробезпеки

Для уникнення ризиків, пов'язаних із використанням електропристроїв, потрібно проводить:

– Регулярну перевірку заземлення електрообладнання.
– Встановити пристрої захисного відключення (ПЗВ).
– Створити безпечне прокладання кабелів і подовжувачів.

10. Створити умови підвищення психологічного комфорту

Дружня атмосфера у колективі сприяє ефективності роботи:

– Проводити командні тренінги і заходи для формування позитивного мікроклімату.
– Встановити політику прозорого розв'язання конфліктів.

11. Моніторинг і контроль

Для оцінки ефективності впроваджених заходів має бути:

– Щорічний аудит умов праці.
– Розробка звітів за результатами аналізу санітарно-гігієнічних умов.
– Залучення працівників до оцінювання та внесення пропозицій щодо

КОПЗ_2024

поліпшення умов.

Таким чином, реалізація зазначених заходів створить комфортне та безпечне середовище, сприятиме підвищенню продуктивності працівників та забезпечить відповідність умов праці чинним нормативним актам України.

8.4 Техніка безпеки та протипожежна профілактика програмно-аналітичного відділу

Організація роботи програмно-аналітичного відділу потребує дотримання норм техніки безпеки та протипожежної профілактики відповідно до законодавства України, зокрема Закону України "Про охорону праці", Кодексу цивільного захисту України, а також Правил пожежної безпеки в Україні. Основним завданням є забезпечення безпечних умов праці, мінімізація ризиків виникнення пожежі та запобігання нещасним випадкам на робочому місці.

Загальні вимоги до техніки безпеки

1. Організація робочих місць :

- Робочі місця програмістів та аналітиків повинні бути обладнані з урахуванням ергономічних стандартів, щоб уникнути перенапруження та травм.

- Кабелі живлення й мережеві дроти повинні бути зафіксовані таким чином, щоб уникнути ризику спотикання.

2. Електробезпека :

- Усі електроприлади мають бути заземлені та перевірені на відповідність стандартам безпеки.

- Забороняється використовувати несправні розетки, подовжувачі або електроприлади.

- Працівники мають бути проінструктовані щодо правил користування електрообладнанням.

3. Санітарно-гігієнічні умови :

КПЗ – 2024

- Провітрювання приміщення слід забезпечувати не менше двох разів на день.

- Встановити систему кондиціонування для підтримання комфортного мікроклімату (температура 20–24 °C, вологість 40–60%).

4. Освітлення :

- Робочі місця повинні мати достатнє природне та штучне освітлення, відповідне нормам ДСТУ.

Протипожежна профілактика

1. Організація системи протипожежного захисту :

- У приміщенні програмно-аналітичного відділу мають бути встановлені автоматичні системи пожежної сигналізації та сповіщення.

- Забезпечити наявність вогнегасників (порошкових або вуглекислотних) із розрахунку один вогнегасник на кожні 20 кв. м площі.

2. Заборона використання небезпечного обладнання :

- У приміщенні забороняється куріння та використання відкритого вогню.

- Використання електрообладнання, яке не відповідає нормам, категорично заборонено.

3. Евакуаційні шляхи :

- Всі евакуаційні шляхи повинні бути вільними та чітко позначеними.

- Евакуаційні виходи мають бути обладнані знаками безпеки, які світяться у темряві, відповідно до ДСТУ 7313:2013 «Знаки безпеки та системи евакуаційні фотолюмінісцентні».

4. Профілактичні заходи :

- Регулярна перевірка електричних розеток, проводів, обладнання на предмет короткого замикання.

- Щоквартальне навчання працівників правилам пожежної безпеки.

5. Навчання працівників

- Перед початком роботи кожен співробітник має пройти інструктаж із техніки безпеки та протипожежної профілактики.

КОПЗ_2024

- Планові інструктажі повинні проводитися щокварталу, а позапланові – у разі змін у законодавстві або умовах праці.

- Працівники повинні знати порядок дій у разі пожежі:

1. Негайно сповістити службу порятунку за телефоном 101.
2. Використовувати засоби пожежогасіння до прибуття пожежних підрозділів.
3. Забезпечити евакуацію згідно з планом.

6. Контроль і моніторинг

- Призначити відповідальну особу за пожежну безпеку, яка регулярно проводитиме перевірки обладнання та умов роботи.

- Щомісячно проводити огляд протипожежного обладнання та стану евакуаційних виходів.

- Вести журнал обліку інструктажів та перевірок.

Дотримання техніки безпеки та протипожежної профілактики в програмно-аналітичному відділі є ключовим чинником запобігання надзвичайним ситуаціям. Систематичне навчання персоналу, належне технічне обладнання та контроль за виконанням норм забезпечать безпечні умови праці та підвищать продуктивність співробітників.

КбПЗ-2024

8.5 Розрахункова частина

Виконати розрахунок захисного занулення електроустановки потужністю 3.8 кВт.

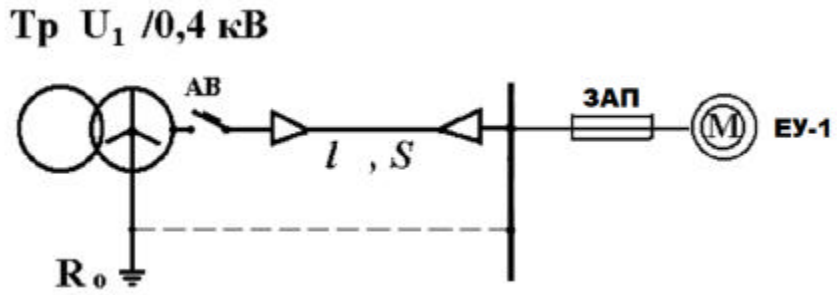


Рисунок 8.1-Схема електричної мережі

Початкові данні:

- Потужність установки Р=3,8кВт
- Лінійна напруга U=380В;
- Довжина кабелю до установки L=30 м.
- Фазна напруга – 220В;

Розрахунок [21]

Визначаємо силу номінального струму електроустановки

$$I_{\text{ном}} = \frac{P \cdot 1000}{\sqrt{3} \cdot U_{\text{л}} \cdot \cos\varphi} = \frac{3.8 \cdot 1000}{\sqrt{3} \cdot 380 \cdot 0.8} = 7.217\ A$$

де Р − номінальна потужність установки, кВт.;

$U_{\text{л}}$ − лінійна напруга, В;

$\cos\phi$ − коефіцієнт потужності, приймається в залежності від типу електрообладнання в межах 0,8..0,87.

Визначаємо силу пускового струму установки, А :

$$I_{\text{пус}} = 5 \cdot I_{\text{н}} = 5 \cdot 7.217 = 36{,}085\ \text{А}$$

Визначаємо номінальну силу струму апарата захисту :

КПЗ_2024

$$I_{\text{н}} = \frac{I_{\text{пус}}}{\beta} = \frac{36.085}{2.5} = 14.434 \text{ А}$$

де ß − коефіцієнт пуску, приймається : − для легких умов пуску − 2,5..3. З таблиці 1[21] обираємо запобіжник ПН 2-100 з плавкою вставкою $I_{ном}$ =30А

Визначаємо найменше допустиме по умовам спрацьовування захисту значення сили струму короткого замикання, А

$$I_{kmin} = I_{н} \cdot К = 30 \cdot 3 = 90 \text{ А}$$

де $I_{н}$ − номінальний струм апарата захисту табл.1)

К − коефіцієнт надійності;

значення коефіцієнта К приймається :

− при захисті автоматичним вимикачем зі зворотною залежністю від струму характеристикою К=3

− при захисті плавкими запобіжниками : К=3

Знаходимо переріз провода або кабеля з умови допустимого нагрівання:

$$I_{доп} \leq I_{max}$$

$$I_{доп} \leq 7.217 \text{ А}$$

де $I_{доп}$ − тривало допустимий із умови нагрівання струм навантаження провідника, А.

Площа перерізу узгоджується з номінальним струмом плавкої вставки запобіжника із умови :

$$I_{доп} \geq I_{вст}/\alpha = 30/3 = 10 \text{ А}$$

де α − коефіцієнт відповідності, який залежить від умов прокладання і нагляду за мережею : α = 3 − для промислових мереж.

Площа перерізу вибирається по більшому із розрахованих по формулам (у нас 7,217 А і 10 А.) з таблиці 3[21].

КбПЗ-2024

Вибираємо площу перерізу 2,5 мм^2 ($S_ф$) при числі проводів і = 3 табл. 3[21], кабель АВВБ розташований у повітрі.

Визначаємо струм спрацювання теплового або електромагнітного розчиплювача у щитовій автоматичного вимикача:

$$I_{спр} \geq I_{роб} \geq 7.217 \text{ А}.$$

Струм спрацювання електромагнітного розчиплювача додатково перевіряємо по максимальному струму перевантаження лінії:

$$I_{спр} \geq 1.25 \cdot I_{пер} = 1.25 \cdot 36{,}085 = 45{,}106 \text{ А}.$$

Струм спрацювання розчіплювача і автоматичний вимикач вибираємо з табл.2[21].

Приймаємо $I_{спр}$ = 80 А. Вимикач : А 3714Б.

Визначаємо площу перерізу нульового захисного провідника S_n із умов:

$$I_{kmin} \geq K \cdot I_н$$

$$\frac{1}{Z_n} \geq 0.5 \cdot \frac{1}{Z_ф}$$

де Z_n і $Z_ф$– повні опори відповідно нульового і фазного провідників, визначаємо орієнтовно площу перерізу провідника. Для кабелю

$$S_n = 0{,}5 \cdot S_ф = 0{,}5 \cdot 2{,}5 = 1{,}2 \text{ мм}^2 .$$

По табл.3 [21] округляємо ці значення до найближчих 2,5мм^2.

Визначаємо активний і індуктивний опір фазного і нульового захисного провідників

КбПЗ_2024

$$R_{\text{ф}} = \rho \cdot \frac{L}{S_{\text{ф}}} = 0{,}028 \cdot \frac{30}{2{,}5} = 0{,}036 \text{ Ом}$$

Так як фазний і нульовий провідники мають однаковий переріз, то і опори фазного і нульового провідників однакові.

$$R_{\text{ф}} = R_{\text{n}} = 0.036 \text{ Ом}$$

Визначаємо індуктивний опір. Для окремо проложених нульових провідників його приймають рівним 0,6 Ом/км. При прокладці кабелем, або в сталевих трубах індуктивним опором нехтують.

Знаходимо дійсне значення (модуль) стума однофазного короткого замикання.

$$I_{kp} = \frac{U_{\text{ф}}}{\frac{Z_T}{3} + \sqrt{(R_{\text{ф}} + R_n)^2}} = 698\ A$$

Із табл.4[21] для трансформатора на 160 кВА Z_T =0,141 Ом.

Якщо значення $I_{кр}$ перевищує значення найменшого допустимого по умовам спрацювання захисту I_{kmin} (90 А), то захисний провідник вибраний вірно.

Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$U_{kmax} = I_{кр} \cdot Z_{н} \leq U_{дан.д.}$$

У нас

$$Z_{\text{n}} = R_{\text{n}} \cdot U_{kmax} = 698 \cdot 0{,}036 \text{ Ом} = 25 \text{ В} < 36 \text{ В}$$

де $U_{дан.д.}$ − це допустима напруга, що нормується по ГОСТ 12.1.038-82. При часі дії більше 1с приймається 36 В. Z_{n}− повний опір нульового провідника. Умова виконується, значить нульовий провідник обрано вірно.

Висновки до розділу

КПІЗ_2024

Охорона праці та техніка безпеки є невід'ємними складовими функціонування програмно-аналітичного відділу будь-якого підприємства. Забезпечення здорових і безпечних умов праці для працівників сприяє не лише збереженню їхнього фізичного та психоемоційного стану, але й підвищує ефективність виконання професійних обов'язків.

У ході аналізу основних аспектів охорони праці було визначено ключові шкідливі та небезпечні фактори, що впливають на працівників програмно-аналітичного відділу. Серед них — тривала робота за комп'ютером, напруження зору, статичне навантаження на опорно-руховий апарат, низька фізична активність, а також можливий вплив електромагнітного випромінювання. Для мінімізації цих ризиків розроблено комплекс заходів, які включають організацію ергономічного робочого місця, впровадження перерв для гімнастики, забезпечення відповідного освітлення, дотримання правил користування комп'ютерною технікою та регулярні профілактичні медичні огляди.

