Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики і радіоелектроніки, Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему: ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ДИСТАНЦІЙНОЇ СИСТЕМИ КОНСУЛЬТАЦІЙ СІМЕЙНОГО ЛІКАРЯ З УРАХУВАННЯМ РІВНЯ УРГЕНТНОСТІ ПАЦІЄНТІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF A SYSTEM FOR REMOTE CONSULTATION BY A FAMILY DOCTOR REGARDING THE URGENCY LEVEL OF THE PATIENTS

Виконав: студент(ка) 2 курсу, групи КНТ-220м

Спеціальності 122 «Комп'ютерні науки»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Спєваков Р.В.
(прізвище та ініціали)

Керівник Зеленьова І.Я.
(прізвище та ініціали)

Рецензент Терещенко Е.В.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 «Комп'ютерні науки»
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
проф., д.т.н., С.О. Субботін
«_____» _____20___року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Спєвакова Романа Валерійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація дистанційної системи консультацій сімейного лікаря з урахуванням рівня ургентності пацієнтів. / Research and Software Implementation of a System for Remote Consultation by a Family Doctor Regarding the Urgency Level of the Patients.
керівник проєкту (роботи) Зеленьова Ірина Яківна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «05»листопада2021року №435
2. Строк подання студентом проєкту (роботи) 10 грудня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1 Огляд та аналіз медичних інформаційних систем в Україні. 2 Визначення вимог та обґрунтування структури дистанційної системи, що проєктується. 3 Основні рішення з реалізації дистанційної системи. 4 Проєктування інтерфейсів системи. 5 Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Зеленьова І.Я., доцент		
Нормоконтролер	Бєлова А.В., асистент		

7. Дата видачі завдання « 06 » вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Огляд та аналіз сучасних медичних систем в Україні	1 тиждень	Висновки за розділом
2	Визначення вимог та обґрунтування структури дистанційної системи	2-3 тиждень	Технічне завдання, структурна схема, функціональна схема
3	Основні рішення з реалізації дистанційної системи	4-5 тиждень	Вибір СКБД, проєктування БД, компоновка програми
4	Проєктування інтерфейсів системи	6-7 тиждень	Екранні форми
5	Керівництво користувача	8-9 тиждень	Структура меню
6	Розробка розділів пояснювальної записки	10-11 тиждень	Пояснювальна записка
7	Розробка слайдів презентації	12-13 тиждень	Слайди презентації
8	Захист магістерської роботи	14 тиждень	

Студент(ка) _____ Спєваков Р.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Зеленьова І.Я.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 168 с., 17 табл., 58 рис., 5 дод., 30 джерел.

ДИСТАНЦІЙНА СИСТЕМА, ТЕЛЕМЕДИЦИНА, БАЗА ДАНИХ, ЗАХВОРЮВАНІСТЬ, СІМЕЙНИЙ ЛІКАР, РІВЕНЬ УРГЕНТНОСТІ, СИМПТОМИ, ІНТЕРФЕЙС.

Об’єкт проєктування – вебсайт для створення пацієнтами звернень до своїх сімейних лікарів та для отримання рекомендацій на звернення.

Предмет проєктування – база даних, де зберігаються звернення пацієнтів до лікарів з поточними даними біометрії та основними симптомами захворювань, а також вебінтерфейс для доступу до неї.

Мета роботи – розробка дистанційної системи консультацій сімейного лікаря, яка дозволить оперативно обчислювати рівень ургентності пацієнтів та виконувати сортування звернень до сімейного лікаря в залежності від тяжкості захворювання відповідного пацієнта.

Наукова новизна роботи полягає в запропонованому способі використання біометричних даних, зокрема температури і кров’яного тиску, для прискорення аналізу та класифікації стану пацієнта.

Практична цінність роботи полягає в тому, що досліджено та реалізовано спосіб обчислення рівня ургентності пацієнта в залежності від його схильностей до захворювань та поточного стану здоров’я. Цей спосіб можна буде запропонувати для використання у сучасних системах телемедицини.

Розроблена дистанційна система дозволить виконувати такі функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації декількох біометричних даних, а також основних симптомів захворювань;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні.

Серверну частину призначено для функціонування під операційною системою Linux, а клієнтську частину для будь-якого браузера.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 168 p., 17 tabs, 58 fig., 5 appendixes, 30 references.

REMOTE SYSTEM, TELEMEDICINE, DATABASE, DISEASE, FAMILY DOCTOR, URGENCY LEVEL, SYMPTOMS, INTERFACE.

Object of design – a website for patients to contact their family doctors and to receive recommendations for treatment.

The subject of the design is a database where patients' appeals to doctors with current biometric data and the main symptoms of diseases are stored, as well as a web interface for accessing it.

The aim of the work is to develop a remote system of family doctor consultations, which will allow to quickly calculate the patients' urgency level and perform calls to the family doctor sorting depending on the severity of the disease of the patient.

The scientific novelty of the work lies in the proposed method of using biometric data, including temperature and blood pressure, to speed up the analysis and classification of the patient's condition.

The practical value of the work is that the method of calculating the level of urgency of the patient depending on his susceptibility to disease and current health is researched and implemented. This method can be proposed for use in modern telemedicine systems.

The developed remote system will allow to perform the following functions:

– storage in the database of patients' appeals with the possibility of registration of several biometric data, as well as the main symptoms of diseases;

– the option for the doctor to remotely make a medical opinion with recommendations for treatment and prescription of drugs;

– automatic sorting of patients' appeals to the doctor depending on the severity of symptoms and the worst biometric indicators in the disease.

The server part is designed to run under the Linux operating system, and the client part should be accessible via any web browser.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

1NF (1 Normal Form)	– перша нормальна форма;
2NF (2 Normal Form)	– друга нормальна форма;
4NF (4 Normal Form)	– четверта нормальна форма;
CSS (Cascading Style Sheets)	– каскадні таблиці стилю;
ER (Entry Relationship)	– діаграма відображення концептуальної моделі бази даних;
HTML (HyperText Markup Language)	– мова розмітки гіпертексту;
JS (JavaScript)	– динамічна, об’єктно-орієнтована прототипна мова програмування;
jQuery	– бібліотека на мові JavaScript;
PHP (Hypertext Preprocessor)	– скриптова мова програмування;
SARS-CoV-2	– код штаму грипу, який викликає захворювання на COVID-19;
SHA-512 (Secure Hash Algorithm)	– безпечний алгоритм ґешування;
SQL (Structured Query Language)	– мова структурованих запитів;
БД	– база даних;
СКБД	– система керування базами даних.

ВСТУП

Майже у цілому Світі найважливішим було, є і буде життя людини. Життя людини пропорційно залежить від її здоров'я. За здоров'ям людей безперервно стежать лікарі – майстри рідкісної найскладнішої професії. Кількість людей у Світі безперервно зростає. Технічний прогрес крокує все швидше уперед, але прогрес медичний не повинен залишатися позаду, тому що життя і здоров'я має великі значення. Наша країна все швидше і швидше крокує шляхом поступової диджиталізації (використання електронно-обчислювальної техніки на допомогу людям). Цей процес не оминув і галузь медицини. Усі сучасні нововведення у медицині повинні покращити якість і швидкість медичного обслуговування та полегшити тяжку працю лікарів. Згідно з проєктами цифрової трансформації медичних послуг та управління медичною інформацією планується запровадити Електронну систему охорони здоров'я (ЕСОЗ), яка має забезпечити автоматизацію ведення обліку медичних послуг та управління медичною інформацією, зокрема спростити роботу закладів охорони здоров'я, сприяти якості та доступності медичних послуг для пацієнтів [1]. Але всі нововведення не впроваджуються миттєво, а якщо вже їх впроваджено, то зрозуміло, що вони не ідеальні та мають будь-які недоліки, що потрібно постійно ліквідовувати у процесі модернізації електронних систем. Вже 2 роки у Світі вирує особливо небезпечний вірус грипу COVID-19 (або SARS-CoV-2), який викликає багато смертельних ускладнень. Цю інфекцію можуть підхопити не тільки пересічні люди, але й лікарі, які ще повинні далі лікувати багато пацієнтів. Адже, розробка дистанційної системи консультацій сімейного лікаря є на сьогодення дуже актуальною темою. У період пандемії COVID-19 кількість звернень пацієнтів до сімейного лікаря зростає майже в десятки разів, і щоб не залишити без уваги пацієнта, який потребує допомоги у першу чергу, виникла ідея запропонувати у дистанційній системі консультацій сімейного лікаря програмний алгоритм урахування рівня ургентності пацієнтів.

1 ОГЛЯД ТА АНАЛІЗ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ В УКРАЇНІ

1.1 Загальні поняття медичної інформаційної системи

На сьогодні в Україні створена та функціонує електронна система охорони здоров'я (ЕСОЗ) – інформаційно-телекомунікаційна система, яка забезпечує автоматизацію ведення обліку медичних послуг та управління медичною інформацією в електронному вигляді. Технічна архітектура ЕСОЗ передбачає так звану гібридну модель – двокомпонентну систему з єдиною центральною базою даних (ЦБД), яка належить державі, та множиною бізнесових електронних медичних інформаційних систем (МІС). При цьому держава встановлює правила та стандарти ЕСОЗ, гарантує безпеку системи, а бізнес відповідальний за надання сервісів користувачам. Зазначене архітектурне рішення має суттєві переваги для користувачів системи. Так, користувачі системи (пацієнти, лікарі, аптечні працівники, менеджери закладів) через наявність великої кількості МІС отримують конкурентний вибір електронних сервісів, належну технічну та консультаційну підтримку щодо роботи в системі. При цьому жодна важлива інформація не буде втрачена через зміну МІС – всі ключові дані зберігаються централізовано в ЦБД. Це є перевагою і для держави – така інформація показує цілісну картину надання медичних послуг для всієї країни, що дозволяє здійснювати якісний аналіз та прогноз необхідних медичних послуг для населення. ЕСОЗ є одним з основних інструментів впровадження нової моделі фінансування медичної галузі. Розвиток ЕСОЗ синхронізовано з іншими системними змінами в охороні здоров'я [2].

Медична інформаційна система (МІС) – інформаційна система для автоматизації лікувальної установи. Початок планування роботи з такими системами запропоновано в Україні з 2017 року та планується завершити основні етапи впровадження до початку 2023 року [3].

Медична інформаційна система складається із спеціалізованого програмного забезпечення, яке розроблене спеціально під потреби системи охорони здоров'я. Від інформаційних систем для інших галузей МІС відрізняється тим, що одночасно зберігає і обробляє персональну, демографічну та медичну інформацію пацієнта. Медична інформаційна система надає функціонал, необхідний медустанові для взаємодії з eHealth та Національною службою здоров'я України. Але тільки цими функціями можливості такої системи не обмежуються. Як правило, розробники МІС надають повний комплекс можливостей для автоматизації різних процесів у клініці [4].

Медична інформаційна система це – комплексний програмний продукт, головне призначення якого автоматизація всіх основних процесів, пов'язаних із роботою медичних установ загальної та вузької спеціалізації. Автоматизовані медичні інформаційні системи дозволяють швидко та ефективно налагодити електронний документообіг, гнучко вибудовувати роботу з пацієнтами, вести оперативний облік роботи адміністративного персоналу, контролювати всі організаційні та фінансові питання [5].

За допомогою МІС медустанова може:

– автоматизувати роботу реєстратури, упорядкувавши та спростивши процедуру запису пацієнтів на прийом;

– систематизувати інформацію щодо всіх пацієнтів клініки, медичних послуг та співробітників;

– керувати матеріальним фондом медустанови, чергою на місця в стаціонарі, стежити за рухом медикаментів на складі та між відділеннями;

– упорядкувати роботу лабораторій та діагностичних кабінетів, організувати оперативне передавання даних щодо результатів досліджень фахівцеві в автоматичному режимі;

– збирати статистику, готувати звіти та аналітику в декілька дій [4].

1.2 Класифікація медичних інформаційних систем

Медичні інформаційні системи класифікують за різними ознаками.

А. Залежно від ступеня автоматизації процесів збору й обробки інформації, МІС поділяються на автоматизовані та автоматичні. В автоматизованих системах частина операцій по збору й обробці інформації виконується людиною. Автоматичні системи припускають повне виключення людини з процесів збору та обробки інформації [3].

Б. Залежно від типу інформаційної бази, МІС поділяються на системи, що оперують даними, та системи, що оперують знаннями. Системи другого типу – це експертні системи. Їхнє функціонування істотно спирається на знання, отримані від експертів, а результати функціонування близькі результатам аналітичної діяльності експертів [3].

В. Залежно від виду розв’язуваних задач, МІС можна розділити на такі групи:

– інформаційно-довідкові – системи автоматизованого пошуку, вимірювальні системи;

– інформаційно-логічні – діагностичні системи, системи прогнозу, системи моніторингу;

– керуючі або автоматизовані системи управління [3].

Інформаційно-логічна система призначена для перетворення інформації таким чином, щоб можна було одержати нову інформацію, відсутню в інформаційному масиві. У системах управління реалізується принципово нова функція – прийняття керуючих рішень. Найбільш широке поширення в медичних установах одержали інформаційно-пошукові системи (ІПС), які у залежності від характеру інформації поділяються на фактографічні та документальні системи [3].

Фактографічні ІПС містять інформаційні масиви фактичних даних. Аналогами таких систем виступають «паперові» довідники, каталоги, технічні паспорти. У комп’ютерних ІПС фактичні дані звичайно зберігаються

в базах даних (БД) і являють собою таблиці, у колонках яких вказано назви різних характеристик об’єктів, а в рядках дані опису (значення характеристик) цих об’єктів. Документальні ІПС оперують з інформацією у вигляді документів. Прикладами таких систем можуть бути бібліографічна картотека, картотека з історіями хвороб, інші картотеки. Виконуючи пошук, документальна ІПС надає або номера необхідних документів, або список заголовків, або адреси зберігання шуканих документів. При цьому оцінку інформації, що знаходиться в знайдених документах, робить людина [3].

Керуючі системи реалізують збір інформації про об’єкт управління, обробку інформації, передачу даних до органу управління, формування керуючого рішення [3].

Г. МІС можна класифікувати і за ієрархічним принципом, що відповідає багаторівневій структурі охорони здоров’я, як галузі. У цьому випадку їх, зазвичай, розподіляють за чотирма рівнями:

– базовий (або клінічний) рівень (лікарі різного профілю);

– рівень лікувально-профілактичного закладу (поліклініка, стаціонар, диспансер, швидка допомога тощо);

– територіальний рівень (профільні та спеціалізовані медичні служби і регіональні органи керування);

– державний рівень (державні заклади та органи управління) [3].

1.3 Інформаційне забезпечення медичних інформаційних систем

Медичні інформаційні системи характеризуються наявністю, як правило, великих обсягів даних і знань. Обробка даних і знань зводиться до трьох основних етапів. На першому етапі елементи інформації розміщуються у визначених структурах – базах даних (БД) і базах знань (БЗ). На другому етапі БД і БЗ піддаються упорядкуванню: змінюється їхня структура, порядок розміщення інформації, характер взаємозв’язків між елементами інформації. На третьому етапі здійснюють експлуатацію БД: пошук потрібної інформації,

прийняття рішень, редагування баз даних і знань. Інформаційне забезпечення МІС складають: історії хвороби, виписки з історій хвороби, епікризів, стандартизованих карт обстеження, діагностичні та інформативні оцінки показників і станів, критерії ефективності обстеження та лікування, каталог медичних понять і термінів [3].

1.4 Основні функціональні компоненти медичних систем

Комплексні медичні інформаційні системи, як правило, складаються з модулів. Це дозволяє зібрати та налаштувати МІС у потрібній конфігурації для установ різного типу і забезпечити необхідний функціонал із можливістю подальшого додавання/видалення модулів. Структура медичної інформаційної системи – окремі компоненти, що можна об’єднати у кілька великих груп [5]:

– аналітичні та управлінські компоненти, – модулі та засоби ведення управлінського обліку, інструменти аналізу якості та ефективності медичних послуг. Ці складові МІС дозволяють проаналізувати стан вашої медичної організації, виявити проблемні місця й оптимізувати бізнес-процеси. На рівні користувача – пошук медичних записів за будь-якими критеріями, з урахуванням обмежень за рівнем доступу. Результати аналізу можна вивести на екран у вигляді графіків, таблиць або на друк;

– медичні компоненти, – усі модулі, пов’язані з реєстрацією пацієнтів, ведення реєстру електронних медичних карт, облік лікарняних листів, ведення протоколів лікування, інформаційний супровід лікування пацієнтів у різних типах установ (амбулаторія, поліклініка, стаціонар), медична статистика й аналітика, історія хвороби та багато іншого;

– фінансово-економічні компоненти, – до них відносяться інструменти ведення обліку медикаментів, управління запасами, розрахунок собівартості лікування та тарифів на надання медичних послуг, розрахунок

надбавок лікарям, інструменти проведення економічного аналізу діяльності організації та інше;

– компоненти обміну даними, – ведення уніфікованих реєстрів, каталогів і довідників, обмін даними в системі закладів охорони здоров'я, обробка отриманих даних;

– загальнотехнічні компоненти, – контроль доступу користувачів і захист бази даних, а також підтримка можливостей інтеграції з іншими системами та програмами [5].

1.5 Переваги використання медичних інформаційних систем

Користь для медичних закладів:

– позбавляє від заповнення паперів. Не потрібно дублювати записи та вносити інформацію до інших документів: лікарі та персонал, що мають доступ до карти пацієнта, можуть ознайомитися з історією хвороби, ходом лікування, результатами досліджень та іншим з єдиної бази даних;

– підвищення якості обслуговування та зниження впливу людського фактору. Автоматизований документообіг дозволяє зменшити кількість паперової роботи, успішно вести базу клієнтів, спираючись на актуальну інформацію про дослідження та надані послуги. Лікарям набагато простіше ставити діагноз, знижується ризик втрати важливих даних, як це часто буває з результатами аналізів у паперовому вигляді: їх можуть банально прикріпити до чужої карти, через що доводиться проходити дослідження повторно;

– телемедицина. Лікарі отримують можливість проконсультуватися в реальному часі з колегами та іншими фахівцями стосовно правильності постановки діагнозу (особливо, в екстреній ситуації), призначення та корекції лікування, вести дистанційний моніторинг стану хворих та інше;

– узгодженість роботи. Клінікам зручно мати онлайн-реєстратуру, вести загальнолікарняні бази пацієнтів, розподіляти їх по філіях із

урахуванням завантаженості та графіка фахівців, маючи при цьому можливість оцінити попит на ті чи інші послуги, а приватним центрам – формувати ціноутворення [5].

Користь для пацієнтів:

– завдяки МІС пацієнти отримують доступ до своїх даних, можуть оперативно отримувати результати лабораторних аналізів і відстежувати їх разом із лікарем, записуватися до нього на прийом, підтримувати зворотний зв’язок та інше;

– знижується ймовірність підробки та втрати медичних даних, адже пацієнт їх моніторить самостійно;

– система попереднього онлайн-запису дозволяє уникнути черги в лікарнях [5].

1.6 Огляд сучасних медичних інформаційних систем в Україні

В Україні медичні заклади зможуть обирати будь-яку медичну інформаційну систему з низки тих, які пройшли перевірку та підключилися до центрального компоненту системи «eHealth» [6].

1.6.1 Health24

Повнофункціональна хмарна МІС, що об’єднує в собі функціональні сервіси, які забезпечують роботу лікаря та медичного закладу відповідно існуючих стандартів медичного документообігу [7].

1.6.2 Helsi

Повнофункціональна система для керування медичним закладом. Функціонал системи розроблений та адаптований з урахуванням специфіки роботи та вимог МОЗ, а також є безкоштовним для державних установ [8].

1.6.3 EMCiMED

Розроблена відповідно до стандартів ISO та МОЗ України, забезпечує технічний захист інформації, містить модулі: електронна медична картка пацієнта, медичні документи (облікові медичні форми МОЗ), медичні кадри, поліклініка та реєстратура, стаціонар, лабораторія (інтеграція лабораторного обладнання), склад та персоніфікований облік ліків, статистика та звіти МОЗ, послуги, контакт-центр, PACS, партнери, онлайн-запис до лікаря, мобільний застосунок пацієнта. Передова українська медична інформаційна система для медичних установ, приватних клінік і лабораторій, підключена до системи eHealth України. Складається з модулів, що легко можна збирати в потрібній конфігурації для кожного окремого закладу. Основні підтримувані модулі: реєстратура, управління персоналом, управління організацією, поліклініка, стаціонар, лабораторія, управління партнерськими відносинами. Можна придбати додаткові модулі: облік послуг, управління запасами, архів медичних зображень PACS та інші. За необхідності вони можуть бути поставлені в складі додаткових пакетів EMCiMED-Поліклініка та EMCiMED-Стаціонар [4], [5], [9].

Переваги інформаційної системи:

– можливість обирати модулі відповідно до вимог організації;

– гнучке налаштування;

– потужна функціональна складова;

– система захищена завдяки використанню USB-брелоків та надійного шифрування всієї інформації;

– підтримує інтеграцію з іншими продуктами, наприклад, із 1С;

– систему перевірено і рекомендовано до використання Міністерством охорони здоров’я України [5], [9].

Недоліки системи:

– якщо виявлені з моменту впровадження, то істотно не впливають на роботу з програмою [5], [9].

Оплата та вартість інформаційної системи:

– застосування як строкової так і безстрокової ліцензії, яка припускає одноразову оплату на весь період використання [4], [5], [9].

1.6.4 Доктор Елекс

Найпоширеніша в Україні медична система, що працює з 2005 року. Станом на 2018 в ній ведуться електронні картки більш ніж 5 млн. пацієнтів. Система забезпечує автоматизацію ключових процесів медичної установи, зокрема ведення електронної історії хвороби, формування управлінської звітності та документації згідно з вимогами МОЗ. Комплексне рішення, що дозволяє оптимізувати роботу клінік будь-якого розміру та профілю (приватних і державних). Розробник – компанія Eleks (Львів, Україна). Doctor Eleks підтримує електронну медичну карту пацієнта, інструменти редагування шаблонів документів, особистий кабінет лікаря, модуль реєстратури та роботи зі звітністю, фінансами, персоналом. Підсистема розкладів дозволяє формувати графіки роботи співробітників з урахуванням побажань лікарів і пацієнтів (рис. 1.1) [10].

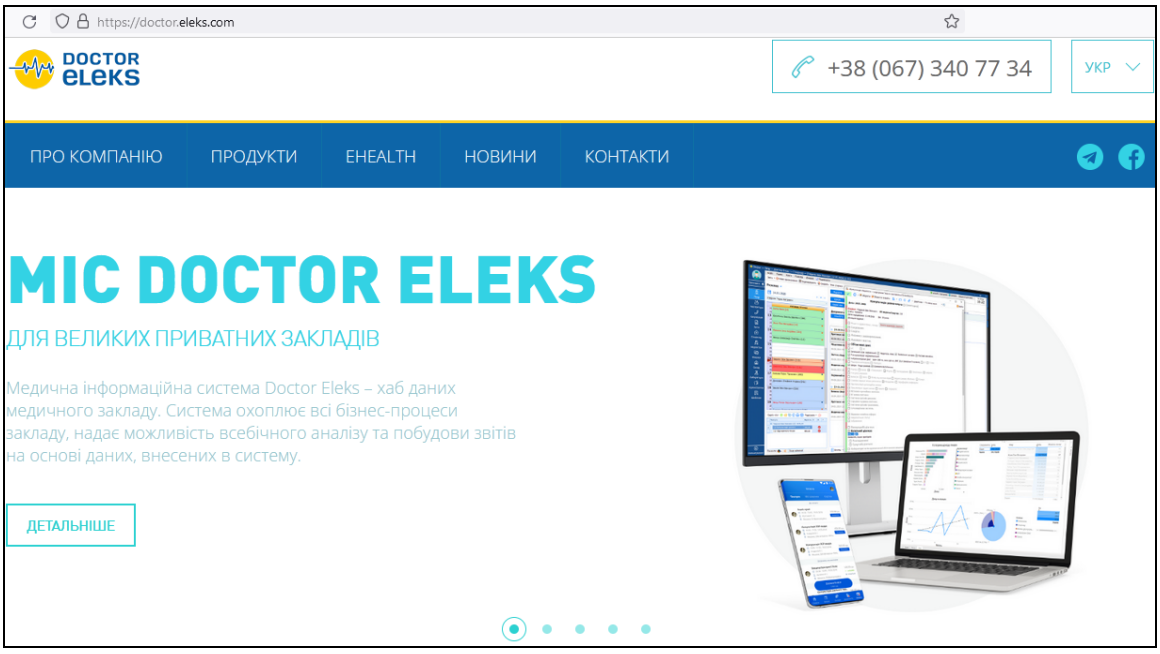


Рисунок 1.1 – Медична інформаційна система «Doctor Eleks»

Лабораторна інформаційна система Doctor Eleks може використовуватися як окремий програмний продукт. Серед додаткових можливостей – повноцінний редактор для обробки відео та зображень, які можна додавати до документів. Гнучка технологія побудови звітів, є можливість проводити аудит медичних документів, підтримується модуль PACS, веб-клієнт і багато іншого [5], [10].

Переваги інформаційної системи:

– потужний функціонал;

– наявність комунікаційного серверу для обміну даними у форматі HL7 із суміжними ІС, зовнішніми лабораторіями, страховими компаніями;

– присутня інтеграція з Toshiba УЗД, підтримується імпорт DICOM-зображень, підключення DICOM-сумісного обладнання та багато іншого;

– систему Doctor Eleks підключено до системи eHealth;

– система пройшла перевірку після якої було її рекомендовано до використання МОЗ України [5], [10].

Недоліки системи:

– не підтримуються електронні напрямки [5], [10].

Оплата та вартість інформаційної системи:

– ліцензування програмного забезпечення проводиться на основі постійних і тимчасових ліцензій;

– окремо оплачуються послуги з впровадження, інтеграції, технічної підтримки, вартість використання мобільного застосунку, веб-послуги та робота зі страховими компаніями [5], [10].

1.6.5 «Поліклініка без черг»

Система керування потоком пацієнтів та автоматизації в лікувальних закладах. Система інтегрується з системами та модулями для роботи з електронною карткою пацієнта, електронним рецептом і обліковими системами [11].

1.6.6 MC Plus

Універсальна медична інформаційна система, основним компонентом якої є електронна медична карта пацієнта, розроблена відповідно до стандартів МОЗ України. Особливістю цієї МІС є використання процесорного підходу до автоматизації роботи медичного закладу у лікувальній та адміністративній діяльності. Станом на вересень 2020 року за допомогою хмарного рішення MC Plus користувачі можуть отримати більшість послуг, які передбачено модулями системи eHealth: робота з деклараціями пацієнтів, електронні направлення та медичні записи, надання або отримання електронного рецепту «Доступні ліки», реєстрація надавачів медичних послуг первинної та спеціалізованої ланки, оформлення та ведення договорів с НСЗУ, ведення електронних медичних записів, діагностичні звіти, ЕМЗ та ЕН для не ідентифікованих пацієнтів [12], [13].

1.6.7 Askep.net

Міжнародне хмарне SaaS рішення для автоматизації робочих процесів медичних закладів, що містить наступні модулі: робота з eHealth, картка пацієнта, поліклініка (запис на прийом), стаціонар (ведення 066 та інших форм), лабораторія (результати аналізів), електронний рецепт, статистика (внутрішня та формування офіційної), спеціалізовані рішення (стоматологія, пологові будинки, онкологія, дерматологія, УЗД, психіатрія та інші), інтеграція із сторонніми сервісами та технікою [14].

1.6.8 SimplexMed

Ця медична інформаційна система забезпечує повний функціонал для медичного закладу. Працює в Україні з 2000 року. Працює як в ЦПМСД, так і в госпітальних закладах другого, третього рівнів та закладах національного

рівня. Є повнофункціональною системою медичного закладу, що охоплює медичні, фінансові, бухгалтерські, складські, кадрові аспекти. Включає систему керування взаємовідносинами з клієнтами, телемедицину на рівні лікар-лікар, лікар-пацієнт, власні лабораторні та діагностичні підсистеми, інтеграцію з лабораторним та діагностичним обладнанням. Взаємодіє з різними національними ресурсами. Також надає комплексні портальні корпоративні рішення для мереж клінік, окремих областей України щодо обміну, агрегації інформації, планування та моніторингу. Надає програмні кабінети та мобільні застосунки для пацієнтів [15].

1.6.9 MedElement

Медична інформаційна система, яка розроблена в Казахстані. Поєднання хмарних сервісів і потужної довідкової системи для лікарів, студентів-медиків і всіх, кому важлива турбота про здоров'я. Сфера застосування MedElement – автоматизація роботи клінік, клінік ДРТ, стоматології, аптек, блоків харчування, приватних медичних практик та інших (рис. 1.2) [16].

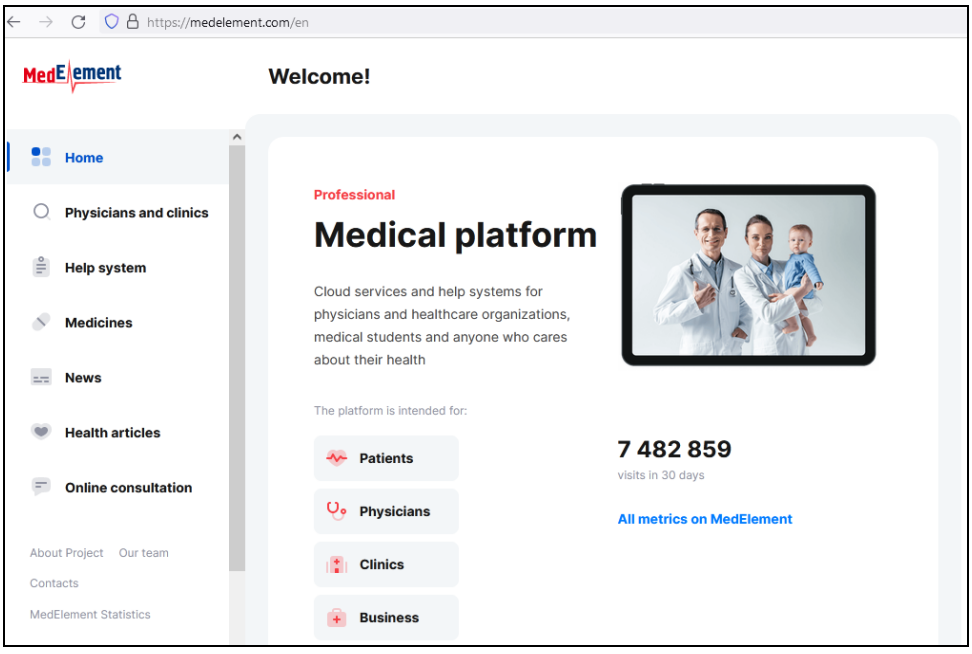


Рисунок 1.2 – Медична інформаційна система «MedElement»

Цікавою особливістю даної МІС є те, що крім підтримки основних модулів, вона є потужною довідковою системою. Тут містяться довідники захворювань, медичних термінів, лабораторних показників, лікарських засобів, розміщуються огляди світової періодики та інше. Більш того, вона має переваги хмарної МІС: підтримується автоматизація всієї медичної документації, формування звітів, збір маркетингової інформації, облік фінансів, послуг та інше [16].

Переваги інформаційної системи:

– зручні веб-сервіси;

– наявність мобільного застосування для швидкого пошуку лікаря, записи на прийом та ведення комунікації;

– застосування технології допомоги щодо прийняття клінічних рішень;

– зв’язок інформаційної системи з онлайн-базою інтерактивних медичних довідників [16].

Недоліки системи:

– не надто зручний інтерфейс;

– підтримка «всього і відразу» не завжди є плюсом [16].

Оплата інформаційної системи:

– за підключення до системи MedElement здійснюється абонентська плата за місяць, півроку, рік [16].

1.6.10 МедІнфоСервіс

Медична інформаційна система, яка охоплює автоматизацію лікувальних процесів амбулаторно-поліклінічних та стаціонарних лікувальних закладів, акредитована МОЗ та підключена до електронної системи охорони здоров’я eHealth. Обираючи МІС «МедІнфоСервіс» можна отримати повний функціонал для передачі електронних медичних записів до ЕСОЗ з функцією імпорту даних з «Медичної карти амбулаторного хворого»

та «Медичної карти стаціонарного хворого» до електронних медичних записів, а це означає, що лікарям не потрібно робити потрійну роботу, а достатньо лише внести дані до електронної документації з подальшою можливістю імпорту та друку документів [17].

Медична інформаційна система «МедІнфоСервіс» вже впроваджена та успішно функціонує більш ніж у 100 медичних закладах України різних ланок надання медичної допомоги, рівнів підпорядкування та форм власності. Що дає змогу більш ніж 3000 лікарів вносити електронні медичні записи до ЕСОЗ та отримувати фінансування (рис. 1.3) [17].

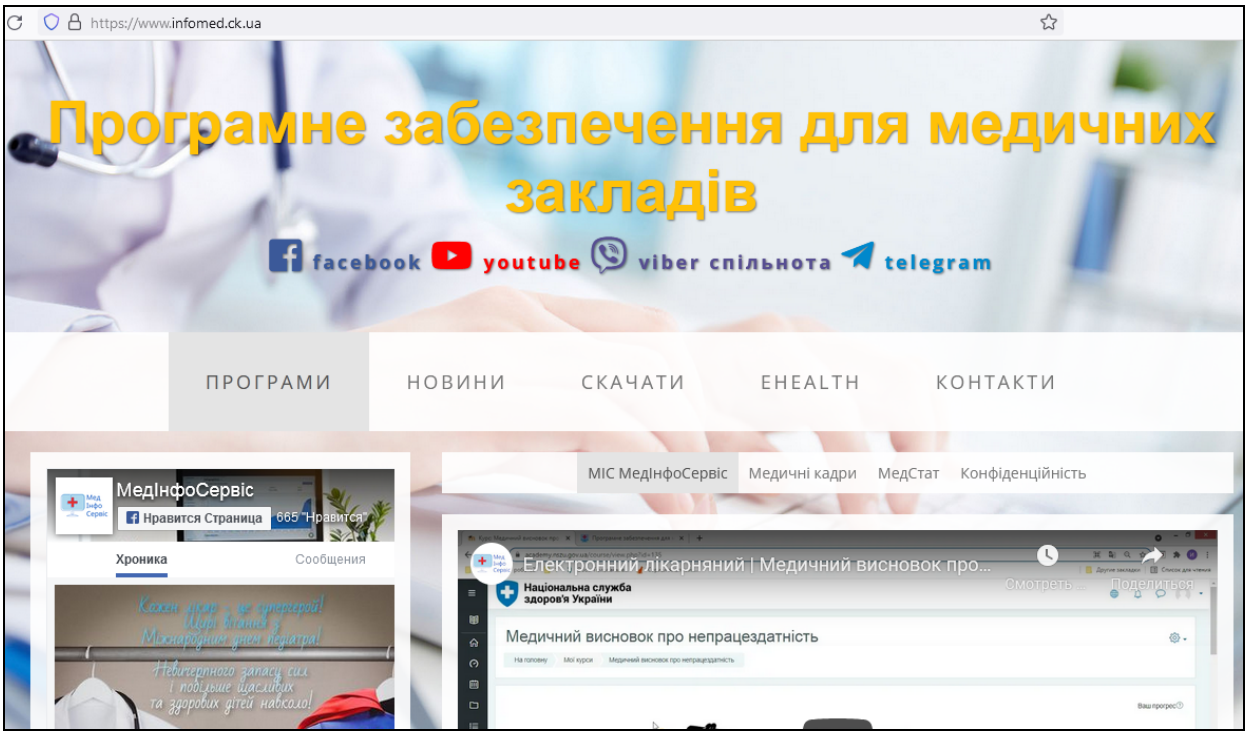


Рисунок 1.3 – Медична інформаційна система «МедІнфоСервіс»

Заклади, які в своїй діяльності користуються МІС «МедІнфоСервіс» одні з перших отримали повний функціонал для роботи та мають успішний досвід передачі інформації до ЕСОЗ та вчасно і в повному обсязі отримують оплату за договорами, а команда МІС «МедІнфоСервіс» використала цей час для того, щоб налаштувати контролі та обмеження для запобігання внесенню некоректних даних та уникненню помилок при звітуванні [17].

Переваги системи:

– підключена до ЕСОЗ з функцією імпорту даних до ЕМЗ;

– зручний у використанні та простий для користувачів інтерфейс;

– постійний youtube-канал МІС «МедІнфоСервіс» з детальними відео-інструкціями з користування, які постійно оновлюються;

– швидке реагування на звернення користувачів технічної підтримки та розробників МІС;

– потребує мінімальних витрат на встановлення та експлуатацію;

– активно розвивається та оновлюється;

– швидке впровадження змін та індивідуальний підхід до кожного клієнта;

– усі документи, які виводяться на друк, можливо формувати, як у Microsoft Office, так і в безкоштовному LibreOffice;

– для запобігання помилок при внесенні інформації та полегшення роботи медичного персоналу реалізовані шаблони на заповнення текстових блоків інформації, а також на набори даних при створенні направлень та заповненні Листка лікарських призначень;

– програма містить велику кількість вкладених довідників, основними з яких є: "Аналізи та дослідження", "Огляд пацієнта", "Анамнези та скарги", "Міжнародна класифікація первинної медичної допомоги ICPC-2", "Міжнародний класифікатор хвороб", "Австралійська модифікація МКХ-10-АМ", "Класифікатор медичних процедур та хірургічних операції", Австралійський класифікатор медичних інтервенцій (АКМІ), "Лікарські засоби" (Державний реєстр лікарських засобів України), "Лікарські засоби, вартість яких підлягає відшкодуванню" (Доступні ліки) [17].

1.6.11 IMED

Інформаційна система для медичних закладів, приватних клінік та медичних центрів (рис. 1.4) [18].

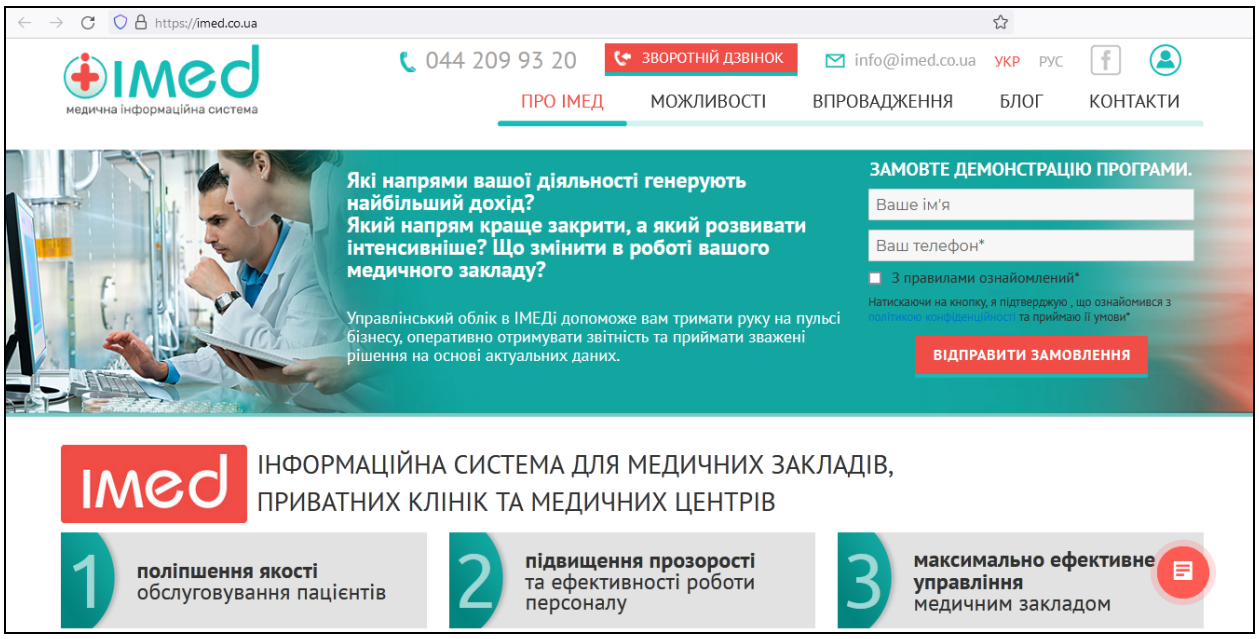


Рисунок 1.4 – Медична інформаційна система «IMED»

Переваги інформаційної системи:

– усі операції та інформація – від реєстрації пацієнтів до фінансових звітів власнику – в одній системі;

– зрозумілий інтерфейс допомагає користувачу швидко знаходити потрібний функціонал;

– система гнучко налаштовується під потреби бізнесу та має можливість доопрацювання необхідного функціоналу;

– керівництво медичного закладу може швидко проаналізувати поточний стан справ завдяки постійному моніторингу показників діяльності;

– у IMED вже реалізована велика кількість звітів – фінансових, управлінських, маркетингових. Система дозволяє створювати нові звіти під потреби користувачів;

– користувачі мають доступ лише до того функціоналу IMED, що потрібен їм для виконання щоденної роботи;

– IMED інтегрується з іншими інформаційними системами (в тому числі лабораторіями), сервісами doc.ua, likarni.com, IP-телефонією, календарем Гугл, порталами Бітрікс24;

– система дозволяє підключення торгового та лабораторного обладнання, необхідного для використання в медичному закладі;

– служба підтримки допоможе вирішити всі питання, що можуть виникнути під час роботи з MIC IMED [18].

1.7 Висновки за розділом 1

Кількість МІС різного спрямування в медичній сфері поступово зростає, однак більшість з них ще на етапі доопрацювань і не можуть забезпечити повний функціонал [19].

Декілька хвиль пандемії акцентували питання щодо розвитку та модернізації у МІС таких функцій як телемедицина. На сьогодні така функція присутня тільки у SimplexMed. Розвиток цієї функції дозволить позбавити контактів хворих пацієнтів з лікарем, що має значно зменшити темпи захворюваності у періодах пандемії, наприклад, зберегти людське життя від поточної пандемії грипу COVID-19 (або SARS-CoV-2).

Зовсім недавно, тільки що з 08 жовтня 2021 року, Міністерство охорони здоров’я України розпочало роботу над розвитком телемедичних технологій і послуг в Україні. Телемедичні технології та сервіси здатні суттєво покращити доступ до медичних послуг для маломобільних груп населення, а також тих пацієнтів, які проживають у віддалених регіонах країни. Завдяки впровадженню телемедичних технологій і сервісів лікарі матимуть змогу проводити консультації з пацієнтами дистанційно, швидко обмінюватися інформацією, проводити консиліуми у вигляді відеоконференцій та оперативно ухвалювати рішення в критичних ситуаціях. Окрім того, в умовах боротьби з пандемією COVID-19 онлайн-інструменти можуть вирішувати нові важливі завдання. А саме – гарантувати безпечний контакт лікарів та пацієнтів, не обмежуючи останніх у своєчасному та якісному наданні медичних послуг. Електронна система охорони здоров’я вже сьогодні застосовує частину з цих сервісів у деяких областях України і

показує свою ефективність. Зокрема, із запуском телемедичних технологій в Одеській області не лише знизилися показники інфарктів поза терапевтичним вікном, але й покращилася якість їх лікування у цілому [20].

2 ВИЗНАЧЕННЯ ВИМОГ ТА ОБҐРУНТУВАННЯ СТРУКТУРИ ДИСТАНЦІЙНОЇ СИСТЕМИ, ЩО ПРОЄКТУЄТЬСЯ

Після проведення у 1 розділі пояснювальної записки детального аналізу медичних інформаційних систем в Україні було виявлено загальні недоліки більшості існуючих на даний час інформаційних систем, а саме:

– відсутність можливості дистанційного спілкування між лікарем та пацієнтом, що має назву телемедицина, де пацієнт зміг би у будь-який час передати до лікаря актуальні біометричні дані, наприклад, поточну температуру тіла, показники артеріального тиску та інші симптоми, а лікар у свою чергу зміг би не вводити ці дані повторно, а використовувати вже введені пацієнтом дані з інформаційної бази;

– відсутність сортування звернень пацієнтів до лікаря за показником ургентності, що може бути вкрай необхідним у періоди пандемії грипу, коли кількість звернень до лікаря збільшується у десятки разів, а лікар не в змозі обрати того пацієнта, котрий потребує найшвидших призначень.

Згідно технічного завдання (додаток А) необхідно дослідити вплив основних симптомів захворювань (грип та ГРВІ) на рівень ургентності пацієнта та виконати програмну реалізацію дистанційної системи консультацій сімейного лікаря (надалі – дистанційної системи) з урахуванням цього рівня. Ця система повинна дозволити пацієнтові звертатися до свого сімейного лікаря в не залежності від того чи знаходиться лікар онлайн.

Розроблювана дистанційна система дозволить виконувати наступні функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації поточної температури тіла та артеріального тиску, а також основних симптомів захворювань;

– зберігання показників схильності пацієнтів для подальшого обчислення рівню ургентності;

– можливість для пацієнтів залишати у зверненні коментарі щодо стану свого здоров’я;

– моніторинг лікарем як відкритих звернень пацієнтів, так і завершених;

– моніторинг пацієнтом як відкритих, так і завершених до лікаря звернень;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні;

– можливість для лікаря проводити пошук у базі звернень за різними категоріями та біометричними даними пацієнтів.