Також важливим аспектом є дотримання нормативних вимог охорони праці, зокрема положень Закону України "Про охорону праці", Кодексу законів про працю та інших нормативно-правових актів. Дотримання цих норм забезпечує зниження ризиків травматизму, професійних захворювань і виникнення аварійних ситуацій. Організаційні заходи, такі як проведення інструктажів, навчання персоналу, моніторинг умов праці та забезпечення працівників необхідними засобами захисту, є обов'язковими для виконання.

Особлива увага приділяється заходам щодо психоемоційного настрою працівників, оскільки специфіка роботи програмно-аналітичного відділу часто пов'язана з високими розумовими навантаженнями та роботою в умовах підвищеної відповідальності. Впровадження гнучкого графіка роботи, забезпечення комфортного мікроклімату в приміщенні, створення дружньої атмосфери в колективі та підтримка ініціатив працівників позитивно впливають на їхню продуктивність і загальне задоволення роботою.

КОПЗ_2024

Отже, реалізація запропонованих заходів щодо охорони праці та техніки безпеки програмно-аналітичного відділу дозволить значно підвищити рівень безпеки та комфорту працівників, зменшити ризик захворювань і травматизму, а також сприятиме створенню позитивного іміджу підприємства як соціально відповідальної організації.

КБПЗ_2024

9 ОСНОВНІ ВИСНОВКИ

Розробка програмного модуля "е-реєстратура" для медичного закладу є важливим кроком у вдосконаленні надання медичних послуг в сучасних умовах цифровізації. Проєкт довів свою актуальність, новизну, а також перспективи практичного застосування.

Актуальність розробки зумовлена потребою оптимізації роботи медичних установ, де значну частину часу займають процеси реєстрації пацієнтів, планування візитів та обробки медичних даних. Традиційні способи управління інформацією часто призводять до перевантаження персоналу, помилок у документації, а також неефективного використання робочого часу лікарів та медичних адміністраторів.

Модуль "е-реєстратура" забезпечує автоматизацію цих процесів, знижує ризик помилок і сприяє підвищенню загальної якості обслуговування пацієнтів.

Проект також відповідає сучасним вимогам українського законодавства щодо ведення електронних медичних записів, враховуючи їх інтеграцію в загальнодержавні системи обліку та звітності, такі як eHealth.

Новизна проекту полягає в інтеграції сучасних технологій для управління даними пацієнтів. Система використовує передові підходи, такі як:

– Впровадження бібліотеки JWT для безпечного обміну даними.

– Застосування мікросервісної архітектури для забезпечення гнучкості та масштабованості модуля.

– Інтеграція пошуку пацієнтів за унікальними ідентифікаторами, такими як штрих-коди або відбитки пальців, що спрощує та прискорює доступ до даних.

Крім того, модуль "е-реєстратура" враховує можливість інтеграції зі сторонніми сервісами, такими як лабораторні системи чи інші медичні підсистеми, що дозволяє забезпечити єдиний інформаційний простір для всіх структур медичного закладу.

КПЗ_2024

Практичне значення розробки полягає у створенні ефективної системи управління взаємодією між пацієнтами та медичним закладом. Основні сфери застосування включають:

– Автоматизація процесів реєстрації: швидке створення облікових записів пацієнтів, реєстрація візитів та керування записами.

– Оптимізація обслуговування пацієнтів: зниження часу очікування завдяки електронному плануванню.

– Захист даних: реалізація сучасних підходів до кібербезпеки для збереження конфіденційності персональної інформації пацієнтів.

– Скорочення витрат: мінімізація потреби у паперовій документації та зниження навантаження на персонал.

Проект також є універсальним і може бути адаптований для використання у медичних установах різного масштабу — від невеликих приватних клінік до багатопрофільних лікарень.

Проект "е-реєстратура" демонструє комплексний підхід до вирішення завдань автоматизації та оптимізації процесів медичних установ. Впровадження такого модуля дозволяє не лише полегшити роботу медичного персоналу, але й створити кращий користувацький досвід для пацієнтів. Новизна підходів та відповідність сучасним технологічним стандартам гарантують його перспективність та ефективність у практичному застосуванні.

Основний результат цієї кваліфікаційної роботи апробовано на IX Міжнародній науково-практичній конференції «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» та опубліковані [51].

КОПЗ-2024

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React - React Native 2020 [Електронний ресурс] - Режим доступу: URL: https://uk.wikipedia.org/wiki/React#React_Native (дата звернення: 27.11.2024).

2. REST [Електронний ресурс] — Режим доступу: URL: https://uk.wikipedia.org/ wiki/REST (дата звернення: 27.11.2024).

3. Сервісно орієнтоване програмування [Електронний ресурс] — Режим доступу: URL: https://www.slideshare.net/v_matush/service-oriented-programming (дата звернення: 27.11.2024).

4. Redux Core [Електронний ресурс] — Режим доступу: URL: https://redux.js.org/introduction/gettingstarted (дата звернення: 27.11.2024).

5. Fullstack React Native [Електронний ресурс] — Режим доступу: URL: https://coursehunter.net/course/knigafullstack-react-native-polnoe-rukovodstvo-po-react-native (дата звернення: 27.11.2024).

7. Practical React Query. [Електронний ресурс] — Режим доступу: URL: https://tkdodo.eu/blog/practical-react-query (дата звернення: 27.11.2024).

8. Сайт React.js [Електронний ресурс] — Режим доступу: URL: https://uk.reactjs.org (дата звернення: 27.11.2024).

9. Сайт Javascript Features [Електронний ресурс] — Режим доступу: URL: http://simplyaccessible.com/article/the-accessibility-stack (дата звернення: 27.11.2024).

10. Стаття про REST [Електронний ресурс] — Режим доступу: URL: https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part1-introduction-b4a072f8740f (дата звернення: 27.11.2024).

11. Стаття про MVC [Електронний ресурс] — Режим доступу: URL: https://alexatnet.com/model-view-controller-mvc-in-javascrip (дата звернення: 27.11.2024).

12. JSON Web Tokens Documentation. [Електронний ресурс] — Режим доступу: URL: https://jwt.io/introduction/ (дата звернення: 27.11.2024).

13. Використання React JS.[Електронний ресурс] — Режим доступу: URL:

КПІЗ_2024

https://uk.education-wiki.com/6785493-usesof-react-js (дата звернення: 27.11.2024).

14. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. 2021. [Електронний ресурс] — Режим доступу: URL: https://uk.reactjs.org/ (дата звернення: 27.11.2024).

15. Container Components in React. 2021 –[Електронний ресурс] — Режим доступу: URL: https://www.section.io/engineering-education/container-components-in-react/ (дата звернення: 27.11.2024).

16. Державні будівельні норми України: ДБН В.2.5-28:2018 — Режим доступу: URL: https://goo.su/9AkQ (дата звернення: 27.11.2024).

17. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. — Режим доступу: URL:https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 27.11.2024).

18. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. — Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 27.11.2024).

19. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». — Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0508 (дата звернення: 27.11.2024).

20. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. Режим доступу: URL:https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 27.11.2024).

21. Охорона праці. Ч. 2. Занулення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ − сумісного типу. / Оришака О.В., Солових Є.К., Оришака В.О., Солових А.Є., Катеринич С.Е.; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький: ЦНТУ, 2019. - 27 с. [Електронний ресурс]. Режим доступу:

КОПЗ_2024

URL:http://dspace.kntu.kr.ua/jspui/handle/123456789/8769 (дата звернення: 27.11.2024).

22. Медичний сервіс для пацієнтів і лікарів Health24 (Здоров'я 24). - [Електронний ресурс]. Режим доступу: URL: https://h24.ua/ (дата звернення: 27.11.2024)

23. ІМЕД — медична інформаційна система з повним набором інструментів для автоматизації медичного бізнесу та його основних бізнес процесів). - [Електронний ресурс]. Режим доступу: URL: https://imed.ua/ (дата звернення: 27.11.2024)

24. Clinica WEB – сучасна електронна медична система, створена для задоволення потреб пацієнтів, лікарів та медичних закладів. - [Електронний ресурс]. Режим доступу: URL: https://www.clinica-web.ua (дата звернення: 27.11.2024)

25. MIC для малого та середнього медбізнесу - Doctor Eleks. - [Електронний ресурс]. Режим доступу: URL: https://doctor.eleks.com/ (дата звернення: 27.11.2024)

26. Медична інформаційна система (MIC) для комплексної автоматизації основних процесів медичних устано.- [Електронний ресурс]. Режим доступу: URL: https://emci.ua/products/emcimed (дата звернення: 27.11.2024)

27. DocDream: Електронна медична карта.- [Електронний ресурс]. Режим доступу: URL: https://docdream.com (дата звернення: 27.11.2024)

28. Clinica WEB – сучасна електронна медична система, створена для задоволення потреб пацієнтів, лікарів та медичних закладів. - [Електронний ресурс]. Режим доступу: URL: https://www.clinica-web.ua (дата звернення: 27.11.2024)

29. Система MedExpert предназначена для автоматизации процессів планирования, учета и анализа роботи клініки. - [Електронний ресурс]. Режим доступу: URL: https://medexpert.net.ua (дата звернення: 27.11.2024)

30. EvoMIS – медична інформаційна система для автоматизації роботи

КОПЗ - 2024

медичних закладів. - [Електронний ресурс]. Режим доступу: URL: https://evomis.com (дата звернення: 27.11.2024)

31. Binomed - ПЗ для медичних установ. - [Електронний ресурс]. Режим доступу: URL: https://evomis.com (дата звернення: 27.11.2024)

32. PostgreSQL: The world's most advanced open source database. - [Електронний ресурс]. Режим доступу: URL: https://www.postgresql.org/ (дата звернення: 27.11.2024)

33. React — це JavaScript-бібліотека для створення інтерфейсів користувача. - [Електронний ресурс]. Режим доступу: URL: https://uk.legacy.reactjs.org/ docs/getting-started.html (дата звернення: 27.11.2024)

34. Роберт Сесіл Мартін. Чиста архітектура: Київ : Фабула, 2019. 368 с.

35. React – JavaScript-бібліотека для створення користувацьких інтерфейсів [Електронний ресурс] – Режим доступу до ресурсу: URL: https://reactjs.org (дата звернення: 27.11.2024)

36. React-Native – фреймворк для створення користувацьких інтерфейсів[Електорнний реурс] – Режим до ресурсу: URL: https://reactnative.dev (дата звернення: 27.11.2024)

37. Використання State Hook – React [Електронний ресурс] – Режим доступу: URL: https://reactjs.org/docs/hooks-state.html (дата звернення: 27.11.2024)

38. React: Virtual DOM [Електронний ресурс] – Режим доступу: URL: https://www.codecademy.com/article/react-virtual-dom (дата звернення: 27.11.2024)

39. Посібник: Quick start. [Електронний ресурс]. – Режим доступу : URL: https://react.dev/learn (дата звернення: 27.11.2024)

40. Tutorial: Tic-Tac-Toe. [Електронний ресурс]. – Режим доступу : URL: https://react.dev/learn/tutorial-tic-tac-toe (дата звернення: 27.11.2024)

41. Head First. Програмування на JavaScript. Ерік Фрімен, Елізабет Робсон. – К.: Фабула, 2022. 672 с.

42. HELSI - інформаційна система для пацієнтів [Електронний ресурс] – Режим доступу до ресурсу: URL: https://helsi.me/ (дата звернення: 27.11.2024)

КПЗ_2024

43. Сервіс запису до лікаря на прийом в Києві онлайн [Електронний ресурс] – Режим доступу до ресурсу: URL: https://doc.ua/ (дата звернення: 27.11.2024)

44. Клієнт-серверна архітектура та ролі серверів. [Електронний ресурс] / Іван Змерзлий // Medium. – 2017. – Режим доступу до ресурсу: URL: https://medium.com/IvanZmerzlyi (дата звернення: 27.11.2024)

45. Zosym M. Діаграми послідовностей (Sequence Diagrams) [Електронний ресурс] / Maxym Zosym // Maxym Zosym. – 2022. – Режим доступу до ресурсу: URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 27.11.2024)

46. Best Programming Languages for Web Development [Електронний ресурс] // MOOC.org. – 2021. – Режим доступу до ресурсу: URL: https://www.mooc.org/blog/best-programming-languages-for-web-development (дата звернення: 27.11.2024).

47. Сучасний підручник з JavaScript [Електронний ресурс] // Javascript.info – Режим доступу до ресурсу: URL: https://uk.javascript.info/ (дата звернення: 27.11.2024).