2.1 Створення структурної схеми дистанційної системи

Розроблювана дистанційна система складається із частин, які повинні взаємодіяти одна з одною. Розділення на частини вказує на модульність структури, що у майбутньому дозволить легко модернізувати програмне забезпечення чи виконати заміну будь-якого модулю у ньому.

У цілому дистанційна система складається із п’яти частин:

– клієнтська частина (Пацієнт), що призначена для пацієнта, який вносить дані у звернення до лікаря;

– клієнтська частина (Лікар), що призначена для сімейного лікаря, який виконує обробку звернень пацієнтів до нього;

– серверна частина, яка виконує функцію віддаленого серверу для зберігання звернень та обчислення рівня ургентності пацієнтів;

– база даних, яка містить інформацію щодо звернень пацієнтів, їх симптомів захворювань, схильностей та основних біометричних показників таких як, поточна температура тіла і артеріальний тиск;

– модуль аналітики, який виконує обчислення рівня ургентності пацієнтів для впорядковування звернень.

База даних та модуль аналітики знаходяться на серверній частині дистанційної системи.

Структурну схему розроблюваної дистанційної системи зображено на рисунку 2.1.

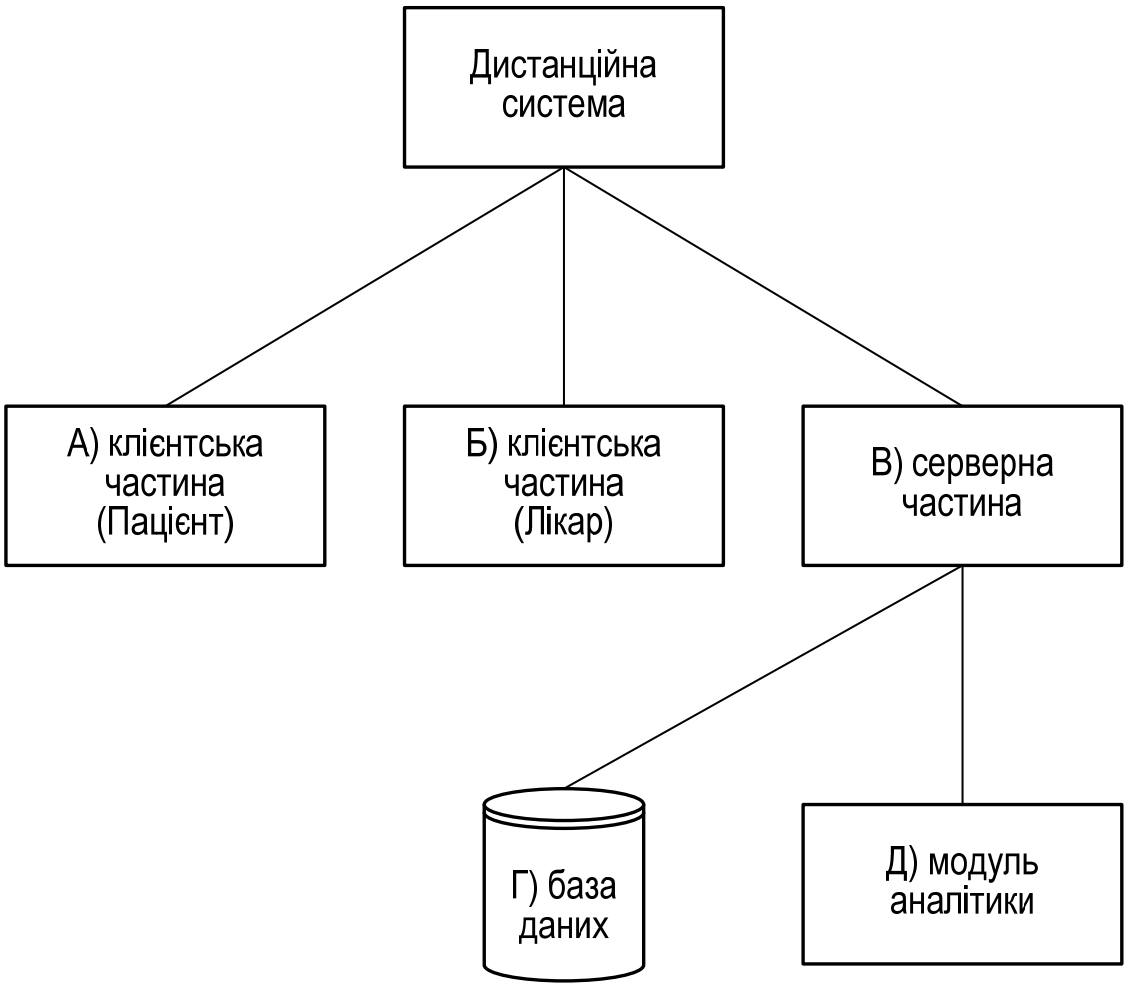


Рисунок 2.1 – Структурна схема дистанційної системи

2.2 Розробка функціональної схеми дистанційної системи

Функціональна схема дистанційної системи відображає склад модулів програмного забезпечення та деталізує процес передачі даних від одного модуля до іншого.

Функціональна схема складається із серверної частини та двох клієнтських частин – для пацієнта і для лікаря відповідно.

Кожен сімейний лікар має низку пацієнтів, які можуть отримувати медичну допомогу. Система призначена для багатьох лікарів, у кожного з яких може бути дуже багато пацієнтів.

На функціональній схемі зображено Адміністратора системи, якого не відокремлено до окремого модуля, тому що він діє поза інтерфейсами та за допомогою скриптів проводить первинну реєстрацію лікарів, виконує системні налаштування бази даних і модуля обчислення ургентності пацієнта, які знаходяться на головному сервері (рис. 2.2).

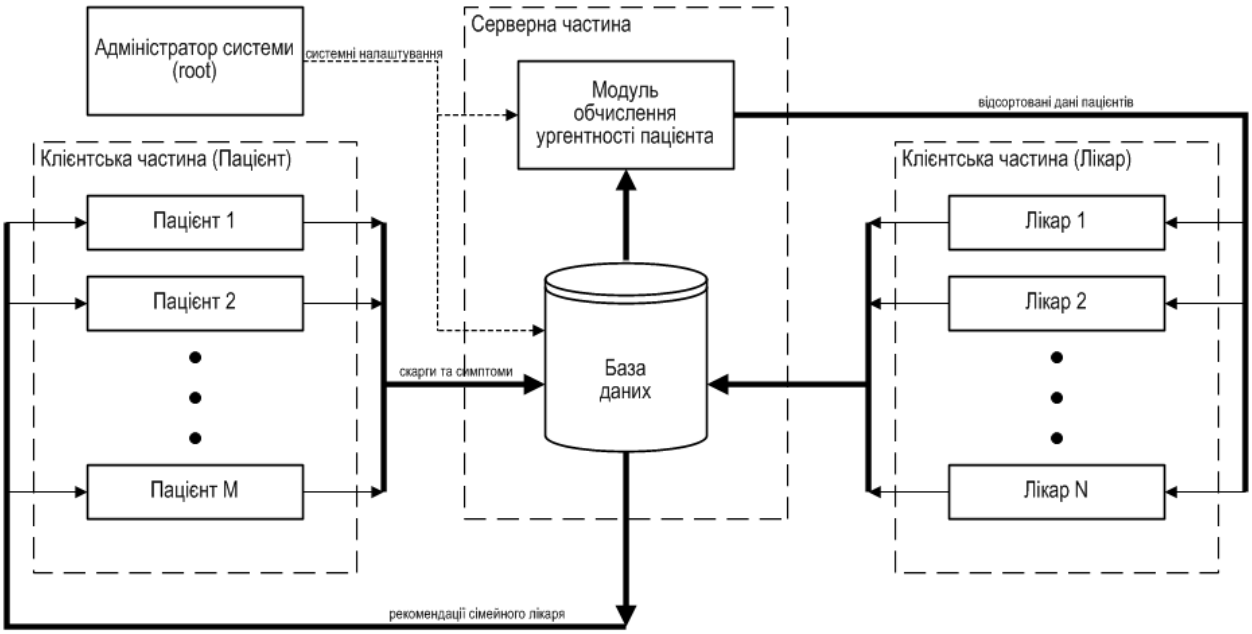


Рисунок 2.2 – Функціональна схема дистанційної системи

Якщо будь-хто з пацієнтів почуває себе недобре, то вони можуть у дистанційній системі створити звернення до сімейного лікаря. Новостворене звернення потрапляє до бази даних серверної частини, надалі оброблюється модулем обчислення рівня ургентності, у якому враховується не тільки різниця між поточною та нормальною температурою тіла та відхилення артеріального тиску від норми відповідного пацієнта, але й схильність конкретного пацієнта до деяких хвороб, що може значно вплинути на

підвищення уваги сімейного лікаря до відповідного пацієнта. Відсортована низка даних пацієнтів потрапляє до відповідного сімейного лікаря, і лікар може в першу чергу звернути увагу саме на того пацієнта, який потребує допомоги як найскоріше.

Сімейний лікар відповідає пацієнтові своїми рекомендаціями та призначеннями. Рекомендації від лікаря потрапляють на сервер, де будуть перенаправлені до відповідного пацієнта.

Усі дані звернень, у тому числі архівні, зберігаються у БД серверної частини дистанційної системи.

2.3 Розробка способу обчислення рівня ургентності пацієнта

Для обчислення рівня ургентності пацієнта необхідно враховувати саме ті дані, які як найбільше впливають на швидкість розвитку хвороби. Це має відношення до особливо небезпечних хвороб, наприклад, COVID-19 (або SARS-CoV-2) чи інших, які викликають тяжкі ускладнення саме у тих пацієнтів, які мають відповідні схильності.

За результатами аналізу епідемічної ситуації в Україні на теперішній момент показник захворюваності на грип та ГРВІ зростає за експоненціальною характеристикою в залежності від віку пацієнта [21]. Тому слід врахувати вплив віку пацієнта на півень ургентності.

За недавні два роки найбільшої шкоди здоров'ю людства завдав вірус грипу SARS-CoV-2. Його дія схожа на дії інших штамів грипу, тому можна використати відому на сьогодення симптоматику.

А. Рецидивна форма грипу або ГРВІ:

1) різка зміна артеріального тиску (навіть у здорової людини);
2) інші больові відчуття (найчастіше у шлунку);
3) діарея;
4) головний біль.

Якщо з’являться одночасно три симптоми – 1, 2, 3, то потрібно негайно викликати швидку. Слід враховувати цей факт для формування загального коефіцієнту впливу.

Б. Тяжка форма грипу або ГРВІ (первинні ознаки):

1) підвищення температури тіла;

2) сухий кашель;

3) затруднене дихання;

4) очна біль, почервоніння або подразнення;

5) втрата координації або порушення розрахунку у свідомості;

6) нежить.

Якщо з’являються одночасно усі три симптоми – 1, 2, 3, то потрібно негайно викликати швидку для пацієнта. Слід врахувати цей факт для формування загального коефіцієнту впливу.

В. Тяжка форма грипу або ГРВІ (вторинні ознаки):

1) втрата нюхових або смакових відчуттів;

2) спонтанний висип на різних місцях шкіри;

3) зміна кольору шкіри (блідість обличчя).

Якщо з’являється будь-який один симптом, декілька, або усі три разом, то потрібно терміново звернутися до сімейного лікаря. Слід врахувати цей факт для формування загального коефіцієнту впливу, але з меншим впливом.

Г. Легка форма грипу або ГРВІ:

1) зниження температури тіла;

2) біль у горлі або ознаки отиту;

3) періодична біль у серці;

4) втомленість.

Якщо з’являються одночасно усі три симптоми – 1, 2, 3, то потрібно терміново звернутися до сімейного лікаря. Слід врахувати цей факт для формування загального коефіцієнту впливу, але з меншим впливом.

У загальному випадку можна вивести нескладну формулу впливів різних першоступеневих та другеступеневих факторів на рівень ургентності пацієнта (2.1):

$$K_U = K_Y + \sum_{i=1}^{3} K_{Pi} + \sum_j C_{Xj} + K_A \cdot \sum_k S_{Ak} + K_B \cdot \sum_l S_{Bl} + K_C \cdot \sum_m S_{Cm} + K_D \cdot \sum_n S_{Dn}, \quad (2.1)$$

де K_Y – коефіцієнт впливу віку пацієнта;

K_{P1} – коефіцієнт впливу перевищення температури тіла;

K_{P2} – коефіцієнт впливу зміни систолічного артеріального тиску;

K_{P3} – коефіцієнт впливу зміни діастолічного артеріального тиску;

C_{Xj} – коефіцієнт схильності пацієнта до відповідної хвороби;

S_{Ak} – коефіцієнт групи симптомів рецидивної форми хвороби;

K_A – коефіцієнт посилення групи рецидивних симптомів;

S_{Bl} – коефіцієнт групи симптомів первинних ознак тяжкої форми;

K_B – коефіцієнт посилення групи первинних ознак тяжкої форми;

S_{Cm} – коефіцієнт групи симптомів вторинних ознак тяжкої форми;

K_C – коефіцієнт посилення групи вторинних ознак тяжкої форми;

S_{Dn} – коефіцієнт групи симптомів легкої форми хвороби;

K_D – коефіцієнт посилення групи симптомів легкої форми.

Так як вплив віку на захворюваність має зростаючу експоненціальну характеристику, то пропонується розраховувати коефіцієнт впливу віку пацієнта за наступною формулою (2.2):

$$K_Y = 1.05^y, \quad (2.2)$$

де y – вік пацієнта, повних років життя.

Коефіцієнт впливу перевищення температури тіла K_{P1} обирається з таблиці 2.1.

Таблиця 2.1 – Залежність коефіцієнту впливу від температури

Перевищення поточної температури тіла над потенційно нормальною *dT*, °C	Значення коефіцієнту впливу K_{P1}
≥ 0.8	1.0
≥ 1.4	2.0
≥ 2.0	3.0
≥ 2.6	4.0

Перевищення поточної температури тіла над потенційно нормальною *dT* розраховується як різниця між поточними замірами, які передані у зверненні до лікаря та даними, які зберігаються у базі даних для відповідного пацієнта.

Коефіцієнти впливу зміни артеріального тиску систолічного K_{P2} та діастолічного K_{P3} обираються з таблиці 2.2.

Таблиця 2.2 – Залежність коефіцієнтів впливу від зміни тиску

Тип артеріального тиску	Перевищення артеріального тиску мм. рт. ст.	Коефіцієнт	Значення коефіцієнтів впливу
Систолічний (*dS*)	≥ 15.0	K_{P2}	1.5
Діастолічний (*dD*)	≥ 10.0	K_{P3}	2.0

Перевищення артеріального тиску систолічного K_{P2} та діастолічного K_{P3} над відповідними потенційно нормальними значеннями розраховується як різниця між поточними замірами, які передані у зверненні до лікаря та даними, які зберігаються у базі даних для відповідного пацієнта.

Якщо перевищення будь-якого з параметрів *dT*, *dS*, *dD* відсутнє, то відповідні їм коефіцієнти K_{P1}, K_{P2}, K_{P3} будуть дорівнювати нулю.

Розрахунок інших коефіцієнтів формули 2.1 буде мати залежність від вмісту довідкових таблиць бази даних.

2.4 Концептуальне проєктування бази даних

Для зберігання звернень пацієнтів, рекомендацій сімейних лікарів та даних, які використовуються для розрахунку рівня ургентності, необхідно розробити базу даних. Для реалізації цієї мети найкраще підійде використання двох наступних моделей даних: концептуальної моделі та реляційної моделі [22].

Завжди створення будь-якої бази даних має впорядковану послідовність із наступних етапів:

– первинний етап концептуального проєктування;

– етап або етапи логічного проєктування;

– заключний етап фізичного проєктування [22].

За час концептуального проєктування закінчується оформлення задумів майбутньої бази даних, але при цьому не повинні враховуватися фізичні аспекти її реалізації. На даному етапі проєктування розробника не повинно цікавити ні конкретна СКБД, яка буде використана для створення БД, ані мови програмування, які буде використано для створення застосувань, ні фреймворки, ні технології, ані особливості апаратних платформ. По-перше потрібно створити загальну модель, яка має відображати уявлення майбутніх користувачів бази даних. Усю для цього необхідну інформацію вже потрібно бути зібрати на попередньому етапі життєвого циклу бази даних [23].

Основним засобом створення концептуальної моделі БД має виступати модель «сутність-відношення» (ER-модель). Такі моделі мають містити прості та водночас ефективні методики, які дозволяють уявляти сенс всього того, що має підлягати зберіганню у БД. На заключному етапі концептуального проєктування розробник бази даних обов’язково повинен перевірити адекватність створеної моделі, і для цього процесу необхідно організувати обговорення отриманих результатів (елементів ER-діаграм) із замовником розробки. Якщо на цьому етапі були виявлені невідповідності, то

до моделі потрібно негайно внести виправлення. Процеси порівняння припиняються тільки тоді, коли всі майбутні користувачі підтвердять коректність концептуальної моделі. Адже, кінцевим результатом етапу концептуального проєктування буде ER-модель майбутньої бази даних, яка має містити опис:

– сутностей;

– відношень між сутностями;

– атрибутів (із описом варіацій та обмежень);

– первинних ключів [24].

Для предметної області дистанційної системи було визначено наступні сутності:

– «Пацієнт»;

– «Персональні дані пацієнта»;

– «Карта вакцинацій»;

– «Схильність»;

– «Лікар»;

– «Персональні дані лікаря»;

– «Графік прийому»;

– «Історія хвороби»;

– «Сеанс»;

– «Симптом».

Кожна визначена сутність має відповідні атрибути:

– «Пацієнт»:

1) код;

2) П.І.Б.;

3) фото;

4) дата народження;

5) стать;

6) нормальна температура;

7) нормальний систолічний тиск;

8) нормальний діастолічний тиск;

9) ступінь ургентності;

– «Персональні дані пацієнта»:

1) код;

2) логін;

3) пароль (парольний ґеш);

4) серія паспорту;

5) номер паспорту;

6) ким видано паспорт;

7) країна реєстрації;

8) область реєстрації;

9) місто реєстрації;

10) вулиця реєстрації;

11) дім реєстрації;

12) корпус реєстрації;

13) квартира реєстрації;

14) телефон 1;

15) телефон 2;

16) номер телефону для Viber;

17) посилання для Telegram;

18) адреса електронної пошти;

– «Карта вакцинацій»:

1) код;

2) дата вакцинації;

3) опис вакцинації;

– «Схильність»:

1) код;

2) найменування;

3) коефіцієнт;

– «Лікар»:

1) код;

2) П.І.Б.;

3) фото;

4) область прийому;

5) місто прийому;

6) департамент прийому;

7) вулиця департаменту;

8) дім департаменту;

9) корпус департаменту;

10) кабінет прийому;

– «Персональні дані лікаря»:

1) код;

2) логін;

3) пароль (парольний ґеш);

4) телефон 1;

5) телефон 2;

6) номер телефону для Viber;

7) посилання для Telegram;

8) адреса електронної пошти;

– «Графік прийому»:

1) код;

2) дата прийому;

3) час початку;

4) час закінчення;

5) кабінет;

– «Історія хвороби»:

1) код;

2) номер;

3) дата та час початку;

4) номер лікарняного;

5) дата початку лікарняного;

6) дата кінця лікарняного;

7) заключний діагноз;

8) ознака відкриття історії хвороби;

– «Сеанс»:

1) код;

2) дата та час звернення;

3) поточна температура;

4) поточний систолічний тиск;

5) поточний діастолічний тиск;

6) додатковий текст;

7) рекомендації лікаря;

– «Симптом»:

1) код;

2) найменування;

3) коефіцієнт.

Дистанційна система проєктується як багатокористувацька та призначена, як для сімейних лікарів медичних клінік щодо обробки звернень пацієнтів так і для самих пацієнтів щодо створення звернень до своїх сімейних лікарів з симптомами захворювання та зворотнім зв’язком.

В результаті концептуального проєктування було створено модель бази даних, яку зображено на ER-діаграмі (рис. 2.3).

У розробленій концептуальній моделі визначено відношення між сутностями «Пацієнт» та «Лікар». Так як один сімейний лікар має можливість обслуговувати багато пацієнтів, але й самі пацієнти мають можливість змінювати свого сімейного лікаря, то відношення між сутностями «Пацієнт» і «Лікар» визначено як багато-до-багатьох. Для реалізації у майбутньому цього відношення на реляційних системах керування базами даних, зв’язок виконано через сутність «Історія хвороби».

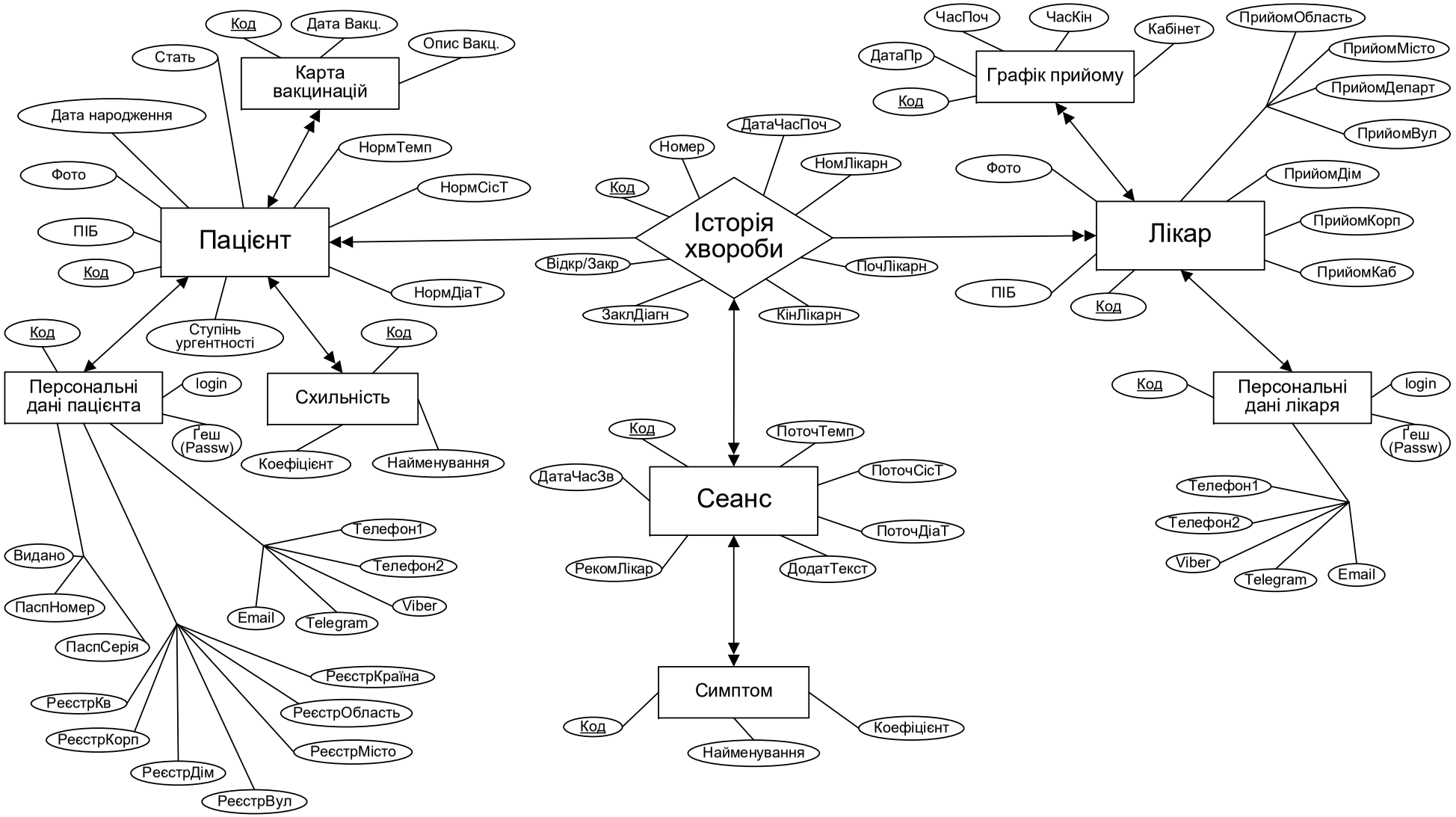


Рисунок 2.3 – Концептуальна модель бази даних

Сутність «Історія хвороби» зв’язує сутності «Пацієнт» і «Лікар» об’єднуючи разом усі атрибути, які відносяться щодо процесу звернення пацієнта до лікаря. Історія хвороби складається із низки звернень, поки пацієнт не одужає. Тому ця низка звернень і представляє собою сутність «Сеанс» з відношенням від «Історії хвороби» до «Сеансу» як один-до-багатьох. Сутність «Сеанс» визначає саме звернення пацієнта до сімейного лікаря або ж відповідь лікаря з рекомендаціями. У кожному зверненні пацієнт може вказати декілька симптомів. Тому відношення від сутності «Сеанс» до сутності «Симптом» як один-до-багатьох.

Для розгалуження доступу щодо збереження персональних даних і захисту інформації від стороннього втручання, як для пацієнта так і для лікаря були створені сутності «Персональні дані пацієнта» та «Персональні дані лікаря». Відношення між сутностями «Пацієнт» та «Персональні дані пацієнта» визначено як один-до-одного. Відношення між сутностями «Лікар» та «Персональні дані лікаря» визначено теж як один-до-одного.

2.5 Логічне проєктування бази даних

На стадії логічного проєктування повинно виконуватись перетворення узагальненої концептуальної моделі до завершеної логічної моделі. Обов’язково завершується уточнення усіх вимог, які було виявлено на етапі концептуального проєктування та виконується спрощення рішення, але не знижуючи функціональних можливостей цих вимог. Для цього концептуальна модель має бути перевірена за допомогою правил нормалізації. В результаті повинно бути отримано не надлишкові реляційні таблиці, які мають бути вільні від аномалій – вставок, редагування та видалення [24].

Крім нормалізації, на етапі логічного проєктування мають бути здійснені наступні дії:

– уточнення обмежень на дані;

– визначення доменів даних;

– уведення бізнес-правил і корпоративних обмежень цілісності;

– визначення місць розташування для майбутніх таблиць [24].

Потім потрібно повторне проведення звіряння логічної моделі з участю майбутніх користувачів бази даних і замовниками проєкту. Проводиться внесення останніх уточнень та виправлень, і усі учасники проєктної групи разом з замовниками проєкту узгоджують єдину думку стосовно створюваної БД [23].

Впродовж концептуального і логічного проєктування розробники звично використовують одну з двох стратегій створення бази даних: висхідне проєктування (знизу-догори) чи низхідне проєктування (згори-донизу). Висхідний підхід зазвичай застосовується для малих проєктів. Суть цього методу в тому, що проєктувальник разом із замовником бази даних створюють повний список тих атрибутів, які мають підлягати зберіганню. Наступним етапом атрибути групуються до типів сутностей, що потрапляють до моделі. Цей процес нормалізації засновано на висхідному підході. Зворотній висхідному, низхідний підхід найкраще підійде щодо середніх і великих проєктів [23].

Проєктування має починатися з виявлення основних типів сутностей але тільки тоді сутності мають «наповнитися» атрибутами. Класичний приклад низхідного методу – модель «сутність-зв’язок» (або ER-модель). У загальному випадку необхідно спільно застосувати як висхідну так і низхідну методології (на даному етапі – нормалізацію таблиць і ER-модель) [23].

2.6 Нормалізація

Процес нормалізації переслідує декілька цілей. Найважливіша з них – це боротьба із надмірністю даних. Досягнути усунення надмірності можливо не тільки за рахунок раціонального призначення розмірностей полям

таблиць, але й за рахунок формування ефективних взаємопов’язаних табличних структур, які об’єднують декілька відношень. Адже, процес нормалізації має статус основної частини з процесів проєктування БД [24].

В результаті процесу нормалізації БД, таблиці повинні прийняти додаткову гнучкість, що дозволить без наслідків змінювати структуру таблиць у майбутньому, наприклад, в процесі модернізації існуючої бази даних. Перемога у боротьбі за надлишковість даних не лише призведе до мінімізації витрат щодо зберігання даних, але також усуне низку непрямих проблем чи аномалії, які властиві некоректно побудованим таблицям. До цих проблем можна віднести аномалії вставки, редагування та видалення даних. Щодо усунення аномалій доводиться проводити нормалізацію даних. Нормалізація – це процес послідовного перетворення таблиць бази даних до вигляду, який прийнято у реляційній моделі даних [25].

Загалом розрізняються п’ять послідовних етапів або форм приведення даних до реляційного виду з першої нормальної форми (First Normal Form, 1NF) до п’ятої (5NF). На кожному з етапів нормалізації, таблиця має прийняти нові риси, які необхідні щодо переходу до наступного етапу. Адже, щоб дійти до 5NF, потрібно крок за кроком на кожному з етапів методично удосконалити структуру таблиць. Пропустити хоч би один з етапів нормалізації не має можливості [26].

Таблиця стає приведеною до першої нормальної форми, якщо у ній відсутні групи, які повторюються та значення, що зберігаються є атомарними (неділимі). У результаті приведення до 1NF у цілях боротьби з повторюваними групами необхідно спростити класифікацію схильностей та симптомів, обмеживши єдиним значенням відповідно до кожної групи. У процесі проєктування таблиць, розробник має намагатися, щоб інформація, яка буде зберігатися у ній, стала не тільки не надлишковою, але й раціонально взаємопов’язаною. Ця раціональність у таблицях, які нормалізовані до 1NF частіше відсутня. Тому на етапі доведення таблиць до

вищих нормальних форм, атрибути відношення потрібно перевірити на функціональну залежність між собою [24].

Таблиця буде відповідати другій нормальній формі, якщо вона вже була приведена до 1NF і в ній відсутні часткові залежності. Інакше, у цій таблиці не повинно бути атрибутів, які залежні тільки від частки первинного ключа. Відмова від складеного первинного ключа має одну безперечну перевагу – спрощується процес приведення таблиці до 2NF. Якщо первинний ключ не є складений, то не буде і часткових залежностей. Залишається лише провести доопрацювання таблиць. Тепер в таблицях повинні з'явитися поля-лічильники. Опісля приведення до 2NF між полем потенційного ключа і будь-яким іншим полем таблиці повинна встановитися однозначна відповідність – усунення часткових залежностей. Відмова від складеного ключа – це найбільш просте, але не єдине рішення задачі приведення відношення до 2NF. Можливо розірвати початкову таблицю на декілька таблиць, щоб атрибути з частковою залежністю і елементи складеного ключа розійшлися на різні відношення. Таблиця буде приведеною до другої нормальної форми, якщо вона відповідає 1NF і в ній відсутні часткові залежності, тобто відсутні не ключові поля, які можуть залежити від частини первинного ключа [25].

На наступному етапі нормалізації за рахунок відмови від складеного ключа необхідно буде позбавлятися від часткових залежностей. Але можуть залишитися транзитивні залежності. Транзитивною залежністю між атрибутами відношення є, коли один з атрибутів пов'язаний з іншим через атрибут-посередник A-B-C. У даному випадку перетворення таблиці з 2NF до 3NF призводить до декомпозиції початкового відношення. Таблиці, які не мають залежності ні від яких інших таблиць, називають «довідковими», оскільки вони є джерелами довідкових даних для підлеглих за відношенням до них таблиць [25].

Довідкові таблиці корисні, завдяки чому:

– істотно скорочено фізичний розмір підлеглої таблиці. Замість реальних даних, які займають більш десятків байт, в підлеглій таблиці вимагається зберігати тільки значення зовнішнього ключа, що зазвичай не перевищує одного байту;

– щодо редагування значень відсутня необхідність переглядати сотні записів але досить змінити одне значення у довідковій таблиці;

– при введенні даних значно знижена вірогідність появи помилки користувача, так як замість введення з клавіатури йому пропонується на вибір заздалегідь підготовлені значення [26].

Таблицю буде приведено до третьої нормальної форми, якщо вона відповідає 2NF і в ній будуть відсутні транзитивні залежності [24].

У випадку, якщо третя нормальна форма використовується для боротьби з транзитивними залежностями, то четверта нормальна форма (4NF) містить конфронтацію з іншим недоліком реляційної моделі – багатозначними залежностями. Багатозначні залежності – це відношення «багато-до-багатьох» [24].

База даних, яка має відповідати четвертій ступені нормалізації, зобов’язана бути позбавлена від багатозначних залежностей між атрибутами відношень. Щодо реалізації відношення «багато-до-багатьох» (M:N), то доведеться увести додаткову таблицю, яка має розділити відношення «M:N» до двох «1:N». Обидва створених нових відношення містять пару зовнішніх ключів щодо зв’язку з лівою та правою таблицями. Таблиця має бути приведена до четвертої нормальної форми, яка обов’язково відповідає 3NF та в ній відсутні багатозначні залежності («багато-до-багатьох»). У даній предметній області багатозначні залежності були між сутностями «Пацієнт» та «Лікар». Ці багатозначні залежності ліквідовано завдяки двом відношенням «один-до-багатьох» між «Пацієнт» і «Історія хвороби» та між «Лікар» і «Історія хвороби». Логічну схему бази даних розроблюваної дистанційної системи зображено на рисунку 2.4.

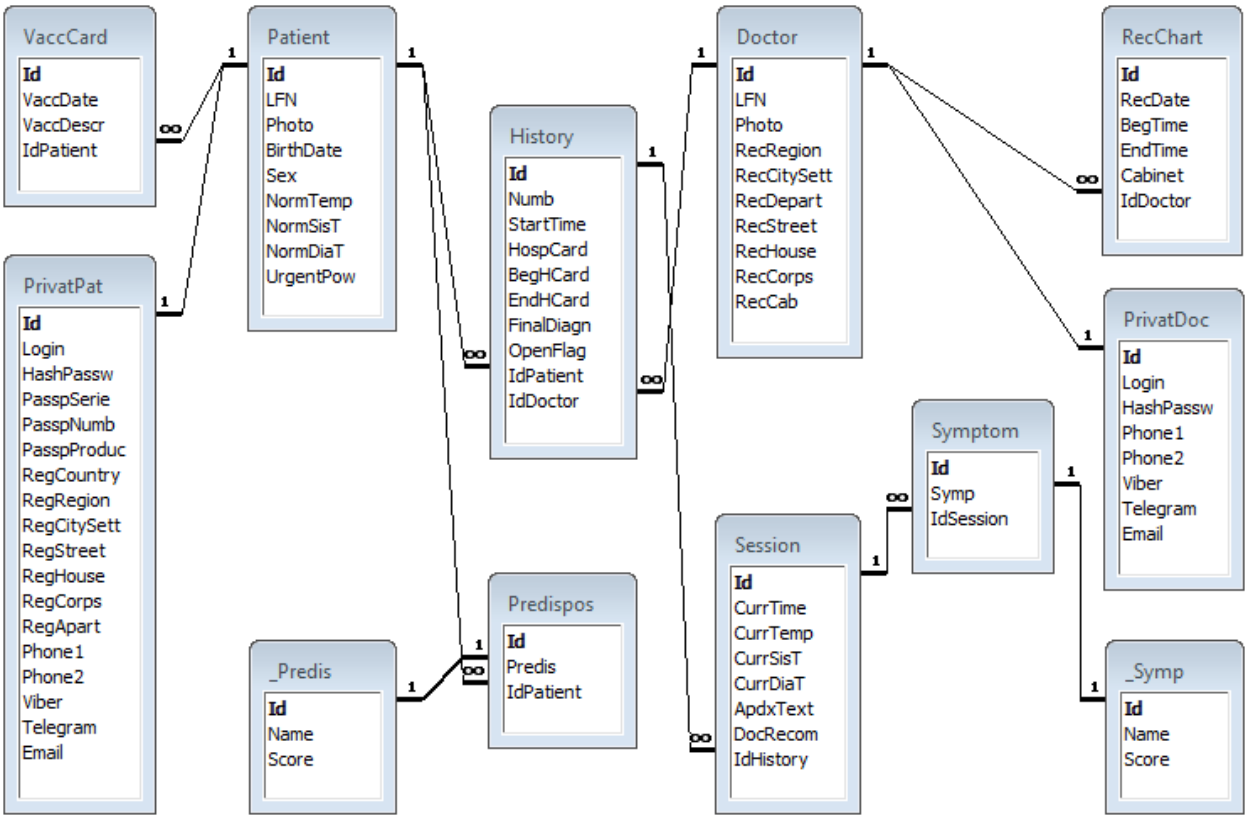


Рисунок 2.4 – Логічна схема реляційної бази даних дистанційної системи

Згідно правилам нормалізації, логічну схему реляційної бази даних розроблюваної дистанційної системи доведено до рівня 4NF, ліквідовані транзитивні залежності, до мінімуму спрощено структури таблиць та визначені зв’язки між таблицями.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ ДИСТАНЦІЙНОЇ СИСТЕМИ

3.1 Обґрунтування вибору середовища розробки

До важливих етапів фізичного проєктування бази даних переходять тільки після вибору цільової системи керування базами даних (СКБД), яка і буде надалі визначати особливості майбутньої програмної розробки. Визначитися та обрати СКБД необхідно до початку етапів фізичної розробки бази даних.

Етап фізичного проєктування – це процеси уточнення рішень з врахуванням існуючих розробників технологій, можливості її реалізації та необхідних продуктивностей. Впродовж процесів фізичного проєктування завданнями проєктувальників є перенесення логічної моделі бази даних до платформи цільової СКБД. Ця мета для розробника визначає виконання наступних дій:

– створення таблиць і зв’язків між таблицями;

– призначення вторинних індексів у таблицях;

– реалізація бізнес-логіки для даної бази даних;

– визначення функціональних характеристик для транзакцій;

– розробка представлень для даної бази даних;

– обов’язкове упровадження механізмів захисту до бази даних, що має передбачати авторизацію користувачів і призначення правил доступу до всієї інформації у базі даних.

Найважливішим для розробника на цьому етапі є визначення призначених для користувача типів даних, створення описів таблиць і зв’язків між ними, встановлення порядку підтримки бізнес-логіки, організація призначень послідовності викликів низки процедур, що зберігаються та створення необхідних тригерів [23].

3.2 Вибір системи керування базами даних

Якість і продуктивність функціонування бази даних безпосередньо пов’язане з СКБД, яку буде застосовано щодо реалізації проєкту. Адже, обрання відповідної платформи багато в чому визначить переваги та недоліки майбутньої бази даних. Для оцінювання СКБД існує великій набір критеріїв, проаналізувавши які проєктувальник зможе вдало зупинити свій вибір на конкретному програмному продукті [27].

На найпершому кроці обов’язково визначається тип моделі даних. Проаналізувавши вже створену ER-модель (рис. 2.3) і розроблену логічну схему (рис. 2.4) бази даних дистанційної системи, було прийнято рішення використовувати реляційну модель даних.

Систему керування базою даних застосовують щодо забезпечення доступу до інформації, яка зберігається у базі даних, а також для керування цією інформацією. СКБД представляє собою комплекс мовних та програмних засобів, які призначено для створення, обслуговування і спільного використання бази даних багатьма користувачами. Найчастіше СКБД розрізняються за використаною моделлю даних. Системи керування базами даних, які базуються на використанні реляційних моделей даних, називають реляційними СКБД [26].

Можливості та функції будь-якої СКБД можуть істотно відрізнятися. Різноманітні СКБД по-різному документовані, – одні більш ретельно, інші менш ретельно. Технічна підтримка надається теж по-різному. У процесі порівняння популярних систем керування базами даних необхідно врахувати, чи буде зручна користувачам ця конкретна СКБД, а також переконатися, що вона має бути масштабована та добре інтегруватися з іншими програмними продуктами, які вже використовуються. Крім того, під час обрання СКБД обов’язково слід брати до уваги вартість системи і вартість технічної підтримки, яка має надаватися розробником [27].

На сьогодення існує декілька найпопулярніших СКБД, як безкоштовних, так і за гроші. Ці системи можна рекомендувати для використання у розроблюваній дистанційній системі консультацій. В процесі виконання пошуку найкращої із СКБД, розглядається перелік, який наведено нижче.

Microsoft SQL-сервер.

Однією з відомих та найпопулярніших СКБД є програмний продукт від відомої однойменної компанії Microsoft SQL-сервер. Дана система керування базами даних, ядро якої функціонує не тільки на хмарних, але й на локальних серверах. При цьому присутня особливість одночасного комбінування типів використовуваних серверів. Після випуску Microsoft SQL-сервер 2016 компанія Microsoft адаптувала продукт для операційної системи Linux.

Одна з унікальних особливостей версії 2016 року – це тимчасова підтримка даних (Temporal Data Support), що дозволить відстежити зміни даних впродовж часу використання. Ця версія Microsoft SQL-сервер підтримує функцію динамічного маскування даних (Dynamic Data Masking), що гарантує приховування конфіденційних даних.

Переваги:

– ядро системи надає можливість відстежувати і регулювати рівні продуктивності, що допомагає у зниженні використання ресурсів;

– розширений доступ щодо візуалізації на мобільних пристроях;

– покращені швидкість і стабільність порівняно з ранніми версіями;

– зручність та легкість у використанні;

– беззаперечна коректність взаємодії з іншими продуктами Microsoft.

Недоліки:

– виявлено проблеми з використанням служби інтеграції щодо імпорту файлів;

– нераціональна потреба у використанні системних ресурсів;

– надвелика вартість програмного продукту щодо юридичних осіб;

– застосування може ідеально підійти тільки для великих організацій, які вже використовують низку продуктів від Microsoft.

MySQL.

На сьогодення це найпопулярніша система керування базами даних для вебзастосунків. Ця СКБД є фактичним стандартом для вебсерверів, які функціонують під операційною системою Linux. СКБД MySQL містить безкоштовний пакет програм, з постійним оновленням версій, які мають за мету розширити функціонал і покращити безпеку. В наявності є спеціальні версії за гроші, що призначені для комерційного використання. У безкоштовних версіях найбільший напрям націлено на швидкість та надійність, але тільки не на повноту функціоналу, який стає як перевагою так і недоліком, залежно від галузі застосування.

Розробку та підтримку MySQL з недавніх часів здійснює корпорація Oracle, яка має права на торгівельну марку разом із Sun Microsystems, що раніше придбала шведську компанію MySQL AB. Ця СКБД поширюється як під ліцензією GNU (General Public License), так і під власною комерційною ліцензією. Розробники постійно створюють та розширюють функціональність за замовленням ліцензійних користувачів. Завдяки таким замовленням вже в самих ранніх версіях з’явився механізм реплікації [28].

Дана система керування базами даних дозволяє обрати окреме ядро щодо системи зберігання даних, яке повинно надавати змінення функціоналу інструментів та виконувати обробку таблиць БД, які зберігаються у різних типах. Неперевершена гнучкість MySQL забезпечується підтримкою великої кількості типів таблиць. Завдяки відкритій архітектурі СКБД і GPL-ліцензуванню в MySQL постійно з’являються нові типи таблиць. Також ця СКБД має простий у використанні інтерфейс та пакетні команди, що дозволяють зручно обробляти величезні об’єми даних. Система керування базами даних MySQL має велику надійність і не має надмірної потреби щодо використання ресурсів [28].

Переваги:

– дуже велика кількість функцій, також у безкоштовній версії;

– підтримка наборів інтерфейсів, призначених для користувача;

– наявність пакету MySQL у стандартних репозиторіях багатьох дистрибутивів операційних систем Linux;

– підтримка сумісності з іншими БД, включно DB2 і Oracle;

– безкоштовність програмного продукту;

– гарна документованість [28].

Недоліки:

– відсутність вбудованої підтримки XML або OLAP;

– складність щодо створення інкрементних резервних копій;

– безкоштовна версія має тільки платну підтримку [28].

PostgreSQL 11.

Система керування базами даних PostgreSQL є одним з декількох популярних безкоштовних варіантів СКБД, які найчастіше використовуються щодо підтримки баз даних вебсайтів. Це була одна із перших розроблених СКБД, тому на сьогодні вона добре розвинена та дозволяє користувачам керувати як структурованими, так і неструктурованими даними. Цю СКБД можливо використовувати на більшості основних платформ, включно Linux. Вона відмінно справляється із завданнями імпорту інформації з інших типів баз даних за допомогою власного інструментарію [29].

Ядро цієї СКБД може бути розташовано у ряді середовищ, у тому числі також і віртуальних, і фізичних, і хмарних. Найсвіжіші версії пропонують обробку великих об’ємів даних і збільшення кількості одночасно працюючих користувачів.

Переваги:

– велика кількість зумовлених функцій;

– гарна масштабованість і здатність обробляти терабайти даних;

– велика низка доступних інтерфейсів;

– підтримка формату даних JSON.

Недоліки:

– зниження швидкості роботи під час проведення пакетних операцій або виконання запитів читання;

– надважка структура конфігурації;

– не деталізована документація;

– ідеально підходить для організацій з обмеженим бюджетом але з кваліфікованими фахівцями.

Oracle 12.

Першу версію Oracle було створено в кінці 70-х років. Актуальну версію Oracle призначено для хмарних середовищ з можливістю розташування на одному або декількох серверах, які дозволяють керувати базами даних, що містять мільярди записів. Деякі з функцій актуальної версії Oracle містять технології Grid Framework і використання як фізичних, так і логічних структур. Фізичне керування даними у цієї СКБД не впливає на доступ до логічних структур. Систему безпеки в цій версії доведено до найвищого рівня завдяки тому, що кожну транзакцію ізольовано від інших.

Переваги:

– дуже гарна надійність;

– впровадження найсвіжіших інновацій та збільшення функціоналу.

Недоліки:

– потреба значних обчислювальних ресурсів;

– дуже велика вартість продукту, особливо для малих організацій.