48. Веб-фреймворк Express (Node.js/JavaScript) [Електронний ресурс] – Режим доступу до ресурсу: URL: https://developer.mozilla.org/ru/docs/Learn/Server- side/Express_Nodejs (дата звернення: 27.11.2024)

49. Introduction to MongoDB [Електронний ресурс] – Режим доступу: https://www.mongodb.com/docs/manual/introduction/ (дата звернення: 27.11.2024)

50. Tutorial: Intro to React [Електронний ресурс] – Режим доступу до ресурсу: URL: https://legacy.reactjs.org/tutorial/tutorial.html (дата звернення: 27.11.2024).

51. Доренський О. П. Структурна модель цифровізованих бізнеспроцесів лікувально-діагностичного центру / О. П. Доренський, С. В. Беседа // Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі : IX Міжнар. наук.-практ. конф., 25-26 кві. 2024 р. - К. : КНУКіМ, 2024. – С. 27-28. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/43248/159542.pdf?sequence=2&isAllowed=y (дата звернення: 11.11.2024).

КПЗ - 2024

Додаток А
(обов'язковий)

Технічне завдання

Зміст

КОПЗ–2024

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію модуля електронного обліку пацієнтів лікувально-діагностичного центру.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 7.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження і програмна реалізація модуля е-реєстратури лікувально-діагностичного центру.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

КБПЗ_2024

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи електронного обліку пацієнтів лікувально-діагностичного центру;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на кількість користувачів та обліку пацієнтів.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КбПЗ_2024

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 та браузер Chrome і працювати з принтерами і реєстраторами фіскальних операцій.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС Windows 10/11 і браузером Chrome.

5.8.1 Обладнання

Комп’ютер Intel® i3-5gen/8Gb/SSD 128Gb/19” або сумісні з ним.

5.8.2 Мова програмування

Бекенд – Java/Spring;

Фронтенд – JavaScript/React

КБПЗ 2024

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

КбПЗ_2024

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема роботи програми – 2 аркуша.
– Маркетингове та економічне обґрунтування ІТ-процесу – 1 аркуш.
– Пояснювальна записка – 113 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 7.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 16.12.2024 р.

КПЗ – 2024

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ О. П. Доренський

КбПЗ_2024

Дослідження і програмна реалізація модуля е-реєстратури лікувально-діагностичного центру

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 45

Літера: РП

Кропивницький – 2024 року

```
//Дашбоард пошуку або заведення картки пацієнта
import HighlightOffIcon from '@mui/icons-material/HighlightOff';
import PersonAddIcon from '@mui/icons-material/PersonAdd';
import { LoadingButton } from '@mui/lab';
import {
  Avatar,
  CardActionArea,
  CardActions,
  Chip,
  Grid,
  InputAdornment,
  Paper,
  TextField,
  Typography,
} from '@mui/material';
import Button from '@mui/material/Button';
import Card from '@mui/material/Card';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import IconButton from '@mui/material/IconButton';
import female_img from 'assets/img/avatar-f.png';
import male_img from 'assets/img/avatar-m.png';
import birth_img from 'assets/img/birthday.png';
import card_image from 'assets/img/peception1.jpg';
import { DataStatus } from 'common/enums/app/data-status.enum';
import { getFormattedDate } from 'helpers/date/get-formatted-date/get-
formatted-date';
import { useAppDispatch, useAppSelector } from 'hooks/hooks';
import { PatientDetailModal } from 'modules/register/modal/patient-detail-
modal';
import * as React from 'react';
import { FC, useEffect, useState } from 'react';
import Barcode from 'react-barcode';
import { getCityList, getFindPatient, getRequisites, getStreetList } from
'store/register/patient/actions';
import { clearFindPatient, clearPatientDetails, clearPatientLogData } from
'store/register/patient/reducer';
import { getPatientFullYears } from '../helper/get-patient-full-years';
const RegisterDashboard: FC = () => {
  const dispatch = useAppDispatch();
  const { patients , patientLogDataStatus }
    = useAppSelector(( { patient } ) => ({
      patients: patient.patientFind,
```

КбТЗ 2024

```
      patientLogDataStatus: patient.patientLogDataStatus ,
    }));
  const [openPatientDetailModal, setOpenPatientDetailModal] =
useState(false);
  const [findPatient , setfindPatient] = React.useState('');
  const [patientCodeToOpen , setPatientCodeToOpen] = React.useState(0);
  const isLoading = patientLogDataStatus===DataStatus.PENDING;
  useEffect(() => {
    dispatch(getRequisites());
    dispatch(getCityList());
    dispatch(getStreetList());
    const delayDebounceFn = setTimeout(() => {
      if (findPatient.length>2) {
        dispatch(getFindPatient(findPatient));
      } else {
        dispatch(clearFindPatient());
      }
    }, 3000);

    return () => clearTimeout(delayDebounceFn);
  }, [dispatch,findPatient]);
  return (
    <div style={{ backgroundImage:  `url(${card_image})`,
      backgroundPosition: 'center',
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      width: '84vw',
      height: '90vh' }}>
      <PatientDetailModal patientCode = { patientCodeToOpen }
        isOpen= { openPatientDetailModal }
        onClose={():void=>setOpenPatientDetailModal(false)}/>
      <div style={ { padding: '10px', display: 'flex', opacity: '0.95' } } >
        <Paper
          component="form"
          sx={{ p: '10px', display: 'flex', alignItems: 'center', width:
'30%' }}
        >
          <TextField
            InputProps={{
              startAdornment: (
                <InputAdornment position="start" style={ { width: '50px',
margin: 0, padding: 0 } }>
                  <LoadingButton
```

КоПЗ – 2024

```
                    style={{ border: 3 , margin: 0, padding: 0 }}
                    size="small"
                    loading={ isLoading }
                    variant="outlined"
                  >
                  </LoadingButton>
                </InputAdornment>
              ),
              endAdornment: (
                <InputAdornment position="start">
                  <IconButton
                    onClick={():void=>{
                      setfindPatient('');
                    }}
                    type="button" sx={{ p: '1px', transform: 'scale(1.2)' }}
>
                    <HighlightOffIcon />
                  </IconButton>
                </InputAdornment>
              ),
            }}
            autoFocus= { true }
            id = "findPatient"
            size="medium"
            variant="standard"
            value={ findPatient }
            sx={ { ml: 0, flex: 1 } }
            placeholder="Пошук..."
            autoComplete={ 'off' }
            onKeyDown={ ( e): void => { e.key === 'Enter' &&
e.preventDefault(); }}
            onChange ={ (event): void => {
              setfindPatient(event.target.value.toLowerCase());
            }}
          />
        </Paper>
        <Button
          style={ { marginLeft: '20%', borderRadius: '10px' } }
          component="label"
          variant="contained"
          onClick ={ (): void => {
            setPatientCodeToOpen(0);
            dispatch(clearPatientDetails());
```

КОПЗ_2024

```
            dispatch(clearPatientLogData());
            setOpenPatientDetailModal(true);
          }}
          color="success"
          tabIndex={-1}
          startIcon={<PersonAddIcon />}
        >
          Новий
        </Button>
      </div>
      <div style={ { overflowY: 'scroll',scrollbarWidth: 'none', height:
'89%' } }>
        <Grid style={{ padding: '7px', opacity: '1' }} container
rowSpacing={2}
          columns={ { xs: 4, sm: 8, md: 10 } } >
          {patients.map((item, index) => (
            <Grid item xs={6} sm={4} md={2} key={index}>
              <Card key={index} sx={{ minHeight: 430, maxWidth: 285,
border:'5px solid #0d47a1'
                , borderRadius: '40px' }} >
                <CardActionArea>
                  <div style={ { backgroundColor:
'#90caf9',borderRadius:'15px 15px 0 0' } }>
                    <CardMedia
                      style={ { border: '5px solid white', borderRadius:
'80px',height: '120px', width: '120px'
                        , margin: 'auto' } }
                      component="img"
                      height="100%"
                      image = {item.sex.trim() === 'M'? male_img : female_img
}
                    />
                    <Avatar sx={{ position: 'absolute' ,bgcolor: 'red', top:
15, left: 50 }}>
                      {getPatientFullYears(item.dateBirth).split('
')[0]}</Avatar>
                  </div>
                  <CardContent sx={{ paddingX: '3px', paddingY: 0 }} >
                    <Typography align="center"  variant="subtitle2"
gutterBottom color="darkblue"
                      style={ { fontSize: 20, textAlign: 'center' } } sx={{
marginY: 0, height: 65 }}>
                      {item.lastName+' '+item.firstName+' '+item.middleName}
```

КбПЗ_2024

```
                    </Typography>
                    <Chip sx={ { marginBottom : 1, fontWeight: 'bold' } }
                      avatar={<Avatar alt="Birth" src={ birth_img } />}
                      label={ getFormattedDate(item.dateBirth, 'dd.MM.yyyy')
}
                      variant="outlined"
                    />
                    <div style={ { height: 60 ,borderRadius:
3,backgroundColor: '#1565c0', padding: 5 } }>
                      <span style={ { fontSize: 15, fontWeight: 'bold',
color: 'whitesmoke' } } >
                        { item.cityName
                        +(item.streetName.includes('.')?',':'
,вул.')+item.streetName
                        +(item.build.length>0?' буд.'+item.build:'')
                        +(item.flat.length>0?' кв.':'')+item.flat }
                      </span>
                    </div>
                    <div style={ { display: 'flex', justifyContent: 'center'
} }>
                      <Barcode textAlign={'center'} value={item.cardId}
height={30}   />
                    </div>
                  </CardContent>
                </CardActionArea>
                <CardActions>
                  <Button size="small" color="primary"
                  onClick={():void=>{
                    setPatientCodeToOpen(item.patientCode);
                    setOpenPatientDetailModal(true);
                  }
                  }>
                  Детальніше
                  </Button>
                </CardActions>
              </Card>
            </Grid>
          ))}
        </Grid>
      </div>
    </div>
  );
};
```

КОПІЗ_2024

```
export { RegisterDashboard };

//Модуль обліку даних лікарів
import HighlightOffIcon from '@mui/icons-material/HighlightOff';
import PersonAddIcon from '@mui/icons-material/PersonAdd';
import {
  Autocomplete, Avatar,
  CardActionArea,
  CardActions, Checkbox, Chip, FormControlLabel,
  Grid, InputAdornment,
  TextField,
  Typography,
} from '@mui/material';
import Button from '@mui/material/Button';
import Card from '@mui/material/Card';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import IconButton from '@mui/material/IconButton';
import doctorf_img from 'assets/img/doctorf.png';
import doctorm_img from 'assets/img/doctorm.png';
import card_image from 'assets/img/peception1.jpg';
import phone_img from 'assets/img/phone.png';
import { DoctorSpecialty } from 'common/types/register/patient/doctor-
specialty';
import { DoctorStuff } from 'common/types/register/patient/doctor-stuff';
import { useAppDispatch, useAppSelector } from 'hooks/hooks';
import { filterDoctorSpecialty } from 'modules/register/helper/get-doctor-
specialty';
import { DoctorCardModal } from 'modules/register/modal/doctor-card-modal';
import * as React from 'react';
import { FC, ReactElement, useEffect, useState } from 'react';
import {
  changeDoctorActive,
} from 'store/register/patient/actions';

import { getPatientFullYears } from '../helper/get-patient-full-years';

const DoctorStuffDashboard: FC = () => {
  const dispatch = useAppDispatch();

  const { doctorStuff , doctorSpecialty, doctorSpecialtyUnite }
    = useAppSelector(( { patient } ) => ({
      doctorStuff: patient.doctorStuff,
```

КПЗ-2024

```
      doctorSpecialty: patient.doctorSpecialty ,
      doctorSpecialtyUnite: patient.doctorSpecialtyUnite,
    }));

  useEffect(() => {
    setActiveChecker(true);
  }, []);

  const [findDoctorStuff , setFindDoctorStuff] = React.useState('');
  const [selectedSpecialtyId, setSelectedSpecialtyId] = React.useState(0);
  const [filteredDoctorStuff, setFilteredDoctorStuff] =
React.useState(doctorStuff);
  const [activeChecker, setActiveChecker] = React.useState(false);
  const [openDoctorCardModalModal, setOpenDoctorCardModal] = useState(false);
  const [doctorIdToOpen , setDoctorIdToOpen] = React.useState<number | null
>(null);

  useEffect(() => {
    filterDoctorStuff();
  }, [activeChecker, selectedSpecialtyId, findDoctorStuff ]);

  const handleChangeActive = (id:number ): void => {
    dispatch(changeDoctorActive(id));
    const newFilteredDoctorList = filteredDoctorStuff.map((i)=>{
      return i.id===id?{ ...i, active: !i.active }: i; });
    setFilteredDoctorStuff(newFilteredDoctorList);
  };

  const handleActiveChecker = (): void => {
    setActiveChecker(!activeChecker);
  };

  const filterDoctorStuff = (): void => {
    let filterOne : DoctorStuff[] = [];
    let filterTwo : DoctorStuff[] = [];
    let filterThird : DoctorStuff[] = [];

    if (findDoctorStuff.length>0){
      filterOne = doctorStuff.filter((item)=> item.lastName.toUpperCase()
        .includes(findDoctorStuff.trim().toUpperCase()));
    } else {
      filterOne = doctorStuff;
    }
```