SAP HANA.

Система керування базами даних, яку розроблено компанією SAP SE з ядром, яке орієнтовано на роботу із стовпцями. Ця СКБД працює не тільки з рідними даними SAP, а також із сторонніми даними. Ядро призначене для збереження та отримання даних із застосувань, а також з інших джерел на декількох рівнях зберігання. Дані цієї СКБД можуть бути розміщені на фізичних і хмарних серверах.

Переваги:

– більшість даних переважно зберігається у пам’яті, що значно скорочує час доступу в деяких випадках;

– ядро знижує вимоги щодо ресурсів за рахунок використання прогресивного стиснення;

– у наявності підтримка SQL, OLTP і OLAP;

– взаємодія з низкою сторонніх застосувань;

– формування звітів у реальному часі.

Недоліки:

– вимагається постійне оновлення;

– надвисока вартість ліцензій.

MongoDB.

Це безкоштовна система керування базами даних, яка додатково має комерційну версію. Дану СКБД призначено для застосувань, які використовують як структуровані, так і неструктуровані дані. Ядро цієї системи неймовірно гнучке та працює при підключенні баз даних до застосувань через драйвери MongoDB.

Спочатку систему MongoDB не було розроблено для обробки моделей реляційних даних (але була така можливість), але й тому можливе виникнення проблем продуктивності, якщо використовувати її таким чином.

Остання версія MongoDB має нову систему механізмів зберігання. Документи можуть бути перевірено в процесі оновлення або виконання вставок, але функції текстового пошуку було значно поліпшено. Здатність часткового індексування може підвищити продуктивність, зменшуючи розмір індексів.

Переваги:

– легкість та швидкість роботи з даними будь-якої структури;

– підтримка JSON та інших традиційних документів NoSQL;

– простота та швидкість у використанні.

Недоліки:

– трудомісткість процесу інсталяції та наладки;

– SQL не використовується як мова запитів;

– підходить для організацій, що працюють із різнорідними даними, які важко піддаються класифікації.

DB2.

Система керування базами даних, яка створена компанією IBM. Це СКБД, яка має можливості NoSQL, але також може читати JSON та XML-файли. Система розроблялася для серверів компанії IBM модельного ряду iSeries, функціонує на Windows, Linux та Unix.

Діалект мови SQL, який використовується у DB2 за рідкісними виключеннями строго декларативний. Цю систему забезпечено багатофазовим оптимізатором, який створює за цими декларативними конструкціями план виконання запитів. У SQL-діалекті DB2 відсутні підказки оптимізатору, мало розвинена мова процедур, що зберігаються.

У системі DB2 відсутні власні засоби автентифікації користувачів, які інтегруються із засобами операційної системи чи зі спеціалізованими серверами безпеки. В рамках DB2 здійснюється тільки авторизація користувачів, яких було автентифіковані системою.

Система керування базами даних DB2 – це єдина реляційна СКБД загального призначення, яка має реалізації на апаратно-програмному рівні.

Сучасні версії DB2 вільно забезпечують розширену підтримку використання даних у форматі XML, у тому числі операції з окремими елементами документів XML. Поточна версія пропонує різноманітні поліпшення та доопрацювання. Одне з них – прискорення Blu, яке призначене для поліпшення швидкодії цієї СКБД. Остання версія DB2 містить також забезпечення вдосконалених функцій аварійного відновлення, сумісності та аналітики.

Переваги:

– можливість для баз даних бути розміщеними у хмарних сховищах, на фізичних серверах або ж і там, і там одночасно;

– технологія Blu Acceleration дозволяє грамотно і чітко задіяти ресурси щодо великих та надвеликих баз даних;

– планувальник завдань дозволяє одночасне виконання декількох процесів;

– коди помилок та коди завершення планувальника завдань дозволяють легко відстежувати виконувані завдання.

Недоліки:

– для функціонування кластерів або декількох вторинних вузлів вимагається стороннє застосування або додаткове програмне забезпечення;

– надвелика вартість програмного продукту;

– базова безкоштовна підтримка доступна тільки впродовж трьох років;

– підходить тільки для великих організацій, які планують отримати максимум з наявних ресурсів та обробляти великі бази даних.

MariaDB.

Це безкоштовна СКБД з відкритим кодом, так само як і багато інших безкоштовних застосувань, пропонує також і платні версії. Існують безліч доступних користувачам доповнень та розширень. Це все завдяки тому, що дана система керування базами даних на даний момент швидко розвивається.

MariaDB фактично являється відгалуженням від СКБД MySQL, яка розроблюється співтовариствами під ліцензією GNU GPL. Розробкою та підтримкою MariaDB займається компанія MariaDB Corporation AB та фонд MariaDB Foundation. Важливим поштовхом щодо створення цього програмного продукту на противагу політиці ліцензування MySQL компанією Oracle, стала необхідність забезпечення вільного статуту СКБД. Система ліцензування сучасної MariaDB зобов’язує розробників, які додають свій код до основної гілці СКБД, обмінюватися своїми авторськими правами

з MariaDB Foundation щодо забезпечення ліцензії та можливостей створення критичних виправлень для MySQL.

Провідним розробником цієї СКБД являється Майкл Віденіус, який є також автором оригінальної версії MySQL і засновником компанії Monty Program AB.

Ядро цієї системи керування базами даних дозволяє виконувати вибір з декількох систем зберігання, що робить використання ресурсів більш оптимізованим і підвищує продуктивність запитів та їх обробки. До загального складу MariaDB включено підсистеми зберігання даних XtraDB щодо можливостей змінення основної підсистеми зберігання InnoDB. Також ця СКДБ містить підсистеми PBXT, Aria і FederateX. Вона має повну сумісність з MySQL і дуже добре підходить як її заміна, оскільки повністю відповідають як набір команд, так і API. До процесів розробки цієї СКДБ було залучено велику кількість розробників MySQL, які зараз також беруть участь у її розвитку.

Переваги:

– добра швидкодія системи;

– розширювана архітектура та наявність доповнень дозволяють налаштувати систему відповідно до потреб користувачів;

– індикація обробки запитів;

– наявність шифрування даних у мережі, на сервері та на рівні застосування.

Недоліки:

– стабільність системи на даний момент нижча, ніж у MySQL;

– підтримка не є безкоштовною;

– ідеальна альтернатива MySQL, якщо MySQL не влаштовує із якихось причин.

Всі розглянуті особливості кожної СКБД впорядковано у звідну таблицю 3.1.

Таблиця 3.1 – Порівняльний аналіз особливостей різних СКБД

Назва СКБД	Розробник	Тип	Операційна система	Початковий код	Ліцензія	Підтримка	Популярність	Стабільність
SQL Server	Microsoft	Реляційна	Linux, Microsoft Windows	Закритий	Комерційна	Безкоштовна	+	+
MySQL	Oracle Corporation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Відкритий	GNU GPL та комерційна	За гроші	+	+
PostgreSQL	PostgreSQL Global Development Group	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX, macOS, QNX	Відкритий	Вільне та відкрите програмне забезпечення, дозвільна ліцензія	За гроші	+	+
Oracle Database	Oracle Corporation	Мульти-модельна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX	Закритий	Комерційна	За гроші	+	+
SAP HANA	SAP SE	Реляційна, in-memory	Linux	Закритий	Комерційна	За гроші	–	+
MongoDB	MongoDB Inc.	Документо-орієнтована	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Відкритий	GNU AGPL (СКБД) та Apache License (драйвери)	За гроші	+	+
DB2	IBM	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Закритий	Пропрієтарна EULA	Безкоштовна	–	+
MariaDB	MariaDB Corporation AB, MariaDB Foundation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Відкритий	GNU GPL	За гроші	–	–

За результатами таблиці можна зробити висновок, що при проєктуванні серверної частини дистанційної системи консультацій слід у першу чергу звернути увагу на MariaDB, яка має відкритий код, велику швидкість і гнучкість. Альтернативами будуть PostgreSQL та MySQL.

3.3 Проєктування таблиць

Система керування базами даних MariaDB версії 10.6.4 дозволяє швидко та легко створити структуру БД, яку зображено на рисунку 2.4. Створення усіх таблиць бази даних виконується у визначеній послідовності. Таблиці створюються на мові структурованих запитів (SQL), обов'язково враховуючи особливості та відмінності поточної версії СКБД MariaDB 10.6.4. Тексти запитів щодо створення таблиць наведено у В.1 – В.15 (додаток В). Запити на створення таблиць виконуються у визначеній послідовності для виключення колізій зв'язків між таблицями БД.

Головні таблиці:

– Patient;

– PrivatPat;

– VaccCard;

– Predispos;

– Doctor;

– PrivatDoc;

– RecChart;

– History;

– Session;

– Symptom.

Характеристики полів для головних таблиць бази даних наведено у таблицях 3.2 – 3.11.

Таблиця 3.2 – Характеристика полів таблиці «Patient»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
LFN	Текст	64 символи	–	П.І.Б. пацієнта
Photo	Об’єкт	4 байти	–	Фото пацієнта
BirthDate	Дата/час	4 байти	–	Дата народження
Sex	Число	1 байт	–	Стать пацієнта
NormTemp	Число	Коротке ціле 1 байт	–	Нормальна температура тіла
NormSisT	Число	Коротке ціле 1 байт	–	Нормальний артеріальний систолічний тиск
NormDiaT	Число	Коротке ціле 1 байт	–	Нормальний артеріальний діастолічний тиск
UrgentPow	Число	Дійсне одинарної точності	–	Рівень ургентності пацієнта

Таблиця 3.3 – Характеристика полів таблиці «PrivatPat»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
Login	Текст	16 символів	–	Логін пацієнта
HashPassw	Текст	64 символи	–	Ґеш паролю
PasspSerie	Текст	8 символів	–	Серія паспорту
PasspNumb	Текст	16 символів	–	Номер паспорту
PasspProduc	Текст	64 символи	–	Видано паспорт
RegCountry	Текст	32 символи	–	Країна реєстрації
RegRegion	Текст	32 символи	–	Область реєстрації
RegCitySett	Текст	32 символи	–	Місто або село реєстрації
RegStreet	Текст	32 символи	–	Вулиця реєстрації
RegHouse	Текст	4 символи	–	Дім реєстрації
RegCorps	Текст	4 символи	–	Корпус реєстрації
RegApart	Текст	4 символи	–	Квартира реєстрації
Phone1	Текст	16 символів	–	Телефон 1
Phone2	Текст	16 символів	–	Телефон 2
Viber	Текст	16 символів	–	Телефон 3
Telegram	Текст	16 символів	–	Логін у ТЕ
Email	Текст	32 символи	–	Електронна пошта

Таблиця 3.4 – Характеристика полів таблиці «VaccCard»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
IdPatient	Число	Довге ціле 4 байти	Зовнішній ключ	Код пацієнта
VaccDate	Дата/час	4 байти	–	Дата вакцинації
VaccDescr	Текст	64 символи	–	Опис вакцинації

Таблиця 3.5 – Характеристика полів таблиці «Predispos»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
IdPatient	Число	Довге ціле 4 байти	Зовнішній ключ	Код пацієнта
Predis	Число	Довге ціле 4 байти	Індекс	Код схильності пацієнта

Таблиця 3.6 – Характеристика полів таблиці «Doctor»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
LFN	Текст	64 символи	–	П.І.Б. лікаря
Photo	Об’єкт	4 байти	–	Фото лікаря
RecRegion	Текст	32 символи	–	Область пункту прийому
RecCitySett	Текст	32 символи	–	Місто або село пункту прийому
RecDepart	Текст	32 символи	–	Пункт прийому
RecStreet	Текст	32 символи	–	Вулиця пункту
RecHouse	Текст	4 символи	–	Дім пункту
RecCorps	Текст	4 символи	–	Корпус пункту
RecCab	Текст	4 символи	–	Кабінет пункту

Таблиця 3.7 – Характеристика полів таблиці «PrivatDoc»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
Login	Текст	16 символів	–	Логін лікаря
HashPassw	Текст	64 символи	–	Ґеш паролю
Phone1	Текст	16 символів	–	Телефон 1
Phone2	Текст	16 символів	–	Телефон 2
Viber	Текст	16 символів	–	Телефон 3
Telegram	Текст	16 символів	–	Логін у ТЕ
Email	Текст	32 символи	–	Електронна пошта

Таблиця 3.8 – Характеристика полів таблиці «RecChart»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
IdDoctor	Число	Довге ціле 4 байти	Зовнішній ключ	Код лікаря
RecDate	Дата/час	4 байти	–	Дата прийому
BegTime	Дата/час	4 байти	–	Час початку прийому
EndTime	Дата/час	4 байти	–	Час закінчення прийому
Cabinet	Текст	4 символи	–	Кабінет пункту прийому

Таблиця 3.9 – Характеристика полів таблиці «History»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
IdPatient	Число	Довге ціле 4 байти	Зовнішній ключ	Код пацієнта
IdDoctor	Число	Довге ціле 4 байти	Зовнішній ключ	Код лікаря
Numb	Число	Довге ціле 4 байти	–	Номер історії хвороби пацієнта
StartTime	Дата/час	4 байти	–	Дата відкриття історії хвороби
HospCard	Текст	16 символів	–	Номер лікарняного
BegHCard	Дата/час	4 байти	–	Час початку лікарняного
EndHCard	Дата/час	4 байти	–	Час закінчення лікарняного
FinalDiagn	Текст	64 символи	–	Заключний діагноз
OpenFlag	Логічне	1 байт	–	Прапорець відкриття історії хвороби

Таблиця 3.10 – Характеристика полів таблиці «Session»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
IdHistory	Число	Довге ціле 4 байти	Зовнішній ключ	Код історії хвороби
CurrTime	Дата/час	4 байти	–	Дата звернення пацієнта
CurrTemp	Число	Коротке ціле 1 байт	–	Поточна температура тіла пацієнта
CurrSisT	Число	Коротке ціле 1 байт	–	Поточний артеріальний систолічний тиск
CurrDiaT	Число	Коротке ціле 1 байт	–	Поточний артеріальний діастолічний тиск
ApdxText	Текст	255 символів	–	Додатковий коментар від пацієнта
DocRecom	Текст	255 символів	–	Рекомендації лікаря

Таблиця 3.11 – Характеристика полів таблиці «Symptom»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
IdSession	Число	Довге ціле 4 байти	Зовнішній ключ	Код сеансу
Symp	Число	Довге ціле 4 байти	Індекс	Симптоми пацієнта

Тексти запитів щодо поповнення головних таблиць наведено у В.16 – В.27 (додаток В).

Також у базі даних реалізовано довідкові таблиці:

– _Predis;

– _Symp.

Характеристики полів для довідкових таблиць бази даних наведено у таблиці 3.12 та таблиці 3.13.

Таблиця 3.12 – Характеристика полів таблиці «_Predis»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле 4 байти	Первинний ключ	Ключове поле
Name	Текст	64 символи	–	Назва схильності
Score	Число	Дійсне одинарної точності	–	Коефіцієнт впливу

Таблиця 3.13 – Характеристика полів таблиці «_Symp»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
Id	Число	Довге ціле	Первинний ключ	Ключове поле
Name	Текст	64 символи	–	Назва симптому
Score	Число	Дійсне одинарної точності	–	Коефіцієнт впливу

3.4 Заповнення довідкових таблиць

Вміст довідкових таблиць відповідно технічному завданню наведено у таблиці 3.14 та таблиці 3.15.

Таблиця 3.14 – Вміст записів довідкової таблиці «_Predis»

Id	Name	Score
1	Алергічні захворювання	6.0
2	Неврологічні хвороби (у т.ч. інсульт)	4.0
3	Гіпертонічні хвороби	4.0
4	Легеневі захворювання	4.0
5	Захворювання кровоносної системи	3.0
6	Серцево-судинні хвороби	3.0
7	Онкологічні захворювання	2.0
8	Хвороби печінки	1.0
9	Хвороби нирок	1.0

Таблиця 3.15 – Вміст записів довідкової таблиці «_Symp»

Id	Name	Score
1	Підвищення температури тіла	2.5
2	Сухий кашель	2.5
3	Затруднення дихання	2.5
4	Очна біль, почервоніння або подразнення	2.5
5	Втрата координації руху та розрахунку чисел	2.5
6	Нежить	2.5
7	Різке змінення артеріального тиску	2.0
8	Інші больові відчуття (найчастіше у шлунку)	2.0
9	Діарея	2.0
10	Головний біль	2.0
11	Втрата нюхових або смакових відчуттів	1.5
12	Спонтанний висип на різних місцях шкіри	1.5
13	Зміна кольору шкіри (блідість обличчя)	1.5
14	Зниження температури тіла	1.0
15	Біль у горлі або в ухах	1.0
16	Періодичний біль у серці	1.0
17	Втомленість	1.0

Тексти запитів щодо поповнення довідкових таблиць наведено у В.28 – В.29 (додаток В).

3.5 Опис процесу функціонування модуля аналітики

Модуль аналітики призначено для обчислення рівнів ургентності пацієнтів. Процес функціонування модуля аналітики – це сортування списку хворих пацієнтів (пацієнти, для яких почато історію хвороби, але ще не завершено) перед дією оновлення форми інтерфейсу лікаря.

Сортування списку пацієнтів виконується в напрямку за зменшенням значення поля UrgentPow таблиці Patient. Але насамперед це значення розраховується за наступним алгоритмом:

– для поточного лікаря обираються (фільтруються) тільки ті пацієнти, щодо яких у пов’язаній між ними таблиці History поле OpenFlag має значення True. Це показує, що пацієнт вже створив нове звернення до лікаря (заповнив поточні показники біометрії та завершив вибір симптомів щодо поточного захворювання);

– значення поля UrgentPow у таблиці Patient буде приймати значення коефіцієнту K_U, який розраховується за формулою 2.1;

– у формулі 2.2 використовується значення повних років життя пацієнта або інакше – його вік y, який обчислюється за формулою (3.1):

$$y = \text{YEAR}([\text{Session}].[\text{CurrTime}] - [\text{Patient}].[\text{BirthDate}]); \quad (3.1)$$

– значення перевищення поточної температури тіла над потенційно нормальною dT для вибору коефіцієнту впливу по таблиці 2.1 розраховується за формулою (3.2):

$$dT = [\text{Session}].[\text{CurrTemp}] - [\text{Patient}].[\text{NormTemp}]); \quad (3.2)$$

– значення перевищень параметрів артеріального тиску систолічного dS та діастолічного dD над відповідними потенційно нормальними значеннями для вибору коефіцієнтів впливу по таблиці 2.2 розраховується за наступними формулами (3.3), (3.4):

$$dS = [\text{Session}].[\text{CurrSisT}] - [\text{Patient}].[\text{NormSisT}]), \quad (3.3)$$

$$dD = [\text{Session}].[\text{CurrDiaT}] - [\text{Patient}].[\text{NormDiaT}]); \quad (3.4)$$

– коефіцієнт схильності пацієнта до відповідної хвороби C_{Xj} як величина впливу для зазначеного індексу *j* обирається з таблиці Predispos, якщо він там існує у відповідному запису *j*=Id довідкової таблиці _Predis (табл. 3.14);

– коефіцієнт групи симптомів рецидивної форми хвороби S_{Ak} для даного пацієнта як величина впливу для зазначеного індексу *k* обирається з таблиці Symptom, якщо він там існує у відповідному запису *k*=Id для діапазону значень групи Id від 7 до 10 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт посилення групи рецидивних симптомів K_A спочатку дорівнює 1.0, але стане дорівнювати 1.7 тільки якщо будуть у наявності усі симптоми з кодами Id від 7 до 9 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт групи симптомів первинних ознак тяжкої форми S_{Bl} для даного пацієнта як величина впливу для зазначеного індексу *l* обирається з таблиці Symptom, якщо він там існує у відповідному запису *l*=Id для діапазону значень групи Id від 1 до 6 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт посилення групи первинних ознак тяжкої форми K_B спочатку дорівнює 1.0, але стане дорівнювати 2.2 тільки якщо будуть у наявності усі симптоми з кодами Id від 1 до 3 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт групи симптомів вторинних ознак тяжкої форми S_{Cm} для даного пацієнта як величина впливу для зазначеного індексу *m* обирається з таблиці Symptom, якщо він там існує у відповідному запису *m*=Id для діапазону значень групи Id від 11 до 13 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт посилення групи вторинних ознак тяжкої форми K_C спочатку дорівнює 1.0, але стане дорівнювати 1.4 тільки якщо будуть у наявності усі симптоми з кодами Id від 11 до 13 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт групи симптомів легкої форми хвороби S_{Dn} для даного пацієнта як величина впливу для зазначеного індексу *n* обирається з таблиці Symptom, якщо він там існує у відповідному запису *n*=Id для діапазону значень групи Id від 14 до 17 довідкової таблиці _Symp (табл. 3.15);

– коефіцієнт посилення групи симптомів легкої форми K_D спочатку дорівнює 1.0, але стане дорівнювати 1.2 тільки якщо будуть у наявності усі симптоми з кодами Id від 14 до 16 довідкової таблиці _Symp (табл. 3.15).

3.6 Опис скриптів обчислення рівня ургентності пацієнта

Розрахунок ургентності передбачає збір даних з багатьох таблиць та обчислення цих значень за формулою 2.1. Для цього необхідно звернутися до таблиць Patient, History, Session, Predispos, _Predis, Symptoms, _Symp.

Усі дані та обчислення можливо виконати за допомогою SQL-запитів до бази даних, однак реалізація у вигляді підпрограми в БД не є надійною, так як вона унеможливлює зміни частин функцій для користувачів та подальшого масштабування розрахунку цього рівня до більшого значення критеріїв. Як було вже зазначено, розрахунок ургентності залежить від груп симптомів. У даній реалізації рівня ургентності використовується 4 групи та загалом 17 симптомів. Саме тому найкращим було створити основу функції мовою програмування PHP, яка в свою чергу буде відправляти SQL-запити до бази даних [28].

Через те, що існують складні умови в обчисленні коефіцієнтів, було вирішено логіку розрахунку кожного з коефіцієнтів виконувати можливостями вебсерверу. Інакше, бази даних, особливо високо навантажені будуть більше часу витрачати на обчислення, аніж на видачу результатів за запитами типу SELECT.

Основна задача для підготовки розрахунку – це поєднати таблиці з даними. Через те, що зв’язки між таблицями існують це зробити досить просто. Інакше була б необхідність у створенні занадто громіздких таблиць.

Поєднання різних таблиць до єдиної тимчасової таблиці – це найкраще рішення для виконання даної реалізації. Звичайно, така таблиця створюється лише в оперативній пам’яті та під час запиту, і не зберігається у подальшому на носіях.

Важливою причиною створення функції розрахунку ургентності для виконання її вебсервером – повторні виклики цієї функції. На разі, створена функція виконується автоматично кожного разу коли пацієнти оформлюють нові звернення. Але деякі розрахунки було простіше виконати під час виклику збору даних.

Повний лістинг функції calc_urgency наведено у Г.9 (додаток Г). Функція calc_urgency в якості вхідного аргументу потребує ідентифікатор пацієнта для якого будуть виконуватись обчислення.

При отриманні даних використовувався метод JOIN для повного об’єднання двох та більше таблиць за певною умовою. Для однозначного отримання поточних значень біометричних показників пацієнта необхідно проводити пошук у таблиці Session, але ця таблиця не вказує на існування нового звернення до лікаря, і крім того не дозволяє однозначно ідентифікувати якому пацієнту ці дані належать. Саме ці дані дозволяє отримати таблиця History. Тому послідовність пошуку виглядає так:

– пошук записів у таблиці History за відомим ідентифікатором пацієнта за прапорцем відкритого сеансу;

– пошук записів цього сеансу в таблиці Session, звідки отримується номер сесії, який можна використовувати як ключ щодо пошуку симптомів пацієнта у таблиці Symptom;

– пошук записів нормальних показників біометрії за ідентифікатором пацієнта у таблиці Patient;

– пошук схильностей пацієнта з таблиці Predispos, відповідно з коефіцієнтом впливу з таблиці _Predis.

Перелік симптомів для пацієнта буде виглядати як масив ідентифікаторів, таким чином буде зручно виконати порівняння масиву

симптомів пацієнта та контрольний масив групи для більш чіткої ідентифікації. Це можливо виконати двома шляхами: фільтрація та пошук базою даних, або видача масиву і порівняння за допомогою PHP функцій. Було обрано другий варіант, так як використання функції array_intersect не потребує від розробника чіткого сортування масивів, адже симптоми могли не додаватися підряд у правильному порядку. Більш того, групи симптомів можуть у подальшому зростати у кількості груп та входжень симптомів у ці групи, і накладатися один на одного, що буде ускладнювати рішення сформульоване мовою SQL. При даному вирішенні, вказується масив із попередньо заданою групою симптомів та за допомогою функції array_intersect відбувається пошук значень у масиві користувача. За закінченням отримується масив лише із перетином даних. Подальші порівняння з ключовими полями надають відповідь, яке значення коефіцієнтам K_a, K_b, K_c, K_d потрібно надати.

Для розрахунку коефіцієнтів $S*$ необхідно сумувати значення Scores, які будуть відповідати значенням ідентифікаторів симптомів у групі симптомів. Наразі ці групи поділено та розташовано один за одним. Розрахунок таких сум було виконано базою, і запит, який при цьому використовується стане більш громіздким при подальшому масштабуванні кількості симптомів (а особливо при накладанні однієї групи на іншу).

Для отримання всіх необхідних даних необхідно було надіслати 8 запитів до бази даних, кожен з них у подальшому може бути змінено, або можливе додавання нових запитів для розрахунку нових коефіцієнтів. Після розрахунку всіх коефіцієнтів виконується останній запит до бази даних, який оновлює дані в полі UrgentPow пацієнта у таблиці Patient.

4 ПРОЄКТУВАННЯ ІНТЕРФЕЙСІВ СИСТЕМИ

4.1 Обґрунтування вибору типу інтерфейсів дистанційної системи

Розроблювану дистанційну систему консультацій призначено для віддаленого та децентралізованого моніторингу стану пацієнтів з подальшим їх консультуванням з урахуванням рівня ургентності пацієнтів.

Дистанційна система використовує можливості СКБД MariaDB 10.6.4. Для ефективності функціонування бази даних найкраще створити вебсайт.

Серед переваг вебсайту можна виділити наступні:

– вебсайт та програмне забезпечення для нього встановлюється і налаштовується лише на сервері;

– для будь-якого користувача вебсайту достатньо лише мати можливість виходу в Інтернет та сучасний браузер. Жодних встановлень і налаштувань зі сторони клієнтів не потрібно;

– незалежність від операційної системи та пристрою, з якого будуть заходити на сайт. Таке рішення спрощує впровадження системи;

– віддаленість надає можливість лікарю працювати без певного робочого місця. Лікар може надавати рекомендації пацієнтам з будь-якого комп’ютера всесвітньої мережі;

– розроблювана система має два інтерфейси, один для користувачів-пацієнтів, а другий для операторів-лікарів, кожен з яких має увійти до системи (за наявності для нього авторизаційного запису) або створити (зареєструвати) новий.

4.2 Використані технології проєктування

У даній розробці при проєктуванні та створенні вебсайту були використані найбільш популярні, а також стабільні технології сайтобудування:

– HTML;

– CSS;

– JavaScript та бібліотека jQuery (для виконання функцій у фоновому режимі та на клієнтській стороні);

– PHP.

Додаткова бібліотека jQuery дозволила виконувати асинхронні запити та запити у фоновому режимі.

Використання оригінальних мов з меншою кількістю додаткових бібліотек дозволяє в подальшому масштабувати та розвивати проєкт. Крім того, використання більшою часткою оригінального коду дозволяє легше його трансформувати до інших мов або фреймворків.

4.3 Структура вебсайту дистанційної системи

Завершену реалізацію дистанційної системи у ієрархічному та файловому вигляді зображено на рис. 4.1 та рис. 4.2 відповідно.

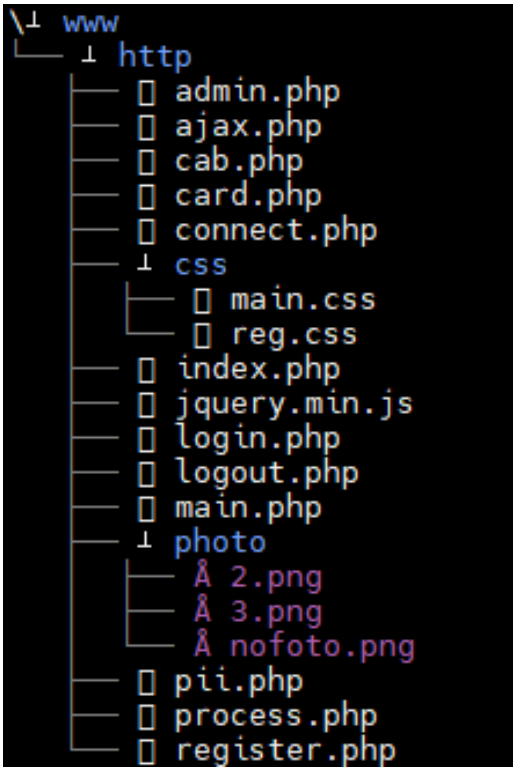


Рисунок 4.1 – Ієрархічна структура вебсайту

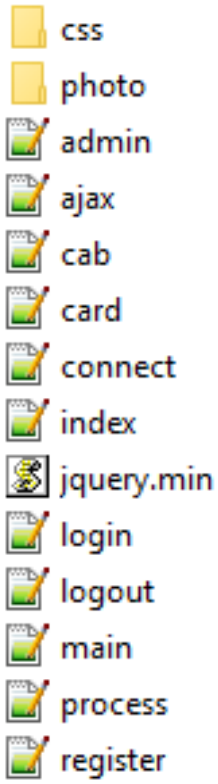


Рисунок 4.2 – Файлова структура вебсайту

Файли з початковим кодом до вебсайту наведено у додатку Г.

Вигляд переліку всіх таблиць бази даних зображено на рисунку 4.3.

```
MariaDB [web]> show tables;
+---------------+
| Tables_in_web |
+---------------+
| ActiveDoc     |
| Admins        |
| Doctor        |
| History       |
| Patient       |
| Predispos     |
| PrivatDoc     |
| PrivatPat     |
| RecChart      |
| Session       |
| SimDoc        |
| Symptom       |
| VaccCard      |
| _Predis       |
| _Symp         |
+---------------+
15 rows in set (0.002 sec)
```

Рисунок 4.3 – Перелік таблиць бази даних у терміналі

У директоріях css та photo (рис. 4.1) містяться відповідні файли. Файл main.css – це головний файл стилю всього сайту. Зміни в ньому впливають на відображення інтерфейсів. Файл reg.css – це файл стилів для сторінки реєстрації register.php. Впливає лише на одну сторінку.

Короткий огляд призначення сторінок вебсайту:

– connect.php – файл для з’єднання вебсайту із базою даних;

– login.php – файл сторінки входу до системи;

– logout.php – файл-функція, що від’єднує користувачів від системи;

– index.php – головна сторінка, вміщує в собі панель навігації, має посилання до інших сторінок, виконує роль контролера;

– register.php – сторінка для реєстрації різних типів користувачів (виключення – адміністраторський тип запису, який треба додавати до бази даних);

– admin.php – файл, що презентує просту сторінку для адміністратора щодо видалення акаунтів користувачів з системи та функцією активування/заблокування акаунтів для лікарів;

– main.php – головна сторінка системи, усі користувачі можуть переглянути після входу до системи;

– process.php – файл, що містить більшість функцій, розроблених для системи (функція розрахунку ургентності міститься саме тут);

– ajax.php – файл, що призначений для зберігання та виконання функцій асинхронно, за запитом користувача та в фоновому режимі (наприклад, автоматичний підрахунок UrgentPow або оновлення частин сторінок без повного їх перезавантаження);

– jquery.min.js – бібліотека JavaScript, яка надає можливість фонового та асинхронного виконання деяких функцій;

– cab.php – сторінка кабінету лікаря;

– card.php – сторінка кабінету пацієнта.

Директорія photo призначена для зберігання фото щодо акаунтів лікарів та пацієнтів.

Головна сторінка сайту являє собою поєднання файлів index.php (додаток Г.1) та main.php (додаток Г.2). Це було реалізовано використанням функції include та запитом суперглобальної змінної $_GET [30].

Функція додавання однієї сторінки до іншої має вигляд <?php include basename($page).`.php`;?> та додає вміст сторінки до тіла index.php. Перелік файлів реалізовано посиланнями за шаблоном ….. Синтаксис «?page=» вказує браузеру яке саме значення треба передати через змінну $_GET. Так як index.php – перша сторінка, на яку за замовчанням потрапляє користувач вебсайту, то всі необхідні для подальшої роботи змінні та сторінки додаються саме до цього файлу (рис. 4.4), де окрім всього іншого існує перенаправлення до сторінки входу, якщо користувача не було під'єднано до системи.

```php
<?php
ob_start();
session_start();
error_reporting(1);
include("connect.php");
include "process.php";
if (!isset($_SESSION['opt'])) {
  header('Location: login.php');}
$page = (isset($_GET['page']) ? $_GET['page'] : 'main');
?>
```

Рисунок 4.4 – Ініціалізація змінних та перевірка входу в index.php

Для спрощення роботи дистанційної системи і полегшення розробки проєкту саме до цієї сторінки додані сторінки-бібліотеки та файл стилів. Усі ці сторінки будуть впливати на всі інші сторінки, які можуть навіть не мати службових міток, крім необхідних функцій. Мається на увазі синтаксис HTML та його мітки типу <html>, <head>, <body>, <footer> та інші. Усі сторінки по суті будуть додаватися до вмісту <body>…</body> одразу після </header> (рис. 4.5).

```php
<?php
ob_start();
session_start();
error_reporting(1);
include("connect.php");
include "process.php";
if (!isset($_SESSION['opt'])) {
  header('Location: login.php');}
$page = (isset($_GET['page']) ? $_GET['page'] : 'main');
?>
<html>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="css/main.css" rel="stylesheet" />
<script type="text/javascript" src="jquery.min.js"></script>
```

Рисунок 4.5 – Додані службові сторінки до файлу index.php

Так як дистанційна система має обслуговувати різних користувачів, з різними необхідними функціями, то адресна панель буде мати різні клавіші, в залежності від того чи пройшов користувач автентифікацію чи ні. Для цього перевіряється глобальна змінна $_SESSION на наявність певних даних. А саме використання змінної $_SESSION[`opt`] буде доступне лише після виконання команди session_start. Ця команда має бути або у кожному файлі вебсайту, або в головній сторінці, до якої додаються усі інші сторінки.

Для підтримки різного рівня доступу та функціоналу були створені сторінка реєстрації register.php (додаток Г.3) та сторінка входу у систему login.php (додаток Г.4). Особливістю даних сторінок є те, що вони створені для універсального використання користувачами. Кожна з цих сторінок має вибір для подальшого користування, а також має необхідність в сторінці з’єднання з базою даних connect.php (додаток Г.5), адже ці сторінки більш самостійні ніж інші, що підпорядковані index.php.

У файлі main.php вказано текст головної сторінки із посиланнями на всі доступні сторінки. Користувачі матимуть доступ лише до деяких сторінок в залежності від їх типу. Так само як і адресна панель, яка буде різною в залежності від користувача (рис. 4.6 – рис. 4.8).

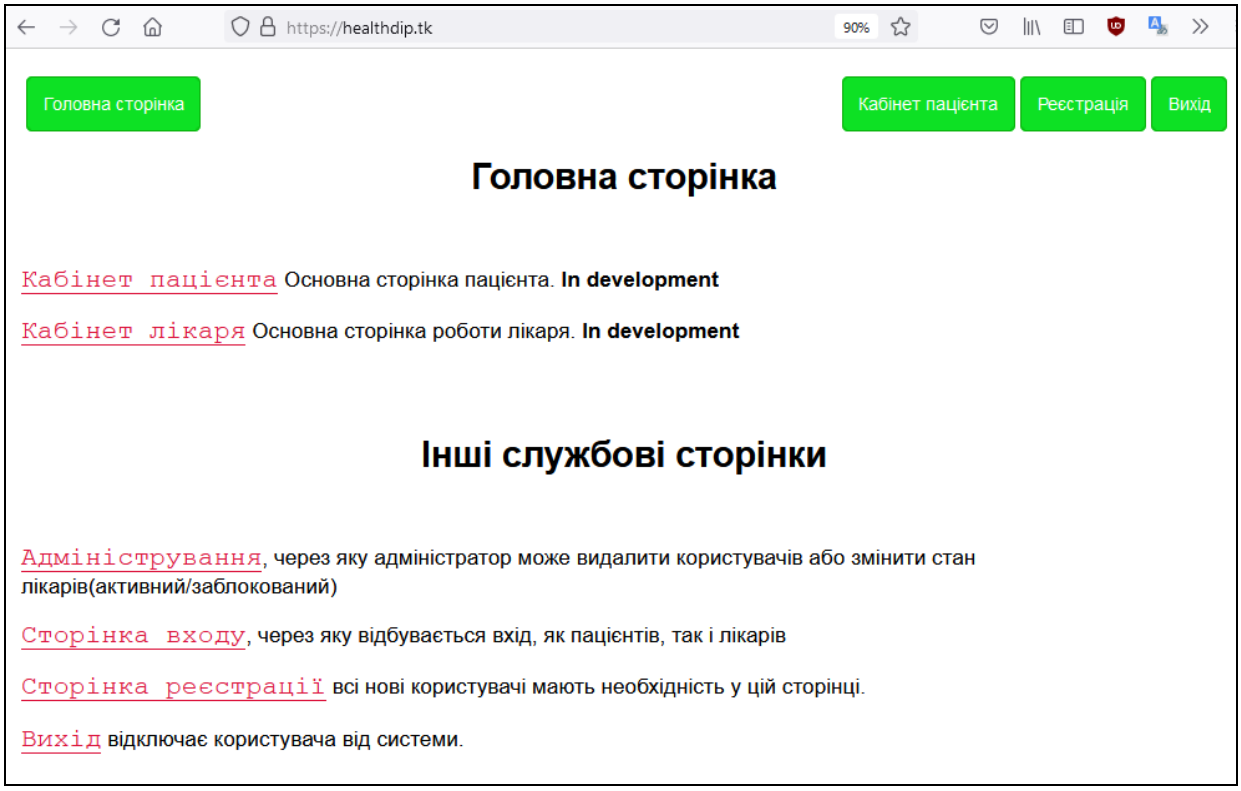


Рисунок 4.6 – Вигляд головної сторінки при вході пацієнта

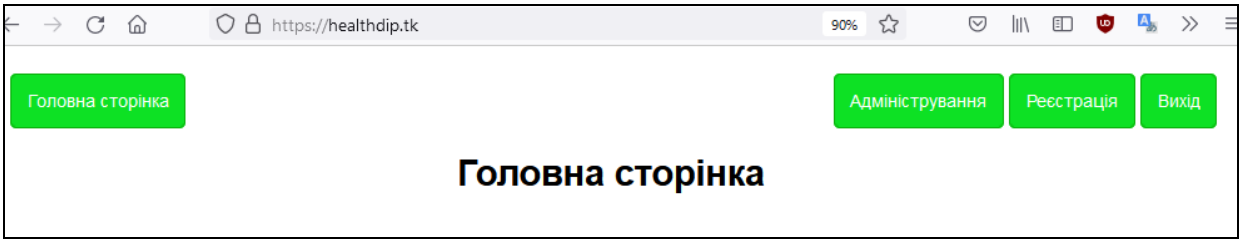


Рисунок 4.7 – Вигляд панелі навігації головної сторінки при вході адміністратора

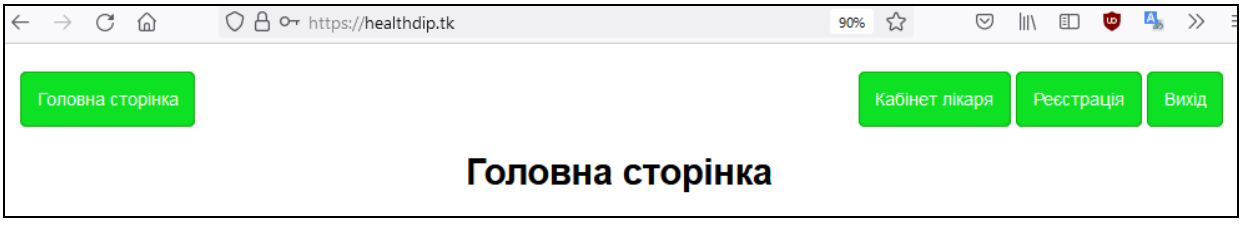


Рисунок 4.8 – Вигляд панелі навігації головної сторінки при вході лікаря

Можливість такої зміни панелей навігації реалізовано у файлі index.php (рис. 4.9).

```
<body>
  <header>
    <div class="navbar">
      <a href="/?page=main">Головна сторінка</a>
      <div class="enterbtn">
        <?php
        if (isset($_SESSION['opt'])) {
          switch ($_SESSION['opt']) {
            case 'adminform':
              echo '<a href="?page=admin">Адміністрування</a>';
              break;
            case 'docform':

              echo '<a href="?page=cab">Кабінет лікаря</a>';
              break;
            case 'patientform':

              echo '<a href="?page=card">Кабінет пацієнта</a>';
              break;
          }
        }


        ?>
        <a href="/register.php">Реєстрація</a>
        <?php
        if (!isset($_SESSION['use'])) {
          echo '<a href="?page=login" >Увійти</a>';
        } else {
          echo ' <a href="?page=logout">Вихід</a>';
        } ?>
      </div>
    </div>
  </header>
  <?php include basename($page) . '.php'; ?>
</body>
```

Рисунок 4.9 – Частина коду файлу index.php, що змінює панель навігації

Усі необхідні дані отримуються у вигляді масивів з БД за допомогою процедурних функцій mysqli_query та mysqli_fetch_assoc.

Сторінки реєстрації та входу до системи мають спільну рису – в них можна обрати для якого типу акаунта буде здійснюватися функція сторінки. Реєструвати можливо пацієнта та лікаря, а виконувати вхід до системи лікарю, пацієнту та адміністратору. Але зареєстрований лікар не зможе одразу увійти до системи, допоки адміністратор не активує його акаунт. Це додатковий рубіж перевірки на валідність акаунту.

Сторінки лікаря (додаток Г.6) та пацієнта (додаток Г.7) розроблені для перегляду архівних даних та створення нових звернень, для перегляду контактної інформації (лікар – пацієнта, пацієнт – сімейного лікаря).

Сторінка «Адміністрування» має на меті лише 2 функції щодо спрощення роботи адміністратора в локальній базі (додаток Г.8)

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Призначення програми

Призначення дистанційної системи консультацій сімейного лікаря − дозволити пацієнтам не виходячи з дому створювати звернення до свого сімейного лікаря, вказуючи поточні параметри температури тіла, артеріального тиску та основних симптомів захворювання, що дозволить лікарю звернути увагу по-перше саме на того пацієнта, який як найбільше потребує медичної допомоги.

У процесі розробки дистанційної системи було створено базу даних, яка дозволить зберігати інформацію щодо звернень пацієнтів, їх деяких біометричних показників, наявні симптоми захворювання, а також відповіді та рекомендації сімейного лікаря. Дистанційна система дозволить оперативно обчислювати рівень ургентності пацієнтів та виконувати сортування звернень до сімейного лікаря в залежності від тяжкості захворювання відповідного пацієнта.

Базу даних створено у СКБД MariaDB версії 10.6.4.

У процесі створення вебінтерфейсів дистанційної системи було застосовано наступні мови: HTML 5, PHP, JavaScript, технологію CSS, бібліотеку jQuery.

5.2 Характеристики програми

Дистанційну систему консультацій сімейного лікаря було розроблено з використанням найсучасніших технологій та призначено для роботи у браузерах будь-яких операційних систем.

Досліджено та реалізовано спосіб обчислення рівня ургентності пацієнта в залежності від його схильностей до захворювань та поточного стану здоров’я. Цей спосіб можна буде запропонувати для використання у сучасних системах телемедицини.

Для захисту від спотворення інформації та шкідливих дій реалізовано технологію розмежування доступу користувачів з використанням ґешування паролю алгоритмом SHA-512 на стороні клієнта.

Дистанційну систему адаптовано для розгортання на сервері будь-якого хостингу з доступом від безлічі клієнтських вузлів, що обмежено тільки можливостями СКБД MariaDB версії 10.6.4.

Розроблювана дистанційна система дозволить виконувати наступні функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації поточної температури тіла та артеріального тиску, а також основних симптомів захворювань;

– зберігання показників схильності пацієнтів для подальшого обчислення рівню ургентності;

– можливість для пацієнтів залишати у зверненні коментарі щодо стану свого здоров’я;

– моніторинг лікарем як відкритих звернень пацієнтів, так і завершених;

– моніторинг пацієнтом як відкритих, так і завершених до лікаря звернень;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні;

– можливість для лікаря проводити пошук у базі звернень за різними категоріями та біометричними даними пацієнтів.

Головною відмінністю розроблюваної дистанційної системи є запропонований спосіб обчислення рівня ургентності пацієнта в залежності від його схильностей до захворювань та поточного стану здоров’я.

5.3 Звертання до програми

5.3.1 Керівництво для пацієнта

Для роботи з дистанційною системою не має потреби у встановленні будь-яких компонентів до складу операційної системи крім стандартного браузеру з останніми оновленнями.