КОПІЯ_2024

```
    if (activeChecker){
      filterTwo = filterOne.filter((item)=> item.active);
    } else {
      filterTwo = filterOne;
    }

    if (selectedSpecialtyId>0){
      const filteredDoctorSpecialtyUnite = doctorSpecialtyUnite
        .filter((item)=>item.specialtyId===selectedSpecialtyId)
        .map(function(item) { return item.doctorId; });
      filterThird = filterTwo.filter((item) =>filteredDoctorSpecialtyUnite
        .includes(item.id));
    } else {
      filterThird = filterTwo;
    }
    setFilteredDoctorStuff(filterThird);
  };

  const handleChangeSpecialty = (specialty:string ): void => {
    if(specialty===''){
      setSelectedSpecialtyId(0);
    }else{
      const filteredSpecialty =
doctorSpecialty.filter((item)=>item.specialty.includes(specialty))[0].id;
      setSelectedSpecialtyId(filteredSpecialty);
    }
  };

  return (
    <div style={{ backgroundImage:  `url(${card_image})`,
      backgroundPosition: 'center',
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      width: '84vw',
      height: '90vh' }}>
      <DoctorCardModal doctorId = { doctorIdToOpen }
        isOpen= { openDoctorCardModalModal }
        onClose={():void=>{
          setDoctorIdToOpen(null);
          filterDoctorStuff();
          setOpenDoctorCardModal(false);
        }}/>
```

КБП3_2024

```
        <div style={ { padding: '10px', display: 'flex',  marginBottom: 20 } }
>
          <div style = { { padding: 8, backgroundColor: 'white',
            marginLeft: 50, borderRadius: 5 } }>
            <TextField
              InputProps={{
                endAdornment: (
                  <InputAdornment position="start">
                    <IconButton type="button"
                      onClick={():void=>{
                        setFindDoctorStuff('');
                      }}
                      sx={{ p: '1px', transform: 'scale(1.2)' }}  >
                      <HighlightOffIcon />
                    </IconButton>
                  </InputAdornment>
                ),
              }}
              size="small"
              value={ findDoctorStuff }
              label="Доктор"
              variant="outlined"
              onChange ={ (event): void => {
                setFindDoctorStuff(event.target.value.toLowerCase());
              }}
            />
          </div>
          <div style = { { backgroundColor: 'white',
            marginLeft: 50, borderRadius: 5  } }>
            <Autocomplete
              disablePortal
              size="small"
              ListboxProps={{ style: { maxHeight: '12rem' } }}
              options={doctorSpecialty.map((item) => item.specialty)}
              sx={ { width: 300 , m: 1 } }
              onChange={(event , specialty: string | null): void =>{
                handleChangeSpecialty(specialty===null?'':specialty);
              }}
              renderInput={ ( params ):
              ReactElement => <TextField {...params} label="Спеціалізація" />}
            />
          </div>
          <div style = { { padding: 3, backgroundColor: 'white',
```

КОПІЯ_2024

```
          marginLeft: 50, borderRadius: 5 } }>
          <FormControlLabel style={ { width: 140 ,margin: 1, border:'1px
solid darkgrey', borderRadius: 5 } }
            onChange={(): void =>{
              handleActiveChecker();
            }}
            control={<Checkbox checked={ activeChecker } />} label="Працюючі"
/>
        </div>
        <Button
          style={ { marginLeft: '10%', borderRadius: '10px' } }
          component="label"
          variant="contained"
          color="success"
          tabIndex={-1}
          onClick={ ():void =>{
            setDoctorIdToOpen(null);
            setOpenDoctorCardModal(true);
          } }
          startIcon={<PersonAddIcon />}
        >
          Новий
        </Button>
      </div>
      <div style={ { overflowY: 'scroll',scrollbarWidth: 'none', height:
'85%' } }>
        <Grid style={{ padding: '7px', opacity: '1' }} container
rowSpacing={2}
          columns={ { xs: 4, sm: 8, md: 10 } } >
          {filteredDoctorStuff.map((item, index) => (
            <Grid item xs={6} sm={4} md={2} key={index}>
              <Card
                key={index}
                sx={{ height: 400, maxWidth: 250, border:'5px solid
#1565c0',borderRadius:'25px' }}>
                <CardActionArea >
                  <div style={ { backgroundColor:
'#90caf9',borderRadius:'15px 15px 0 0' } }>
                    <CardMedia
                      style={ item.active? { border: '5px solid white',
borderRadius: '80px',height: '120px'
                        , width: '120px', margin: 'auto'  }:
```

КОПЗ_2024

```
                        { border: '5px solid white', borderRadius:
'80px',height: '120px', width: '120px'
                          , margin: 'auto',filter: 'grayscale(100%)',
opacity: '50%' } }
                      component="img"
                      onClick={ ():void =>{
                        handleChangeActive(item.id);
                      } }
                      height="100%"
                      image= { item.sex==='M'?doctorm_img:doctorf_img }
                    />
                  </div>
                  <CardContent sx={{ paddingX: '5px', paddingY: 0 }} >
                    <Typography sx={{ marginY: 0 }} variant="subtitle2"
style={ { fontSize: 20, textAlign: 'center' } }
                      gutterBottom color="darkblue">
                      {item.lastName+' '+item.firstName }
                    </Typography>
                    <Typography sx={{ marginY: 0 }} variant="subtitle2"
                      style={ { fontWeight: 'bold',fontSize: 20, textAlign:
'center' } }
                      gutterBottom >{getPatientFullYears(item.dateBirth)}
                    </Typography>
                    <Typography variant="subtitle2" style={ { fontWeight:
'bold',fontSize: 20, textAlign: 'center' } }
                      sx={{ marginY: 0 }}
                      gutterBottom color="red">Кабінет
                      {' '+item.cabinet }
                    </Typography>
                    <Chip sx={ { marginBottom : 1, fontWeight: 'bold' } }
                      avatar={<Avatar alt="Phone" src={ phone_img } />}
                      label={ item.phone }
                      variant="outlined"
                    />
                    <div style={ { height: 100 ,borderRadius:
3,backgroundColor: '#2196f3', padding: 5 } }>

{filterDoctorSpecialty(item.id,doctorSpecialtyUnite,doctorSpecialty)
                        .map((i: DoctorSpecialty)=>(
                          <span key={ i.id } style={ { fontSize: 15,
fontWeight: 'bold', color: 'whitesmoke' } } >
                            {i.specialty+' ' }
                          </span>
```

КОПІЯ_2024

```
                    ))}
                  </div>
                </CardContent>
              </CardActionArea>
              <CardActions sx={{ paddingY: 0 }}>
                <Button size="small" color="primary"
                  onClick={ ():void =>{
                    setDoctorIdToOpen(item.id);
                    setOpenDoctorCardModal(true);
                  } }>
                  Детальніше
                </Button>
              </CardActions>
            </Card>
          </Grid>
        ))}
      </Grid>
    </div>
  </div>
  );
};
export { DoctorStuffDashboard };

//Модуль виконаних послуг
import HighlightOffIcon from '@mui/icons-material/HighlightOff';
import PrintIcon from '@mui/icons-material/Print';
import WheelchairPickupIcon from '@mui/icons-material/WheelchairPickup';
import {
  Autocomplete,
  FormControl,
  InputAdornment, InputLabel, MenuItem,
  Paper, Select,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead, TablePagination,
  TableRow,
  TextField, Tooltip,
} from '@mui/material';
import Button from '@mui/material/Button';
import IconButton from '@mui/material/IconButton';
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs';
```

КОПЗ_2024

```
import { DatePicker } from '@mui/x-date-pickers/DatePicker';
import { DemoContainer } from '@mui/x-date-pickers/internals/demo';
import { LocalizationProvider } from '@mui/x-date-
pickers/LocalizationProvider';
import card_image from 'assets/img/peception1.jpg';
import { ArcElement,Chart } from 'chart.js';
import { LogDataPeriodDTO } from 'common/types/register/patient/log-data-
period-dto';
import { PatientLogdata } from 'common/types/register/patient/patient-
logdata';
import dayjs, { Dayjs } from 'dayjs';
import { getFormattedDate } from 'helpers/date/get-formatted-date/get-
formatted-date';
import { useAppDispatch, useAppSelector } from 'hooks/hooks';
import { OperCategory } from 'modules/register/common/oper-category.enum';
import { getIconColor } from 'modules/register/helper/get-icon-color';
import { getLogDataTotalSum } from 'modules/register/helper/get-log-data-
total-sum';
import { getNewIconElement } from 'modules/register/helper/get-new-icon-
element';
import { getPatientSum } from 'modules/register/helper/get-patient-sum';
import { PatientDetailModal } from 'modules/register/modal/patient-detail-
modal';
import * as React from 'react';
import { FC, ReactElement, useEffect, useState } from 'react';
import { getLogDataForPeriod, getPatient, getRequisites } from
'store/register/patient/actions';
import { clearFindPatient, clearPatientDetails, clearPatientLogData } from
'store/register/patient/reducer';

import { BreafLogData } from '../../../common/types/register/patient/breaf-
log-data';
import { PatientResponse } from
'../../../common/types/register/patient/patient-response';
import { getEmptyBriefLogData } from '../helper/get-empty-briefLogData';
import { getEmptyPatientResponce } from '../helper/get-empty-patient-
responce';
import { AddDiagnosisModal } from '../modal/add-diagnosis-modal';
import { PatientOneProcedurePrint } from '../modal/patient-one-proclist-
print';
Chart.register(ArcElement);
const RegisterMedicalProceduresLogDashboard: FC = () => {
  const dispatch = useAppDispatch();
```

КОПІЯ 2024

```
  const { logDataForPeriod, doctorSpecialty, doctorStuff }
    = useAppSelector(( { patient } ) => ({
      logDataForPeriod: patient.logDataForPeriod,
      doctorSpecialty: patient.doctorSpecialty,
      doctorStuff: patient.doctorStuff,
    }));
  const [page, setPage] = React.useState(0);
  const [rowsPerPage, setRowsPerPage] = React.useState(20);
  const [rowsInPage, setRowsInPage] = React.useState<PatientLogdata[]>([]);
  const [findText, setFindText] = React.useState('');
  const [endDay, setEndDay] = React.useState<Dayjs| null>(dayjs(Date.now()));
  const [startDay, setStartDay] = React.useState<Dayjs|
null>(dayjs(Date.now()));
  const [specialtyLabel, setSpecialtyLabel] = React.useState('');
  const [findSpecialtyId, setFindSpecialtyId] = React.useState(0);
  const [findDoctorStuffId, setFindDoctorStuffId] = React.useState(0);
  const [doctorStuffLabel, setDoctorStuffLabel] = React.useState('');
  const [paymentTypeFilter, setPaymentTypeFilter] = useState('ALL');
  const [openPatientDetailModal, setOpenPatientDetailModal] =
useState(false);
  const [openAddDiagnosisModal, setOpenAddDiagnosisModal] = useState(false);
  const [patientCodeToOpen , setPatientCodeToOpen] = React.useState(0);
  const [procedureCodeToDiagnosisOpen , setProcedureCodeToDiagnosisOpen] =
React.useState(0);
  const [briefDataToPrint, setBriefDataToPrint] =
React.useState<BreafLogData|null>(null);
  const [patientToPrint, setPatientToPrint] =
React.useState<PatientResponse|null>(null);
  const [openPatientOneProceduresPrint, setPatientOneProceduresPrint] =
useState(false);
  useEffect(() => {
    const payload: LogDataPeriodDTO = { text: '',startDate:new
Date(),endDate:new Date(), specialtyId: 0, doctorId: 0,
      paymentType: paymentTypeFilter };
    dispatch(getLogDataForPeriod(payload));
    dispatch(getRequisites());
  }, []);
  useEffect(() => {
    setPage(0);
    setRowsInPage(logDataForPeriod.slice(0, rowsPerPage));
  }, [logDataForPeriod]);
  useEffect(() => {
```

КОПИЯ 2024

```
    if(patientToPrint!==null && briefDataToPrint!==null)
setPatientOneProceduresPrint(true);
  }, [patientToPrint,briefDataToPrint]);
  const handleFilter = (): void => {
    const payload: LogDataPeriodDTO = { text: findText, startDate:
startDay===null?new Date(): startDay.toDate()
      , endDate: endDay===null?new Date(): endDay.toDate(), specialtyId:
findSpecialtyId
      , doctorId: findDoctorStuffId, paymentType: paymentTypeFilter };
    dispatch(getLogDataForPeriod(payload));
  };
  const prepareDataToPrint = (patientLog: PatientLogdata): void => {
    setBriefDataToPrint( {
      id: patientLog.id,
      medicalProcedure: patientLog.medicalProcedure,
      doctorStuff: patientLog.doctorStuff,
      specialty: patientLog.doctorSpecialty,
      guidingDoctor: patientLog.guidingDoctor,
      enterprise: patientLog.guidingCompany,
      operationCat: patientLog.operCategory,
      kol: patientLog.kol,
      price: patientLog.payment,
    });
  };
  const preparePatientToPrint = (patientLog: PatientLogdata): void => {
    const date: string[]=patientLog.patientName.split('(')[1].replace(')','')
      .replaceAll('.','-').split('-');
    setPatientToPrint( {
      id: patientLog.id ,
      cardId: '',
      patientCode: patientLog.patientCode,
      edrpou: 0,
      dateCreated: patientLog.dateTime,
      firstName: '',
      middleName: '',
      lastName: patientLog.patientName.split('(')[0],
      city: 0,
      street: 0,
      sex: '',
      passportId: '',
      dateBirth: new Date(date[2]+'-'+date[1]+'-
'+date[0]+'T22:00:00.000+00:00'),
      cityName: '',
```

КбП3_2024

```
      streetName: '',
      korp: '',
      build: '',
      flat: '',
      phone: '',
      workplace: '',
      pilgSvidot: '',
    });
  };
  const handleRowsInPage = ( pageNumber: number): void => {
    const newRowArray = logDataForPeriod.slice(pageNumber*rowsPerPage,
(pageNumber+1)*rowsPerPage);
    setRowsInPage(newRowArray);
  };
  const handleChangeSpecialty = (specialty:string ): void => {
    setSpecialtyLabel(specialty);
    specialty===''?
      setFindSpecialtyId(0):

setFindSpecialtyId(doctorSpecialty.filter((item)=>item.specialty===specialty)
[0].id);
  };
  const handlePatientOpen = (patientCode:number): void => {
    dispatch(getPatient(patientCode));
    setOpenPatientDetailModal(true);
  };
  const handleDiagnosisOpen = (): void => {
    setOpenAddDiagnosisModal(true);
  };
  const handleSetFilteredfDoctorStuffId = (doctor:string ): void => {
    setDoctorStuffLabel(doctor);
    doctor===''?
      setFindDoctorStuffId(0):

setFindDoctorStuffId(doctorStuff.filter((item)=>item.lastName===doctor)[0].id
);
  };
  const handleChangePage = (_event: unknown, newPage: number): void => {
    setPage(newPage);
    handleRowsInPage(newPage);
  };
  const handleChangeRowsPerPage = (event:
React.ChangeEvent<HTMLInputElement>): void => {
```

КОПІЯ 2024

```
    setRowsPerPage(parseInt(event.target.value, 10));
    setPage(0);
  };

  return (
    <div style={{ backgroundImage:  `url(${card_image})`,
      backgroundPosition: 'center',
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      width: '85vw',
      height: '85vh' }}>
      <PatientDetailModal patientCode = { patientCodeToOpen }
        isOpen= { openPatientDetailModal }
        onClose={():void=>{
          dispatch(clearFindPatient());
          dispatch(clearPatientDetails());
          dispatch(clearPatientLogData());
          setOpenPatientDetailModal(false); }}/>
      <PatientOneProcedurePrint dataToPrint = {
briefDataToPrint===null?getEmptyBriefLogData():briefDataToPrint }
        isOpen= { openPatientOneProceduresPrint } patient={
patientToPrint===null?
          getEmptyPatientResponce():patientToPrint }
        onClose={():void=>{
          setPatientToPrint(null);
          setBriefDataToPrint(null);
          setPatientOneProceduresPrint(false);
        }} />
      <AddDiagnosisModal procedureId= { procedureCodeToDiagnosisOpen }
        isOpen= { openAddDiagnosisModal }
        onClose={():void=>{
          setProcedureCodeToDiagnosisOpen(0);
          setOpenAddDiagnosisModal(false); }}/>
      <div style={ { padding: 1, display: 'flex', justifyContent: 'center',
opacity: '0.95', height: '100%' } } >
        <Paper
          component="form"
          style={ { flexDirection: 'column' } }
          sx={{ p: '5px', display: 'flex', textAlign: 'center', width: '95%'
}}
        >
          <h2 style={ { color: '#0d47a1' } }>ЖУРНАЛ ПОСЛУГ</h2>
```

КОПІЯ 2024

```
          <div style={ { display: 'flex', border:'1px solid darkblue',
borderRadius: 5, margin: 5,
            marginBottom: 5 } }>
            <div style={ { display: 'flex', marginTop: 30 }}>
              <TextField
                InputProps={{
                  endAdornment: (
                    <InputAdornment position="start">
                      <IconButton type="button" sx={{ transform: 'scale(1.2)'
}}
                        onClick= { (): void => {
                          setFindText('');
                        }}>
                        <HighlightOffIcon  />
                      </IconButton>
                    </InputAdornment>
                  ),
                }}
                sx={ { margin: '8px', marginRight: 2 } }
                size="medium"
                style={ findText.length>2?{ color: 'red' } : { color: 'black'
} }
                value={ findText }
                label="Пацієнт(від 3-х симв.)"
                variant="outlined"
                onChange ={ (event): void => {
                  setFindText(event.target.value.toLowerCase());
                }}
              />
              <LocalizationProvider dateAdapter={AdapterDayjs}
adapterLocale="en-gb">
                <DemoContainer components={['DatePicker', 'DatePicker']}>
                  <DatePicker
                    label="Початкова дата"
                    value={startDay}
                    onChange={(newValue):void => setStartDay(newValue)}
                  />
                  <DatePicker
                    label="Кінцева дата"
                    value={endDay}
                    onChange={(newValue):void => setEndDay(newValue)}
                  />
                </DemoContainer>
```

КбТЗ_2024

```
            </LocalizationProvider>
            <Autocomplete
              disablePortal
              size="medium"
              value = {specialtyLabel}
              ListboxProps={{ style: { maxHeight: '12rem' } }}
              options={doctorSpecialty.map((item) => item.specialty)}
              sx={ { width: 250 , m: 1 } }
              onChange={(_event , specialty: string | null): void =>{
                handleChangeSpecialty(specialty===null?'':specialty);
              }}
              renderInput={ ( params ):
              ReactElement => <TextField {...params} label="Спеціалізація"
/>}
            />
            <Autocomplete
              disablePortal
              style={ { marginLeft: 5 } }
              size="medium"
              value={doctorStuffLabel}
              ListboxProps={{ style: { maxHeight: '12rem' } }}
              options={doctorStuff.map((item) => item.lastName)}
              sx={ { width: 250 , m: 1 } }
              onChange={(_event , doctorStuff: string | null): void =>{

handleSetFilteredfDoctorStuffId(doctorStuff===null?'':doctorStuff);
              }}
              renderInput={ ( params ):
              ReactElement => <TextField {...params} label="Лікар" />}
            />
            <FormControl sx={ { paddingLeft: '15px', paddingTop: '8px',
width: '20ch' } }>
              <InputLabel sx={ { paddingLeft: '18px', paddingTop: '10px' }
} id="payment">Оплата</InputLabel>
              <Select
                size={'medium'}
                id="payment_select"
                label="Оплата"
                onChange={(e):void=>{
                  setPaymentTypeFilter(String(e.target.value));
                }}
              >
                <MenuItem value="ALL">Усі</MenuItem>
```

КбПЗ_2024

```
                {OperCategory.map((option) => (
                  <MenuItem key={option.cod}
value={option.cod}>{option.name}</MenuItem>
                ))}
              </Select>
            </FormControl>
            <Button
              style={{ margin: '8px' , height: 55 }}
              size="large"
              variant="contained"
              color = "primary"
              onClick={handleFilter}
            > Знайти </Button>
          </div>
        </div>
        <div style={ { display: 'flex', flexDirection: 'column', textAlign:
'left' } }>
          <TableContainer component={Paper} style={ { height: '68vh' } }>
            <Table sx={{ minWidth: 650 }} size="small" >
              <TableHead>
                <TableRow  style={ { backgroundColor: '#64b5f6' } }>
                  <TableCell align="center">№</TableCell>
                  <TableCell align="center">Друк</TableCell>
                  <TableCell align="center">Дата</TableCell>
                  <TableCell >Пацієнт</TableCell>
                  <TableCell >Спеціалізація</TableCell>
                  <TableCell >Виконавець</TableCell>
                  <TableCell >Процедура</TableCell>
                  <TableCell >Направив</TableCell>
                  <TableCell ></TableCell>
                  <TableCell align="right">Сума(грн)</TableCell>
                </TableRow>
              </TableHead>
              <TableBody style={ { height: '100%' } }>
                {rowsInPage.map((row, index) => (
                  <TableRow
                    key={index}
                    sx={{ '&:last-child td, &:last-child th': { border: 0 }
}}
                  >
                    <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
```

КПЗ 2024

```
                        : { backgroundColor: 'white' } } component="th"
scope="row" >
                        {row.id}
                      </TableCell>
                      <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
                        : { backgroundColor: 'white' } } component="th"
scope="row" >
                        <IconButton type="button" sx={{ transform:
'scale(1.2)', color: 'green' }}
                          onClick={():void=>{
                            prepareDataToPrint(row);
                            preparePatientToPrint(row);
                          }}
                        >
                          <PrintIcon  />
                        </IconButton>
                      </TableCell>
                      <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
                        : { backgroundColor: 'white' } } align="center"
                      >{getFormattedDate(row.dateTime, 'dd-MM-yyyy HH:mm') }
                      </TableCell>
                      <Tooltip title={'Відкрити картку пацієнта
'+row.patientName} componentsProps={{
                        tooltip: { sx: { bgcolor: 'black' } } }}>
                        <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
                          : { backgroundColor: 'white' } } align="left"
                        sx={ { cursor: 'pointer' } }
                        onClick={(): void =>{
                          setPatientCodeToOpen(row.patientCode);
                          handlePatientOpen(row.patientCode);
                        }}
                        >{row.patientName}</TableCell>
                      </Tooltip>
                      <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
                        : { backgroundColor: 'white' } }
align="left">{row.doctorSpecialty}</TableCell>
                      <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
```

КОПІЯ_2024

```
                        : { backgroundColor: 'white' } }
align="left">{row.doctorStuff}</TableCell>
                      <Tooltip title={'Заповнити діагноз послуги
'+row.medicalProcedure}
                        componentsProps={{ tooltip: { sx: { bgcolor:
'#0d47a1' } } }}>
                        <TableCell sx={ { cursor: 'pointer' } } style={
index%2===1? { backgroundColor: '#e3f2fd' }
                          : { backgroundColor: 'white' } }
                        align="left"
                        onClick={(): void =>{
                          setProcedureCodeToDiagnosisOpen(row.id);
                          handleDiagnosisOpen();
                        }}
                        >{row.medicalProcedure}</TableCell>
                      </Tooltip>
                      <TableCell style={ index%2===1? { backgroundColor:
'#e3f2fd' }
                        : { backgroundColor: 'white' } }
align="left">{row.guidingDoctor}</TableCell>
                      <TableCell sx={ { color: getIconColor(row.operCategory)
} }
                        style={ index%2===1? { backgroundColor: '#e3f2fd' }
                          : { backgroundColor: 'white'  } }
                        align="center">{ getNewIconElement[row.operCategory]
}
                      </TableCell>
                      <TableCell style={ index%2===1? { fontWeight:
'bold',backgroundColor: '#e3f2fd' }
                        : { fontWeight: 'bold', backgroundColor: 'white' } }
align="right">{row.payment===0?' '
                          : row.payment }</TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
            </TableContainer>
            <div style={ { display: 'flex' } }>
              <TablePagination
                rowsPerPageOptions={[ 10, 20]}
                component="div"
                count={logDataForPeriod.length}
                rowsPerPage={rowsPerPage}
```