При першому вході до вебсайту користувача буде направлено до сторінки входу (рис. 5.1), на сторінці якого у випадаючому списку він має обрати значення «Пацієнт» та увести власні дані для входу (логін та пароль) (рис. 5.2). Якщо користувач зробив помилку при введені, то він отримає повідомлення (рис. 5.3), але при коректному введені даних користувач опиниться на головній сторінці та побачить головну сторінку, яку зображену на рисунку 4.6.

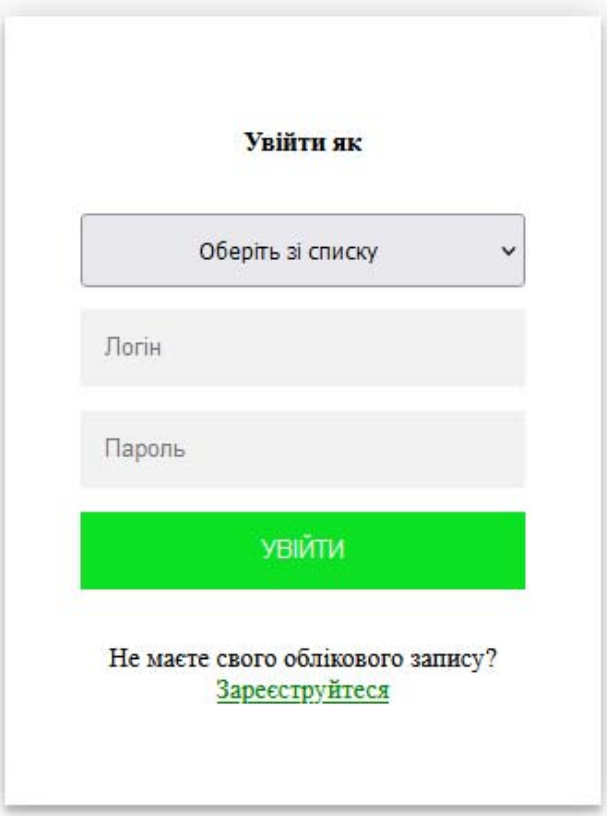


Рисунок 5.1 – Сторінка входу до вебсайту

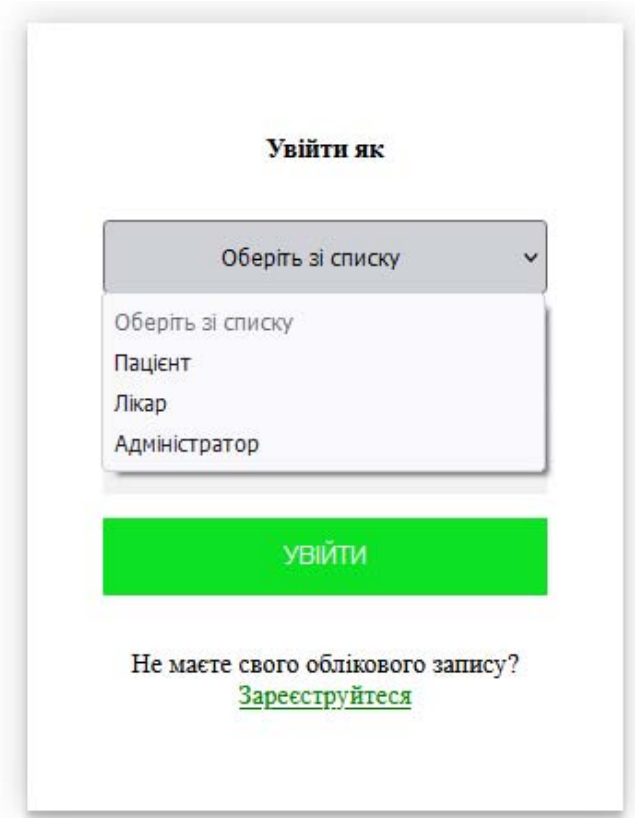


Рисунок 5.2 – Вибір необхідного типу акаунту на сторінці входу

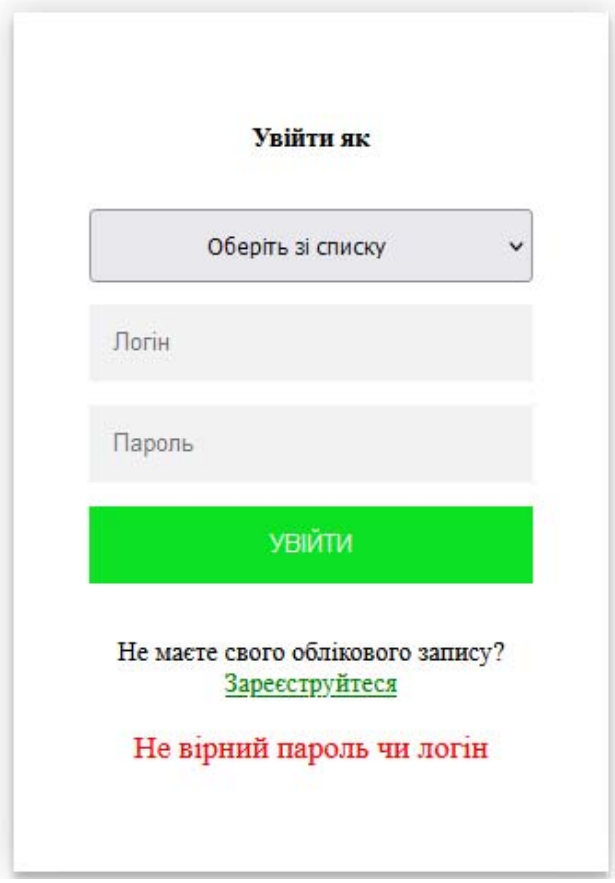


Рисунок 5.3 – Невдалий вхід користувача

Якщо користувача ще не зареєстровано у системі, то йому варто перейти на сторінку реєстрації (рис. 5.4 – рис. 5.6) та заповнити форму. Після цього користувач отримає доступ до системи.

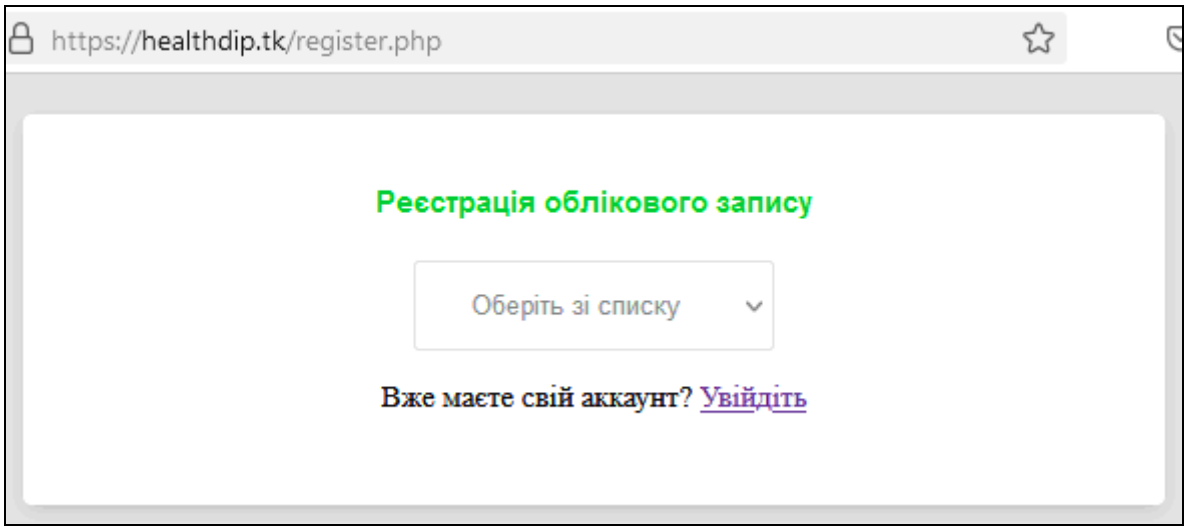


Рисунок 5.4 – Початковий вигляд сторінки реєстрації

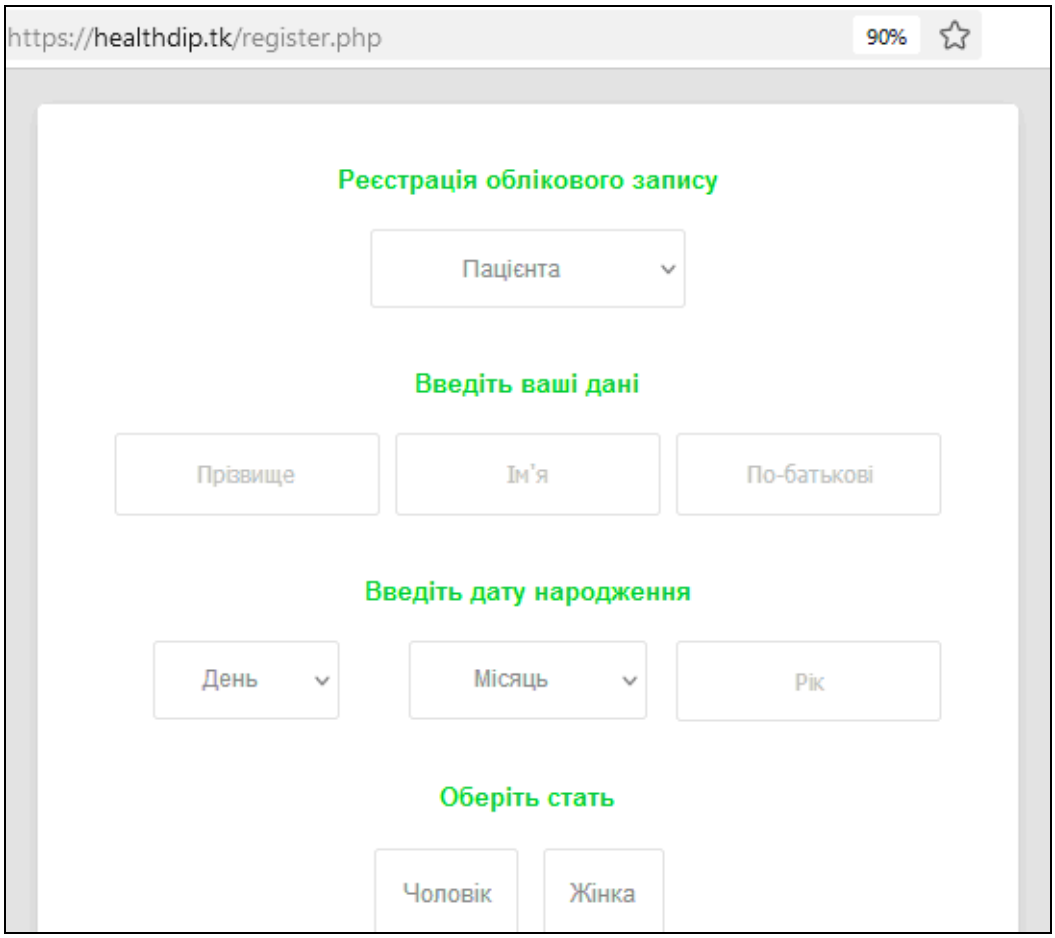


Рисунок 5.5 – Зміна вигляду сторінки реєстрації після вибору типу акаунту

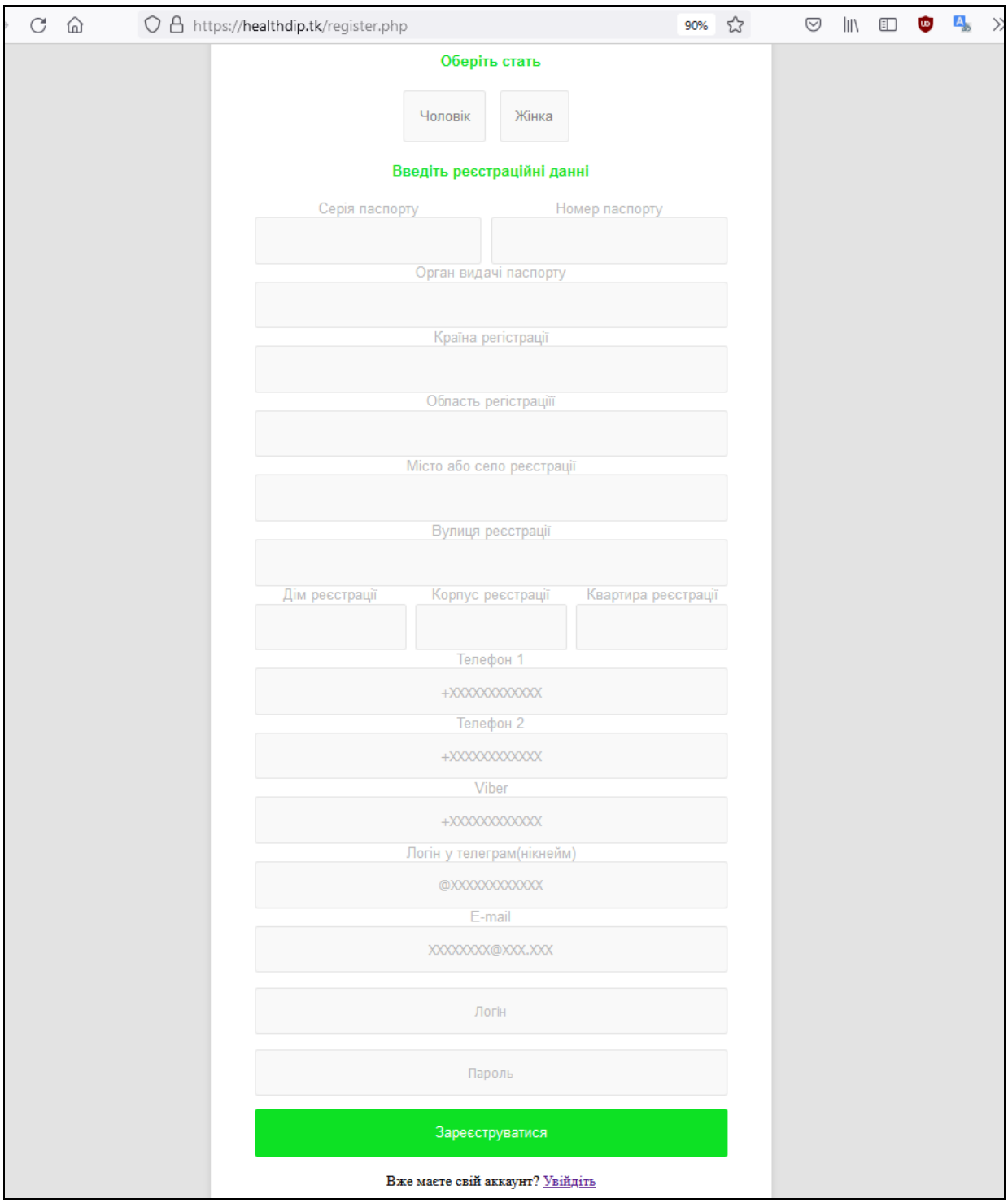


Рисунок 5.6 – Продовження сторінки реєстрації для пацієнта

Одразу після входу користувач зможе перейти на сторінку «Кабінет пацієнта» (рис. 5.7). Ліворуч буде вказано обраного сімейного лікаря, клавіші для розгортання його контактів та місця роботи, праворуч будуть вказані клавіші для створення нового запиту до лікаря, відображено архів звернень, та буде можливість відкрити більш детально звернення.

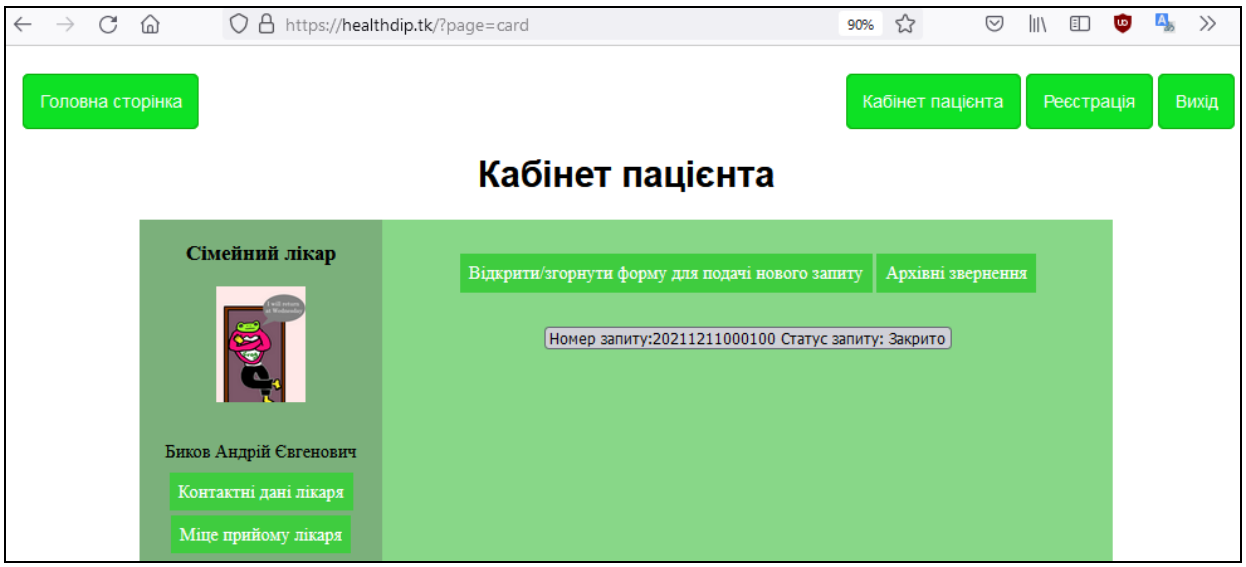


Рисунок 5.7 – Кабінет пацієнта із архівним зверненням

Також можлива ситуація, якщо це перший вхід користувача, то вірогідно, що він ще не обрав лікаря, який буде для нього сімейним і він отримає вигляд сторінки, як на рисунку 5.8. Для вибору лікаря буде достатньо натиснути на його картку і після цього стануть доступними його контактні дані у формі ліворуч.

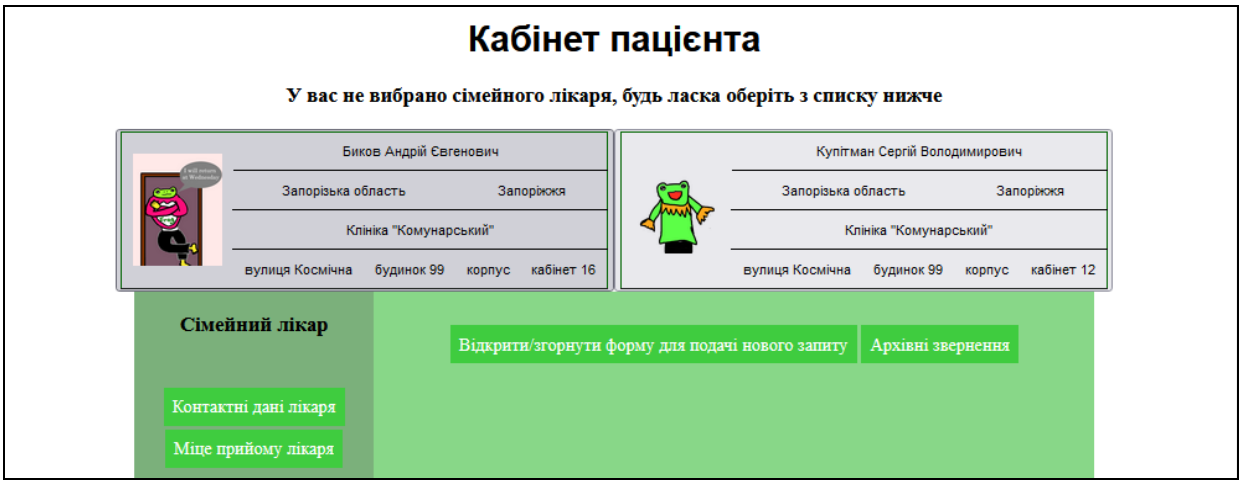


Рисунок 5.8 – Кабінет пацієнта, із можливістю обрання сімейного лікаря

На рисунку 5.9 зображено інтерфейс користувача, у якого обрано сімейний лікар, відкриті його контактні та робочі дані, а також відкрите нещодавно створене звернення.

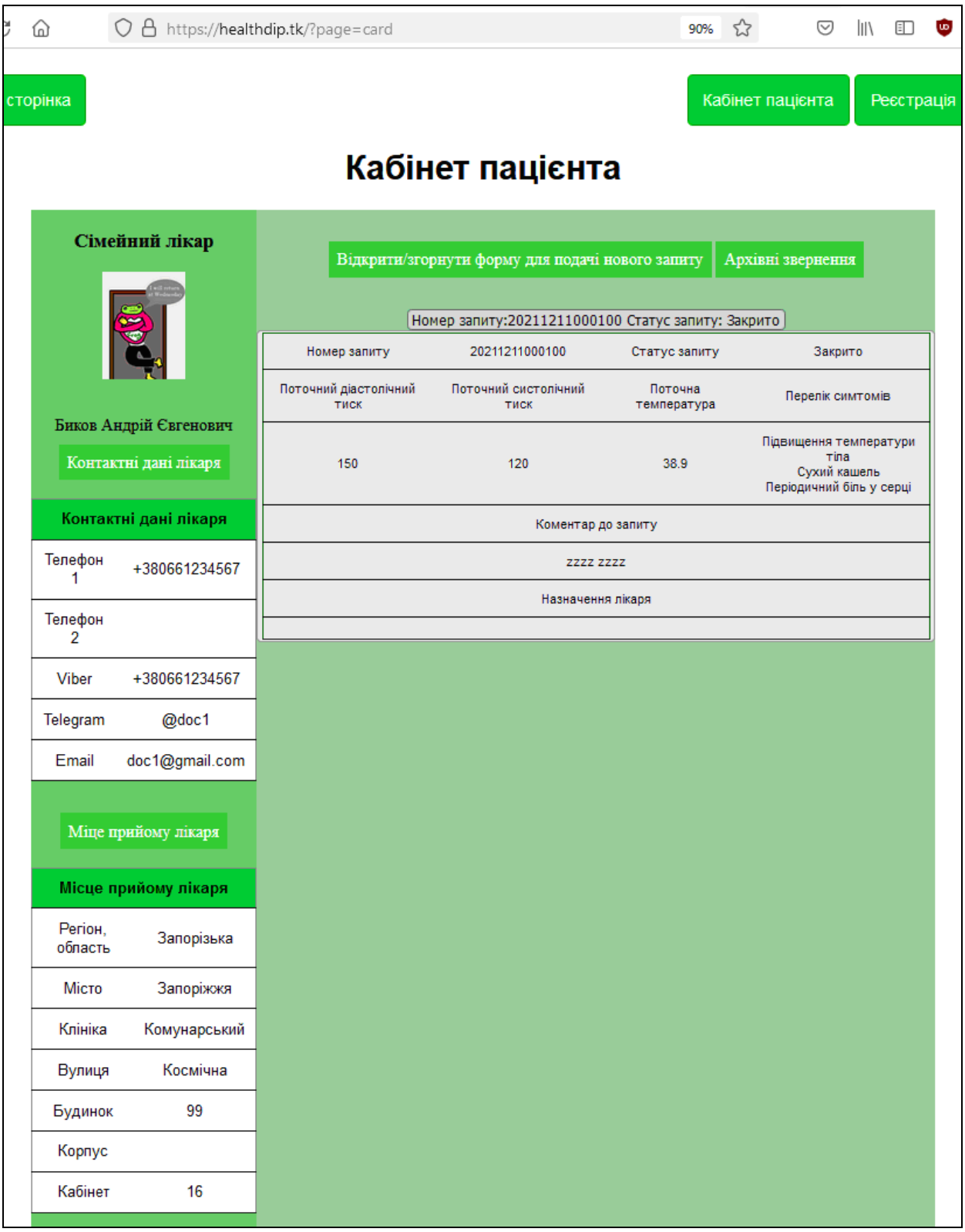


Рисунок 5.9 – Кабінет пацієнта із відкритими формами контактів лікаря та архівним зверненням

На рисунку 5.10 зображена форма нового звернення до лікаря.