КбПЗ-2024

```
                page={page}
                onPageChange={handleChangePage}
                onRowsPerPageChange={handleChangeRowsPerPage}
              />
              <span style={ { fontSize: 20,fontWeight: 'bold', margin: 10,
color: 'darkblue' } }>
                <WheelchairPickupIcon/>{'
'+getPatientSum(logDataForPeriod)}</span>
              <span style={ { fontSize: 20,fontWeight: 'bold', margin: 10,
color: 'darkblue' } }>
                СУМА</span>
              <span style={ { fontSize: 20,fontWeight: 'bold', margin: 10,
color: 'darkblue' } }>
                {getLogDataTotalSum(logDataForPeriod)+' '}грн</span>
            </div>
          </div>
        </Paper>
      </div>
    </div>
  );
};
export { RegisterMedicalProceduresLogDashboard };

//Модальний компонент даних пацієнта
import Brightness1Icon from '@mui/icons-material/Brightness1';
import CancelIcon from '@mui/icons-material/Cancel';
import ChangeCircleIcon from '@mui/icons-material/ChangeCircle';
import EditIcon from '@mui/icons-material/Edit';
import { TabContext, TabList, TabPanel } from '@mui/lab';
import {
  Autocomplete,
  Box,
  FormControlLabel,
  Paper,
  Radio,
  RadioGroup,
  Tab, Table, TableBody, TableCell,
  TableContainer, TableHead, TablePagination, TableRow,
  TextField,
} from '@mui/material';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
```

КПЗ_2024

```
import DialogContent from '@mui/material/DialogContent';
import DialogTitle from '@mui/material/DialogTitle';
import IconButton from '@mui/material/IconButton';
import { DataGrid, GridColDef } from '@mui/x-data-grid';
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs';
import { DatePicker } from '@mui/x-date-pickers/DatePicker';
import { DemoContainer } from '@mui/x-date-pickers/internals/demo';
import { LocalizationProvider } from '@mui/x-date-
pickers/LocalizationProvider';
import { PatientResponse } from 'common/types/register/patient/patient-
response';
import { CityStreet, PatientLogdata } from 'common/types/types';
import dayjs, { Dayjs } from 'dayjs';
import { getFormattedDate } from 'helpers/date/get-formatted-date/get-
formatted-date';
import { useAppDispatch, useAppSelector } from 'hooks/hooks';
import { getStatusName } from 'modules/register/helper/get-status-name';
import { EditCityStreetModal } from 'modules/register/modal/city-street-edit-
modal';
import { DoctorInvoiceCancelModal } from 'modules/register/modal/doctor-
invoice-cancel-modal';
import { InvoiceDoctorModal } from 'modules/register/modal/patient-invoice-
doctor-modal';
import { CallDoctorModal } from 'modules/register/modal/patient-visit-doctor-
modal';
import React, { FC, ReactElement, useEffect, useState } from 'react';
import { useForm } from 'react-hook-form';
import {
  createPatient,
  getEnterprise,
  getGuidingCompany,
  getGuidingDoctor,
  getPatientDetail, getPatientDoctorInvoice,
  getPatientLogData, saveEditedCity, saveEditedStreet,
  updatePatient,
} from 'store/register/patient/actions';
import { DataStatus } from '../../../common/enums/app/data-status.enum';
import { Spinner } from '../../../components/common/spinner/spinner';
import { clearPatientDoctorInvoice } from
'../../../store/register/patient/reducer';
import { getIconColor } from '../helper/get-icon-color';
import { getNewIconElement } from '../helper/get-new-icon-element';
```

КОПЗ_2024

```
type Props = {
  isOpen: boolean;
  onClose: () => void;
  patientCode: number ;
};

const PatientDetailModal: FC<Props> = ({ isOpen, onClose, patientCode  }) =>
{
  const dispatch = useAppDispatch();
  const { patientDetails, patientLogData, street, city
    ,patientDoctorInvoice ,patient, patientGetLogDataDataStatus }
    = useAppSelector(( { patient } ) => ({
      patientDetails: patient.patientDetail,
      street: patient.street,
      patientGetLogDataDataStatus: patient.patientGetLogDataStatus,
      patientDoctorInvoice : patient.patientDoctorInvoice,
      city: patient.city,
      patient: patient.patientFind.filter((item)=> item.patientCode ===
patientCode)[0],
      patientLogData: patient.patientLogData,
    }));
  const isLoading = patientGetLogDataDataStatus === DataStatus.PENDING;
  const { register , handleSubmit,reset } =
    useForm< PatientResponse >();
  const [birthDay, setBirthDay] = React.useState<Dayjs>(dayjs(Date.now()));
  const [patientEditState, setPatientEditState]
    = React.useState(patientDetails===null);
  const [cityId, setCityId] = React.useState(0);
  const [streetId, setStreetId] = React.useState(0);
  const [forEditId, setForEditId] = React.useState(0);
  const [doctorInvoiceId, setDoctorInvoiceId] = React.useState(0);
  const [forEditName, setForEditName] = React.useState('');
  const [streetLabel, setStreetLabel] = React.useState('');
  const [patientSex, setPatientSex] = React.useState('');
  const [cityLabel, setCityLabel] = React.useState('');
  const [openCallDoctorModal, setOpenCallDoctorModal] = useState(false);
  const [value, setValue] = React.useState('1');
  const [openEditCityStreetModal, setOpenEditCityStreetModal] =
useState(false);
  const [openInvoiceDoctorModal, setOpenInvoiceDoctorModal] =
useState(false);
  const [openDoctorInvoiceCancelModal, setOpenDoctorInvoiceCancelModal] =
useState(false);
```

КОПІЯ 2024

```
  const [editCityStreetMode, setEditCityStreetMode] = useState( '' );
  const [page, setPage] = React.useState(0);
  const [rowsPerPage, setRowsPerPage] = React.useState(10);
  const [rowsInPage, setRowsInPage] = React.useState<PatientLogdata[]>([]);
  const handleChange = (event: React.SyntheticEvent, newValue: string):void
=> {
    setValue(newValue);
  };
  const columnsInvoice: GridColDef[] = [
    { field: 'payed', headerName: '', width: 40, renderCell: (params) =>
      <Brightness1Icon style={ params.row.status==='NEW' ||
params.row.status==='DONE'?
        { color: 'green' } : params.row.status==='CANCEL'?{  color: 'red' } :
{  color: 'orange' } }/> },
    { field: 'status', headerName: 'Статус', width: 130, renderCell: (params)
=>
      getStatusName(params.row.status)  },
    { field: 'visitDate', headerName: 'Дата/час', width: 200, renderCell:
(params) =>
      getFormattedDate(params.row.visitDate, 'dd-MM-yyyy HH:mm') },
    { field: 'doctorName', headerName: 'Виконавець', width: 200 },
    { field: 'specialtyName', headerName: 'Спеціалізація',  width: 200 },
    { field: 'description', headerName: 'Примітки', width: 400 },
    { field: 'toEdit', headerName: 'Перенести', width: 90, renderCell:
(params) =>
      <IconButton color="primary" disabled={params.row.status==='DONE' ||
params.row.status==='CANCEL'
      || params.row.status==='TRANSF' }
      onClick={():void =>{
        setDoctorInvoiceId(params.row.id);
        setOpenInvoiceDoctorModal(true);
      }}>
        <ChangeCircleIcon />
      </IconButton> },
    { field: 'toCancel', headerName: 'Відміна', width: 70, renderCell:
(params) =>
      <IconButton color="error" disabled={params.row.status==='DONE' ||
params.row.status==='CANCEL'
        || params.row.status==='TRANSF' }
      onClick={():void =>{
        setDoctorInvoiceId(params.row.id);
        setOpenDoctorInvoiceCancelModal(true);
      }
```

КОПЗ_2024

```
      }>
        < CancelIcon />
      </IconButton> },
  ];
  useEffect(() => {
    if( patientCode>0 ) {
      dispatch(getPatientDetail(patientCode));
      dispatch(getPatientLogData(patient?.patientCode ?? 0));
      dispatch(getPatientDoctorInvoice(patientCode));
    }
    dispatch(getGuidingDoctor());
    dispatch(getGuidingCompany());
    dispatch(getEnterprise());
  }, []);
  useEffect(() => {
    setPage(0);
    setRowsInPage(patientLogData.slice(0, rowsPerPage));
  }, [patientLogData]);
  const handleRowsInPage = ( pageNumber: number): void => {
    const newRowArray = patientLogData.slice(pageNumber*rowsPerPage,
(pageNumber+1)*rowsPerPage);
    setRowsInPage(newRowArray);
  };
  const handleChangePage = (event: unknown, newPage: number): void => {
    setPage(newPage);
    handleRowsInPage(newPage);
  };
  const handleChangeRowsPerPage = (event:
React.ChangeEvent<HTMLInputElement>): void => {
    setRowsPerPage(parseInt(event.target.value, 10));
    setPage(0);
  };
  useEffect(() => {
    setPatientEditState(patientCode===0);
    if( patientCode>0 ){
      dispatch(getPatientDoctorInvoice(patientCode));
      setBirthDay(dayjs(patient?.dateBirth?? Date()));
      setStreetId(patient?.street?? 1);
      setCityId(patient?.city?? 1);
      setCityLabel(patient?.cityName?? ' ');
      setPatientSex(patient?.sex?? 'M');
      setStreetLabel(patient?.streetName?? ' ');
    } else {
```

КбГЗ_2024

```
      setBirthDay(dayjs(new Date()));
      setCityId(1);
      setStreetId(1);
      setCityLabel('немає');
      setPatientSex('M');
      setStreetLabel('нет');
    }
    setBirthDay(dayjs(patient?.dateBirth));
    reset({ firstName: patient?.firstName ?? ' ',
      lastName: patient?.lastName ?? ' ' ,
      middleName: patient?.middleName?? ' ',
      cityName: patient?.cityName?? ' ',
      city: patient?.city?? 1,
      street: patient?.street?? 1,
      cardId: patient?.cardId?? ' ',
      streetName: patient?.streetName?? ' ',
      build: patient?.build?? ' ',
      korp: patient?.korp?? ' ',
      sex: patient?.sex?? 'M',
      passportId: patientDetails?.passportId?? ' ',
      flat: patient?.flat?? ' ',
      dateBirth: patient?.dateBirth?? Date(),
      phone: patientDetails?.phone?? ' ',
      workplace: patientDetails?.workplace?? ' ',
      pilgSvidot: patientDetails?.pilgSvidot?? ' ',
    } );
  }, [dispatch,reset,patient, patientDetails?.phone,
patientDetails?.pilgSvidot, patientDetails?.workplace]);
  const handleClose = (): void => {
    // reset();
    clearStates();
    onClose();
  };
  const editCityStreetFunction = (item: CityStreet): void => {
    if(editCityStreetMode==='street') {
      dispatch(saveEditedStreet(item));
      setEditCityStreetMode('');
    } else if (editCityStreetMode==='city') {
      dispatch(saveEditedCity(item));
      setEditCityStreetMode('');
    }
  };
  const clearStates = (): void => {
```

КОПІЯ 2024

```
    dispatch(clearPatientDoctorInvoice());
    setBirthDay(dayjs(Date.now()));
    setCityId(0);
    setStreetId(0);
    setValue('1');
  };
  const getLocateId = (array: CityStreet[], item: string): number => {
    return array.filter((e) => e.name === item)[0].id;
  };
  const onSubmit = (data: PatientResponse): void => {
    if (patientCode===0) {
      const payload = { ...data, id: null,
        dateBirth: birthDay.toDate(), sex: patientSex, city: cityId, street:
streetId };
      dispatch(createPatient(payload)).unwrap().then(() => {
        handleClose();
      });
    }else{
      if (patientEditState) {
        const payload: PatientResponse = { ...data, id: patient.id, city:
cityId, street: streetId
          , dateBirth: birthDay.toDate(), sex: patientSex };
        dispatch(updatePatient(payload));
      }  }
    setPatientEditState(!patientEditState);
  };
  return ( <Dialog open={isOpen} maxWidth="xl">
    <DoctorInvoiceCancelModal  invoice={ doctorInvoiceId } isOpen= {
openDoctorInvoiceCancelModal }
      onClose={():void=>{
        setOpenDoctorInvoiceCancelModal(false);
      }}/>
    <EditCityStreetModal  nameValue={ forEditName } isOpen= {
openEditCityStreetModal }
      customFunction={( name: string ):void=>{
        editCityStreetFunction({ id:forEditId,name } );
      }}
      onClose={():void=>{
        setOpenEditCityStreetModal(false);
      }}/>
    <InvoiceDoctorModal  patientCode={ patient?.patientCode?? 0 } invoiceId =
{doctorInvoiceId}
      isOpen= { openInvoiceDoctorModal }
```

КПЗ_2024

```
      onClose={():void=>{
        setDoctorInvoiceId(0);
        setOpenInvoiceDoctorModal(false);
      }}/>
    <form  onSubmit={handleSubmit( onSubmit )} >
      <DialogTitle style={ { textAlign: 'center', color: '#0d47a1' } }>Картка
пацієнта</DialogTitle>
      <DialogContent style={{ padding: 1 }} >
        <div style={ { display: 'block', border:'2px solid #0d47a1',
borderRadius: 4, margin: 5 } }>
          <div style={ { display: 'inline-flex' }}>
            <TextField sx={{ m: 1, width: '30ch','& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
              'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'}
            disabled={!patientEditState}
            label="Призвіще" variant="outlined" {...register('lastName') }
            />
            <TextField sx={{ m:1, width: '15ch','& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                  'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'}
            disabled={ !patientEditState }
            label="Ім'я" variant="outlined" {...register('firstName') }
            />
            <TextField sx={{ m:1, width: '25ch','& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                  'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'}
            disabled={ !patientEditState }
            label="По-батькові" variant="outlined" {...register('middleName')
}
            />
            <LocalizationProvider dateAdapter={AdapterDayjs}
adapterLocale="en-gb">
              <DemoContainer sx={{ marginRight: 1, marginLeft: 1 }}
                components={['DatePicker']} >
                <DatePicker
                  disabled={ !patientEditState }
```

КбПЗ 2024

```
                sx={{ '& .MuiInputBase-input.Mui-disabled':{
WebkitTextFillColor: 'rgba(0, 0, 0, 0.7)' } }}
                label="Дата народження"
                slotProps={{ textField: { size: 'small' } }}
                value={birthDay}
                onChange={(newValue):void =>
                  setBirthDay(newValue===null?dayjs(Date.now()):newValue)}
              />
            </DemoContainer>
          </LocalizationProvider>
          <RadioGroup
            row
            value= { patientSex }
            onChange={(event, value):void =>{
              setPatientSex(value); }}
          >
            <FormControlLabel disabled={ !patientEditState } value="M"
control={<Radio />} label="Чол" />
            <FormControlLabel disabled={ !patientEditState } value="F"
control={<Radio />} label="Жін" />
          </RadioGroup>
          <Button sx={{ m:1, marginLeft: 10,marginRight: 10, width: 150 }}
size="small"
            style={ patientEditState? { backgroundColor: '#1565c0' }
              :{ backgroundColor: '#1565c0' } } variant="contained"
type="submit"
          >{patientEditState?'Зберегти':'Редагувати'}</Button>
        </div>
        <div style={ { display: 'flex' } }>
          <Autocomplete
            disablePortal
            size="small"
            disabled={ !patientEditState }
            ListboxProps={{ style: { maxHeight: '12rem' } }}
            disableClearable= { true }
            value={ cityLabel }
            options={city.map((item) => item.name)}
            sx={ { width: 250 , m: 1, marginLeft: 3, '& .MuiInputBase-
input.Mui-disabled':{ WebkitTextFillColor:
                  'rgba(0, 0, 0, 0.6)' } } }
            onChange={(_event , name: string | null): void =>{
              setCityLabel(name===null?'':name);
              setCityId(getLocateId(city,name===null?'':name));
```

КбТЗ_2024

```
        }}
        renderInput={ ( params ):
          ReactElement => <TextField {...params} label="Місто" />}
      />
      <IconButton color="primary"
        disabled={ !patientEditState }
        onClick={(): void =>{
          setEditCityStreetMode('city');
          setForEditId( cityId );
          setForEditName(cityLabel);
          setOpenEditCityStreetModal(true);
        }}>
        <EditIcon />
      </IconButton>
      <Autocomplete
        disablePortal
        size="small"
        disabled={ !patientEditState }
        ListboxProps={{ style: { maxHeight: '12rem' } }}
        disableClearable= { true }
        value={ streetLabel }
        options={street.map((item) => item.name)}
        sx={ { width: 250 , m: 1, marginLeft: 5, '& .MuiInputBase-
input.Mui-disabled':{ WebkitTextFillColor:
          'rgba(0, 0, 0, 0.6)' } } }
        onChange={(event , name: string | null): void =>{
          setStreetLabel(name===null?'':name);
          setStreetId(getLocateId(street,name===null?'':name));
        }}
        renderInput={ ( params ):
          ReactElement => <TextField {...params} label="Вулиця" />}
      />
      <IconButton color="primary"
        disabled={ !patientEditState }
        onClick={(): void =>{
          setEditCityStreetMode('street');
          setForEditId(streetId);
          setForEditName(streetLabel);
          setOpenEditCityStreetModal(true);
        }} >
        <EditIcon />
      </IconButton>
      <TextField  size={'small'} disabled={ !patientEditState }
```

КОПІЯ 2024

```
              sx={{ m: 1, marginLeft: 10 , width: '8ch' , '& .MuiInputBase-
input.Mui-disabled':{ WebkitTextFillColor:
                    'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
              label="буд" variant="outlined"
              {...register('build') }
            />
            <TextField  size={'small'} disabled={ !patientEditState }
              sx={{ m: 1, width: '8ch', '& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                    'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
              label="корп" variant="outlined"
              {...register('korp') }
            />
            <TextField  size={'small'} disabled={ !patientEditState }
              sx={{ m: 1, width: '6ch', '& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                    'rgba(0, 0, 0, 0.7)' , backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
              label="кв" variant="outlined"
              {...register('flat') }
            />
          </div>
          <div style={ { display: 'inline' }}>
            <TextField sx={{ m:1, width: '15ch', '& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'} disabled={ !patientEditState }
            label="№ картки" variant="outlined" {...register('cardId') }
            />
            <TextField sx={{ m:1, width: '25ch', '& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                'rgba(0, 0, 0, 0.7)' , backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'} disabled={ !patientEditState }
            label="Телефон" variant="outlined" {...register('phone') }
            />
            <TextField sx={{ m:1, width: '30ch', '& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                'rgba(0, 0, 0, 0.7)' , backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
```

КОПЗ_2024

```
            size={'small'} disabled={ !patientEditState }
            label="Місце роботи" variant="outlined" {...register('workplace')
}
            />
            <TextField sx={{ m:1, width: '20ch', '& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                'rgba(0, 0, 0, 0.7)' , backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'} disabled={ !patientEditState }
            label="№ пільгового свідотцтва" variant="outlined"
{...register('pilgSvidot') }
            />
            <TextField sx={{ m:1, width: '15ch','& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'}
            disabled={ !patientEditState }
            label="Документ" variant="outlined" {...register('passportId') }
            />
          </div>
        </div>
        <div style={ { border:'2px solid #0d47a1', borderRadius: 4, margin: 5
} }>
          <Box sx={{ width: '100%', typography: 'body1' , alignItems: 'start'
}}>
            <TabContext value={value}>
              <Box>
                <TabList onChange={handleChange} aria-label="lab API tabs
example">
                  <Tab label="Послуги" value="1" />
                  <Tab label="Записи до лікаря" value="2" />
                  <Tab label="Оплати" value="3" />
                </TabList>
              </Box>
              <TabPanel value="1" sx={ { p: 1 } } >
                <div style={ isLoading?{ height: 300, marginTop: 200 ,
marginLeft: 600, visibility: 'visible' }:
                  { marginTop: 0 , marginLeft: 0, visibility:
'hidden',width:0, height: 0 } }>
                  <Spinner size={ 150 }/>
                </div>
```

КОПІЯ - 2024

```
                <div style={isLoading?{ height: 0,width: 0,visibility:
'hidden' }:
                  { height: 510,visibility: 'visible' }}>
                  <TableContainer component={Paper}>
                    <Table sx={{ minWidth: 650 }} size="small" >
                      <TableHead>
                        <TableRow  style={ { backgroundColor: '#1565c0' } }>
                          <TableCell align="center" style={ { color:
'whitesmoke' } }>№</TableCell>
                          <TableCell align="center" style={ { color:
'whitesmoke' } }>Дата</TableCell>
                          <TableCell style={ { color: 'whitesmoke' } }
>Пацієнт</TableCell>
                          <TableCell style={ { color: 'whitesmoke' } }
>Лікар</TableCell>
                          <TableCell style={ { color: 'whitesmoke' } }
>Процедура</TableCell>
                          <TableCell style={ { color: 'whitesmoke' } }
>Категорія</TableCell>
                          <TableCell align="right" style={ { color:
'whitesmoke' } }>Сума(грн)</TableCell>
                        </TableRow>
                      </TableHead>
                      <TableBody>
                        {rowsInPage.map((row, index) => (
                        <TableRow
                        sx={{ '&:last-child td, &:last-child th': {
border: 0 } }}
                        >
                          <TableCell style={ index%2===1? {
backgroundColor: '#e3f2fd' }
                            : { backgroundColor: 'white' } } component="th"
scope="row" >
                            {index+1}
                          </TableCell>
                          <TableCell style={ index%2===1? {
backgroundColor: '#e3f2fd' }
                            : { backgroundColor: 'white' } } align="center"
                          >{getFormattedDate(row.dateTime, 'dd-MM-yyyy
HH:mm') }
                          </TableCell>
                          <TableCell style={ index%2===1? {
backgroundColor: '#e3f2fd' }
```

КОПЗ_2024

```
                                : { backgroundColor: 'white' } } align="left"
                              >{row.patientName}</TableCell>
                              <TableCell style={ index%2===1? {
backgroundColor: '#e3f2fd' }
                                : { backgroundColor: 'white' } }
align="left">{row.doctorStuff}</TableCell>
                              <TableCell style={ index%2===1? {
backgroundColor: '#e3f2fd' }
                                : { backgroundColor: 'white' } }
align="left">{row.medicalProcedure}</TableCell>
                              <TableCell sx={ { color:
getIconColor(row.operCategory) } }
                                style={ index%2===1? { backgroundColor:
'#e3f2fd' }
                                  : { backgroundColor: 'white'  } }
                                align="center">{
getNewIconElement[row.operCategory] }
                              </TableCell>
                              <TableCell style={ index%2===1? { fontWeight:
'bold',backgroundColor: '#e3f2fd' }
                                : { fontWeight: 'bold', backgroundColor:
'white' } } align="right">{row.payment===0?' '
                                : row.payment }</TableCell>
                            </TableRow>
                          ))}
                        </TableBody>
                      </Table>
                    </TableContainer>
                    <div style={ { display: 'flex' } }>
                      <TablePagination
                        rowsPerPageOptions={[10]}
                        component="div"
                        count={patientLogData.length}
                        rowsPerPage={rowsPerPage}
                        page={page}
                        onPageChange={handleChangePage}
                        onRowsPerPageChange={handleChangeRowsPerPage}
                      />
                    </div>
                  </div>
                </TabPanel>
                <TabPanel value="2" sx={ { p: 1 } }>
                  <div style={{ height: 510, width: '100%' }}>
```

КОПИЯ 2024

```
                <DataGrid
                  rows={patientDoctorInvoice}
                  columns={columnsInvoice}
                  initialState={{
                    pagination: {
                      paginationModel: { page: 0, pageSize: 5 },
                    },
                  }}
                  pageSizeOptions={[5, 10]}
                />
              </div>
            </TabPanel>
            <TabPanel value="3" sx={ { p: 1 } } >
              Проведені оплати
              <div style={{ height: 480, width: '100%' }}>
              </div>
            </TabPanel>
          </TabContext>
        </Box>
      </div>
    </DialogContent>
    <DialogActions>
      <Button color="success"
        style={ { backgroundColor: '#2196f3' } }
        size="large" variant="contained"
        onClick={():void =>{
          setDoctorInvoiceId(0);
          setOpenInvoiceDoctorModal(true);
        }}>
        Запис до лікаря</Button>
      <Button sx={ { color: 'white', backgroundColor: '#1565c0' } }
        size="large" variant="contained"
        disabled={ patientEditState }
        onClick={():void =>setOpenCallDoctorModal(true)}
      >
        Візит до лікаря</Button>
      <Button size="large" variant="contained"
        disabled={ patientEditState && patientCode>0 }
        color = "inherit" onClick={handleClose}> Закрити </Button>
    </DialogActions>
  </form>
  <CallDoctorModal patient = { patient } isOpen= { openCallDoctorModal }
    onClose={():void=>setOpenCallDoctorModal(false)}/>
```

КОПЗ_2024

```
  </Dialog>
  );
};
export { PatientDetailModal };
//Модальний модуль запису до лікаря
import { Autocomplete, Chip, TextField } from '@mui/material';
import Button from '@mui/material/Button';
import Dialog from '@mui/material/Dialog';
import DialogActions from '@mui/material/DialogActions';
import DialogContent from '@mui/material/DialogContent';
import DialogTitle from '@mui/material/DialogTitle';
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs';
import { DatePicker } from '@mui/x-date-pickers/DatePicker';
import { DemoContainer } from '@mui/x-date-pickers/internals/demo';
import { LocalizationProvider } from '@mui/x-date-
pickers/LocalizationProvider';
import { DoctorInvoiceData, DoctorInvoiceSave } from 'common/types/types';
import dayjs, { Dayjs } from 'dayjs';
import { useAppDispatch, useAppSelector } from 'hooks/hooks';
import React, { FC, ReactElement, useEffect } from 'react';
import { useForm } from 'react-hook-form';
import {
  changeDoctorInvoice,
  createDoctorInvoice,
  getAllActiveByDoctorId,
  getDoctorSpecialty,
  getDoctorSpecialtyUnite,
  getDoctorStuff,
} from 'store/register/patient/actions';

import { getFormattedDate } from '../../../helpers/date/get-formatted-
date/get-formatted-date';
import { checkIfVisitTimeAnableInSchedule } from '../helper/check-if-visit-
time-anable-in-schedule';
import { getPatientFullYears } from '../helper/get-patient-full-years';

type Props = {
  isOpen: boolean;
  invoiceId: number;
  onClose: () => void;
  patientCode: number ;
};
```

КоПЗ_2024

```
const InvoiceDoctorModal: FC<Props> = ({ isOpen, onClose, patientCode,
invoiceId }) =>  {
  const dispatch = useAppDispatch();
  const { doctorSpecialtyUnite ,patientDoctorInvoice,doctorSpecialty,
doctorStuff
    , patient , user, activeDoctorInvoice }
    = useAppSelector(( { patient, auth } ) => ({
      doctorSpecialty: patient.doctorSpecialty,
      activeDoctorInvoice: patient.activeDoctorInvoice,
      patientDoctorInvoice: patient.patientDoctorInvoice.filter((item)=>
item.id===invoiceId)[0],
      doctorSpecialtyUnite: patient.doctorSpecialtyUnite,
      doctorStuff: patient.doctorStuff,
      patient: patient.patientFind.filter((item)=> item.patientCode ===
patientCode)[0],
      user: auth.currentUser?.username,
    }));
  const { register, reset,  handleSubmit } =
    useForm< DoctorInvoiceSave >();
  const [filteredDoctorStuff, setFilteredDoctorStuff] =
React.useState(doctorStuff);
  const [visitDay, setVisitDay] = React.useState<Dayjs>(dayjs(Date.now()));
  const [dayOfWeek, setDayOfWeek] =
React.useState<number>((dayjs(Date.now())).toDate().getDay());
  const [visitTime, setVisitTime] = React.useState<string>('');
  const [doctorSchedule, setDoctorSchedule] = React.useState<string[]>([]);
  const [selectedSpecialtyId, setSelectedSpecialtyId] = React.useState(0);
  const [selectedSpecialtyLabel, setSelectedSpecialtyLabel] =
React.useState('');
  const [selectedDoctorStuffId, setSelectedDoctorStuffId] =
React.useState(0);
  const [selectedDoctorStuffLabel, setSelectedDoctorStuffLabel] =
React.useState('');
  useEffect(() => {
    if (invoiceId>0){
      setSelectedDoctorStuffId(patientDoctorInvoice.doctorId);
      setSelectedSpecialtyId(patientDoctorInvoice.specialtyId);
      setVisitDay(dayjs(patientDoctorInvoice.visitDate));
      setSelectedSpecialtyLabel(patientDoctorInvoice.specialtyName);
      setSelectedDoctorStuffLabel(patientDoctorInvoice.doctorName);
    }
    dispatch(getDoctorSpecialty());
    dispatch(getDoctorSpecialtyUnite());
```

КОПІЯ_2024

```
    dispatch(getDoctorStuff());
    reset({ description:patientDoctorInvoice?.description??  '',
    } );
  }, [ dispatch,reset, patient, invoiceId ]);
  const handleClose = (): void => {
    setDayOfWeek((dayjs(Date.now())).toDate().getDay());
    setDoctorSchedule([]);
    setVisitDay(dayjs(Date.now()));
    setVisitTime('');
    setSelectedDoctorStuffId(0);
    setSelectedSpecialtyId(0);
    setSelectedSpecialtyLabel('');
    setSelectedDoctorStuffLabel('');
    reset();
    onClose();
  };
  const selectDoctorSchedule = (doctor: string, day: number): void => {
    setDoctorSchedule(doctorStuff.filter((item)=>item.lastName===doctor)[0]
      .schedule.split('|')[day===0?6:day-1].split(','));

checkIfVisitTimeAnableInSchedule(activeDoctorInvoice,doctorStuff.filter((item
)=>item.lastName===doctor)[0]
      .schedule,visitDay.toDate() );
  };
  const handleSetVisitTimeToDoctor = (newVisitTime: string): void => {
    setVisitTime(newVisitTime===visitTime?'':newVisitTime);
  };
  const handleSetSelectedDoctorStuffId = (doctor:string ): void => {
    if(doctor===''){
      setSelectedDoctorStuffId(0);
      setSelectedDoctorStuffLabel('');
      setDoctorSchedule([]);
    }else{
      const invoiceDoctorStuff =
doctorStuff.filter((item)=>item.lastName===doctor)[0];
      dispatch(getAllActiveByDoctorId(invoiceDoctorStuff.id));
      setSelectedDoctorStuffLabel(doctor);
      setSelectedDoctorStuffId(invoiceDoctorStuff.id);
      selectDoctorSchedule(doctor, dayOfWeek);
    }
  };
  const handleChangeVisitDay = (doctorVisitDay: Dayjs ): void => {
    setDayOfWeek(doctorVisitDay.toDate().getDay());
```

КбПЗ_2024

```
    setVisitDay(doctorVisitDay);
    selectDoctorSchedule(selectedDoctorStuffLabel, doctorVisitDay.day());
  };
  const handleChangeSpecialty = (specialty:string ): void => {
    if(specialty===''){
      setSelectedSpecialtyLabel('');
      setFilteredDoctorStuff(doctorStuff);
    }else{
      const filteredSpecialty =
doctorSpecialty.filter((item)=>item.specialty.includes(specialty))[0].id;
      const filteredDoctorSpecialtyUnite = doctorSpecialtyUnite

.filter((item)=>item.specialtyId===filteredSpecialty).map(function(item) {
return item.doctorId; });
      const filteredDoctorStuff = doctorStuff.filter((item)
=>filteredDoctorSpecialtyUnite.includes(item.id));
      setFilteredDoctorStuff(filteredDoctorStuff);
      setSelectedSpecialtyId(filteredSpecialty);
      setSelectedSpecialtyLabel(specialty);
    }
  };
  const onSubmit = (data: DoctorInvoiceSave ): void => {
    if(invoiceId>0){
      const payload: DoctorInvoiceData = { id:
invoiceId,doctorId:selectedDoctorStuffId,doctorName:'',specialtyName: ''
        , visitDate:visitDay.set('hours',Number(visitTime.split(':')[0]))
          .set('minutes',Number(visitTime.split(':')[1])).set('seconds',
0).toDate(), userId: user?? ''
        , specialtyId: selectedSpecialtyId, description: data.description,
patientCode: patientCode,patientName:'' };
      dispatch(changeDoctorInvoice(payload));
    }else{
      const payload: DoctorInvoiceSave = { ...data, doctorId:
selectedDoctorStuffId
        ,patientCode: patientCode,specialtyId: selectedSpecialtyId,
visitDate:visitDay.set('hours'
          ,Number(visitTime.split(':')[0]))
          .set('minutes',Number(visitTime.split(':')[1])).set('seconds',
0).toDate(), userId: user?? '' };
      dispatch(createDoctorInvoice(payload));
    }
    onClose();
  };
```

КОПІЯ_2024

```
  return ( <Dialog open={isOpen} maxWidth="xl" >
    <form  onSubmit={handleSubmit( onSubmit )} style ={ { minHeight: '400px'
} }>
      <DialogTitle style={ { textAlign: 'center', color: '#0d47a1' } }>Запис
до лікаря</DialogTitle>
      <DialogContent style={{ padding: 1 }} >
        <div style={ { padding: 5, display: 'block', border:'2px solid
darkblue', borderRadius: 4, margin: 5
          , backgroundColor: '#1565c0' } }>
          <h2 style={ { color: 'whitesmoke', marginLeft: '5%' } }>{
patient?.lastName+' '+patient?.firstName+' '
              + patient?.middleName+' ' }({ patient?.dateBirth!=null?
            getFormattedDate(patient?.dateBirth, 'dd.MM.yyyy'):'' })
          {' '+getPatientFullYears(patient?.dateBirth) }</h2>
        </div>
        <div style={ { padding: 5, display: 'block', border:'2px solid
darkblue', borderRadius: 4, margin: 5 } }>
          <div  style={ { display: 'flex', justifyContent: 'space-between' }
}>
            <Autocomplete
              disablePortal
              disabled={ invoiceId>0 }
              size="small"
              value={ selectedSpecialtyLabel }
              ListboxProps={{ style: { maxHeight: '12rem' } }}
              options={doctorSpecialty.map((item) => item.specialty)}
              sx={ { width: 200 , m: 1 } }
              onChange={(_event , specialty: string | null): void =>{
                handleChangeSpecialty(specialty===null?'':specialty);
              }}
              renderInput={ ( params ):
                ReactElement => <TextField {...params} label="Спеціалізація"
/>}
            />
            <Autocomplete
              disablePortal
              disabled={ invoiceId>0 }
              style={ { marginLeft: 5 } }
              size="small"
              value={ selectedDoctorStuffLabel }
              ListboxProps={{ style: { maxHeight: '12rem' } }}
              options={filteredDoctorStuff.map((item) => item.lastName)}
```

КПЗ_2024

```
              sx={ { width: 200 , m: 1 } }
              onChange={(_event , doctorStuff: string | null): void =>{

handleSetSelectedDoctorStuffId(doctorStuff===null?'':doctorStuff);
              }}
              renderInput={ ( params ):
                ReactElement => <TextField {...params} label="Лікар" />}
            />
            <LocalizationProvider dateAdapter={AdapterDayjs}
adapterLocale="en-gb">
              <DemoContainer components={['DatePicker','TimePicker']} sx={ {
marginRight: 1 } }>
                <DatePicker
                  value={ visitDay }
                  slotProps={{ textField: { size: 'small', sx: { width:
'50px' }  } }}
                  onChange={(newValue):void =>{

handleChangeVisitDay(newValue===null?dayjs(Date.now()):newValue);
                  }}
                  label="Дата" />
              </DemoContainer>
            </LocalizationProvider>
            <TextField sx={{ m:1, width: 90, backgroundColor: 'white' }}
              size={'small'}
              label="Час" variant="outlined" value = { visitTime }
            />
          </div>
          <div style={ { padding: 5, display: 'block', border:'1px solid
darkblue', borderRadius: 4, margin: 5
            , minHeight: 65 } }>
            {doctorSchedule
              .map((i: string)=>(
                <Chip
disabled={checkIfVisitTimeAnableInSchedule(activeDoctorInvoice,i,visitDay.toD
ate() )}
                  label={ i } size="medium"
                  color="primary" variant="outlined" onClick={():void =>{
                    handleSetVisitTimeToDoctor(i);
                  }}
                  style={ { margin: 10, padding: 5, fontWeight: 'bold',
fontSize: 25 } }
                />
```

КбТЗ_2024

```
              ))}
          </div>
          <div>
            <TextField sx={{ m:1, width: '100ch','& .MuiInputBase-input.Mui-
disabled':{ WebkitTextFillColor:
                  'rgba(0, 0, 0, 0.7)', backgroundColor: 'white' },
backgroundColor: 'rgb(232, 241, 250)' }}
            size={'small'}
            label="Примітки" variant="outlined" {...register('description') }
            />
          </div>
        </div>
      </DialogContent>
      <DialogActions style={ { marginTop: '100px' } }>
        <Button color="primary"
          disabled={ selectedSpecialtyId===0 || selectedDoctorStuffId===0 ||
visitTime ==='' }
          type="submit" size="large"
          variant="contained" >Записать</Button>
        <Button size="large" variant="contained" color = "inherit"
onClick={handleClose}> Закрити </Button>
      </DialogActions>
    </form>
  </Dialog>
  );
};

export { InvoiceDoctorModal };
```

КбПЗ_2024