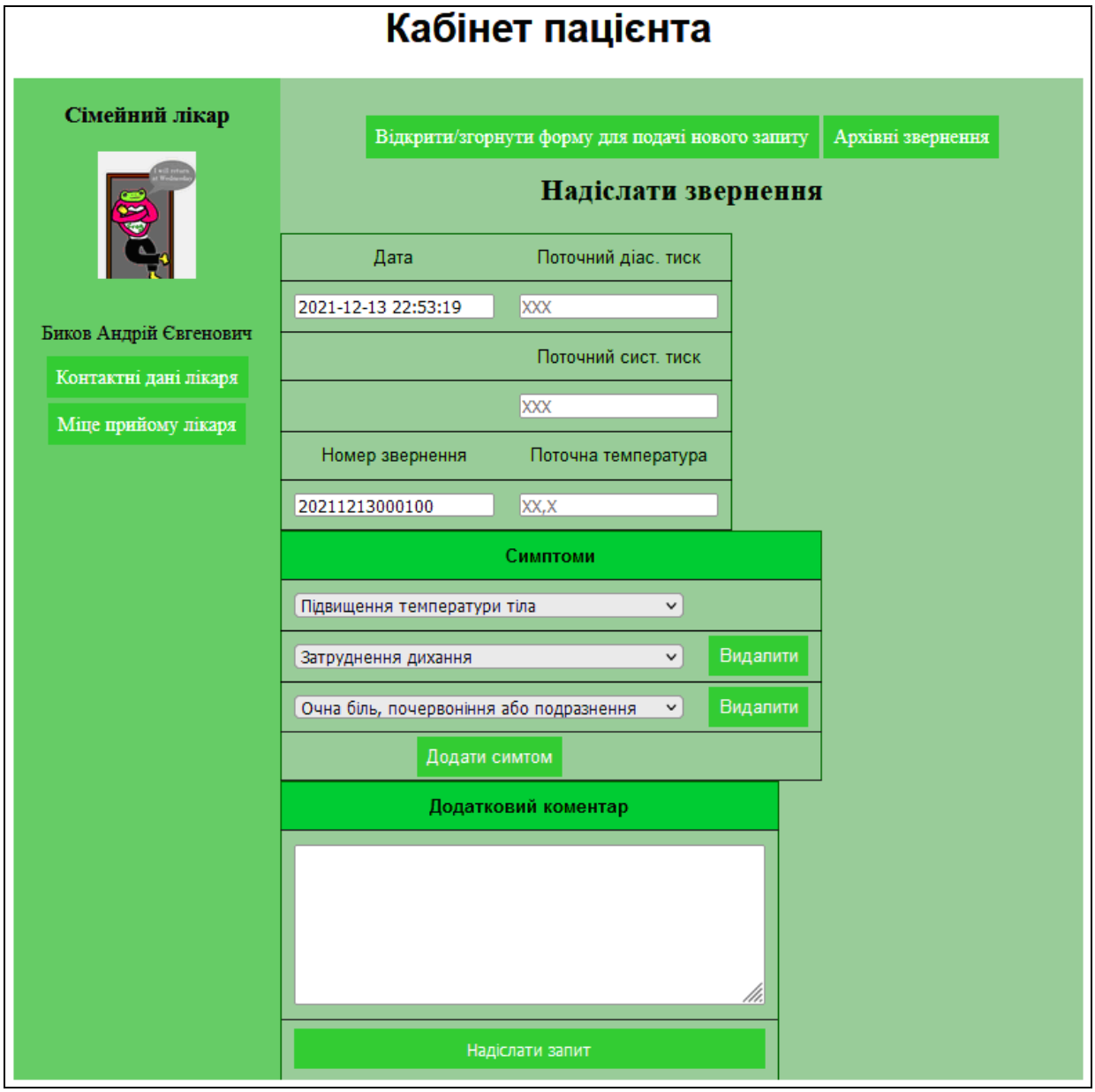


Рисунок 5.10 – Кабінет пацієнта та інтерфейс додавання нового звернення

5.3.2 Керівництво для лікаря

Лікар при вході до системи також зустрінеться із сторінками входу як на рис. 5.1 – рис. 5.2, але крім цього не лише помилкове введення даних не зможе дозволити лікарю увійти до системи, але й статус акаунту. Так на рисунку 5.11 відображена помилка, коли лікар не зміг потрапити до системи через те що його акаунт заблоковано.

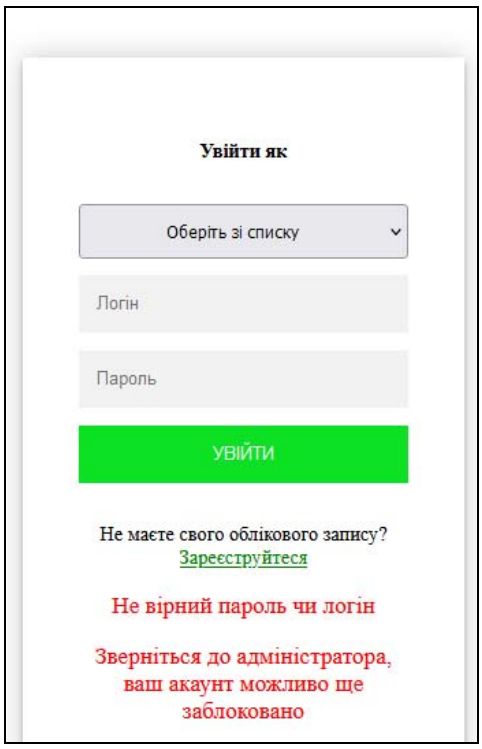


Рисунок 5.11 – Помилка входу акаунта лікаря через заблокований статус

Сторінка «Кабінет лікаря» має вигляд як на рисунку 5.12. Допоки не обрано жодного користувача, ліворуч таблиці знаходиться перелік пацієнтів сортований за рівнем ургентності від найбільшого до найменшого показника.

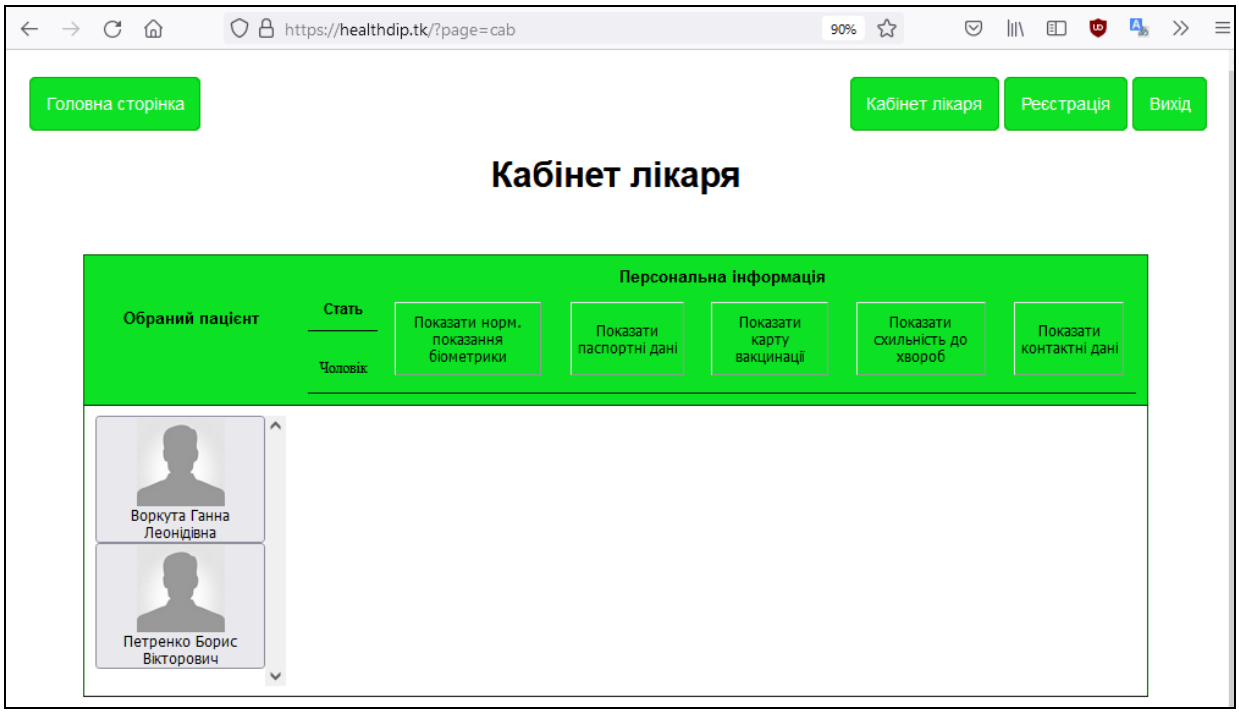


Рисунок 5.12 – Початковий вигляд «Кабінету лікаря»

Після дії обрання пацієнта форма оновлюється (рис. 5.13), фото пацієнта та основні його дані розташовуються у верхньому лівому куті форми, з’являється додаткова інформація.

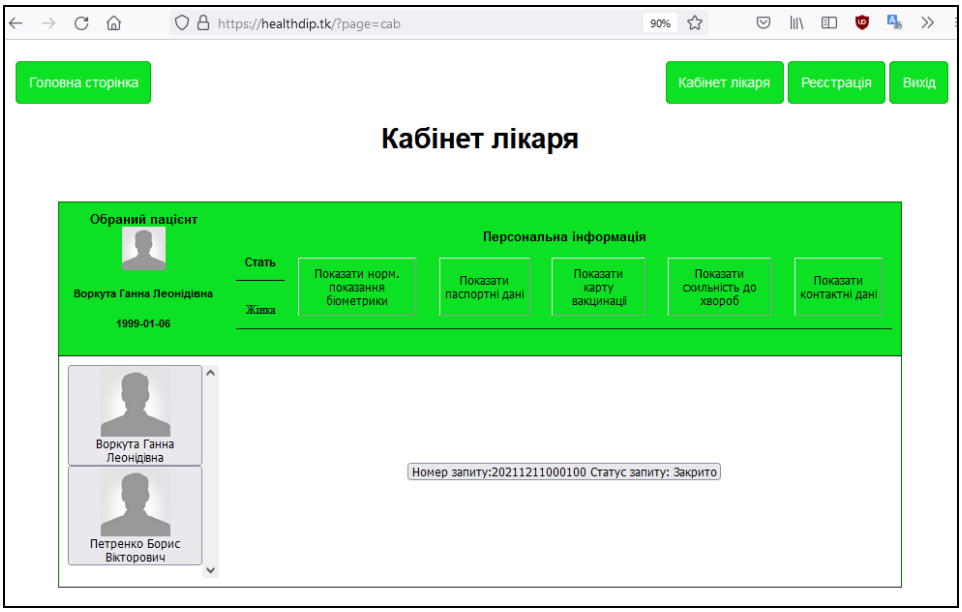


Рисунок 5.13 – На сторінці «Кабінет лікаря» обрано пацієнта для перегляду

Серед додаткової інформації – дата народження пацієнта, стать, оновлюється перелік звернень пацієнта. При натисканні кнопок у верхній стрічці відкриваються форми з додатковою інформацією (рис.5.14 – рис.5.18)

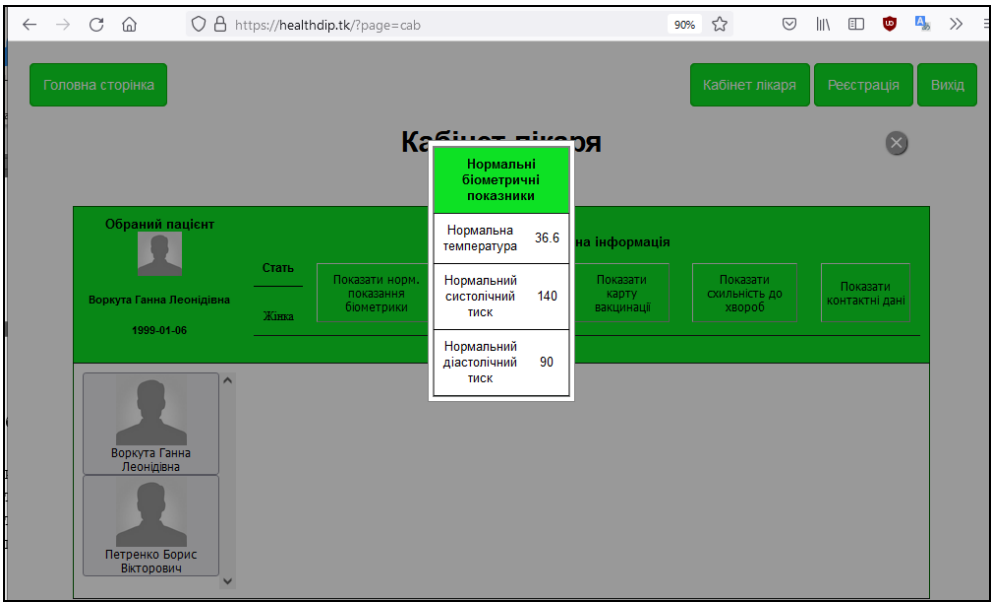


Рисунок 5.14 – Перегляд нормальних біометричних показників пацієнта

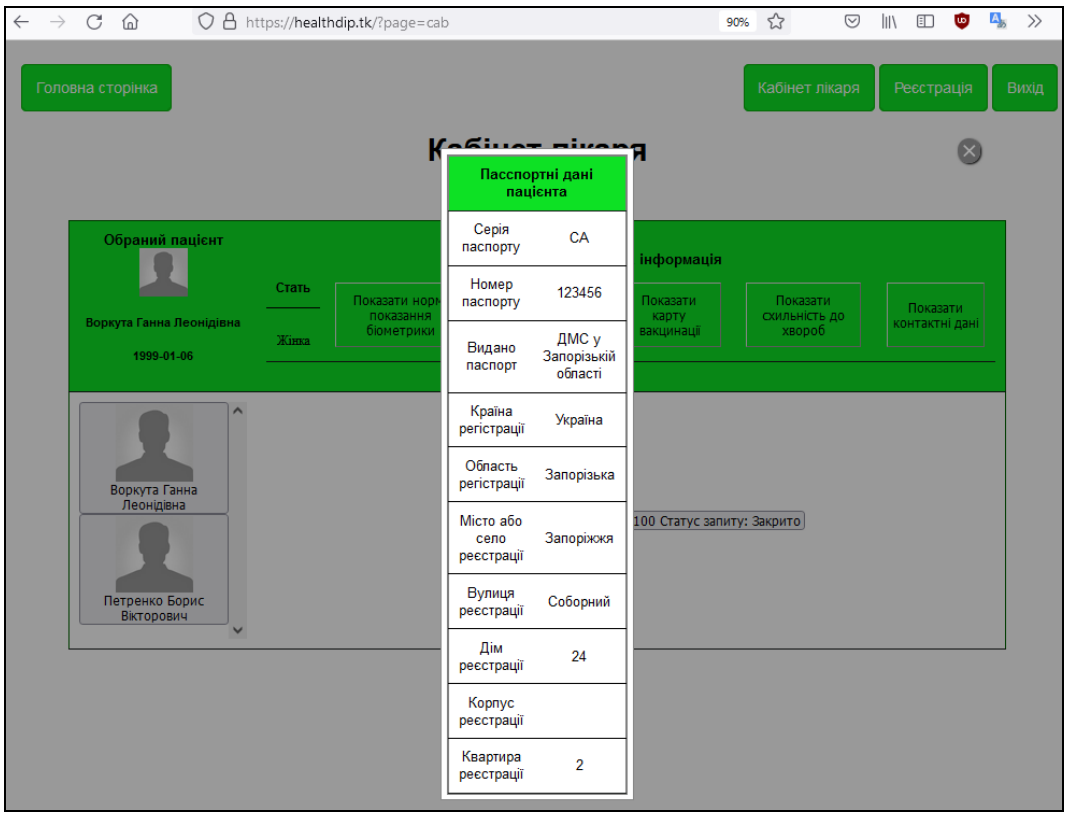


Рисунок 5.15 – Перегляд паспортних (реєстраційних) даних пацієнта

Лікар може також додавати або видаляти записи для користувача щодо щеплень, схильностей до захворюваностей (рис. 5.16 – рис. 5.17). Ці зміни також будуть мати свій вплив у розрахунку ургентності.

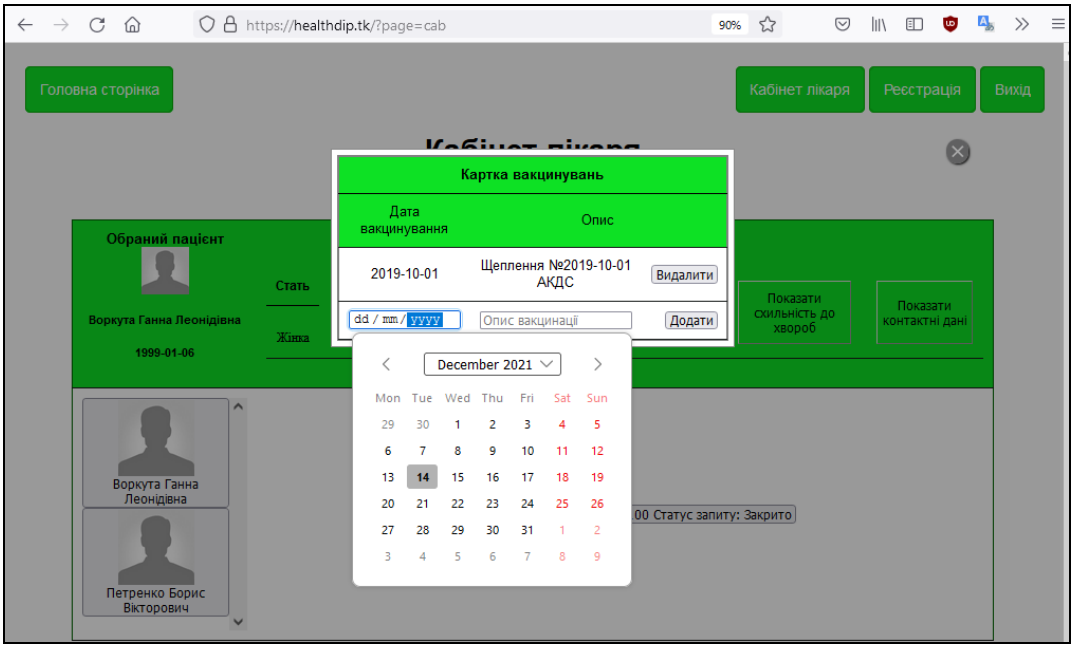


Рисунок 5.16 – Перегляд та додавання щеплень пацієнта

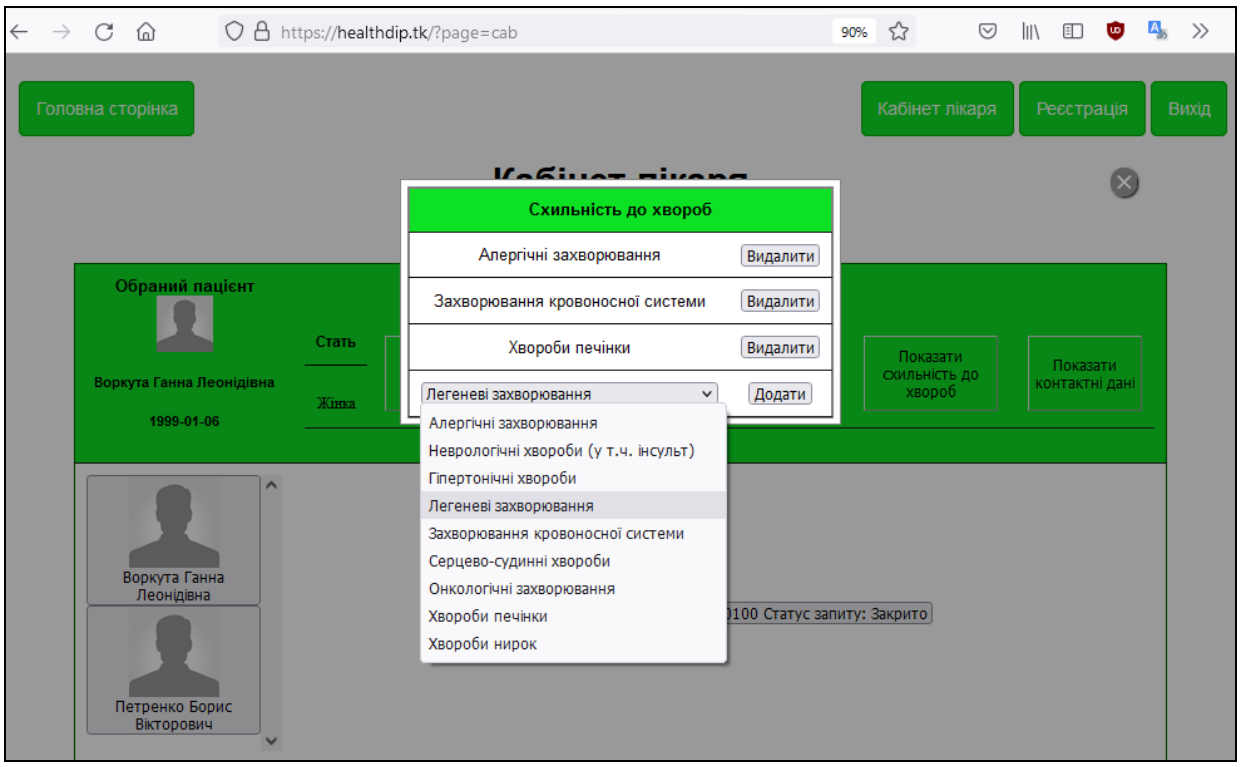


Рисунок 5.17 – Додавання та видалення схильностей пацієнта

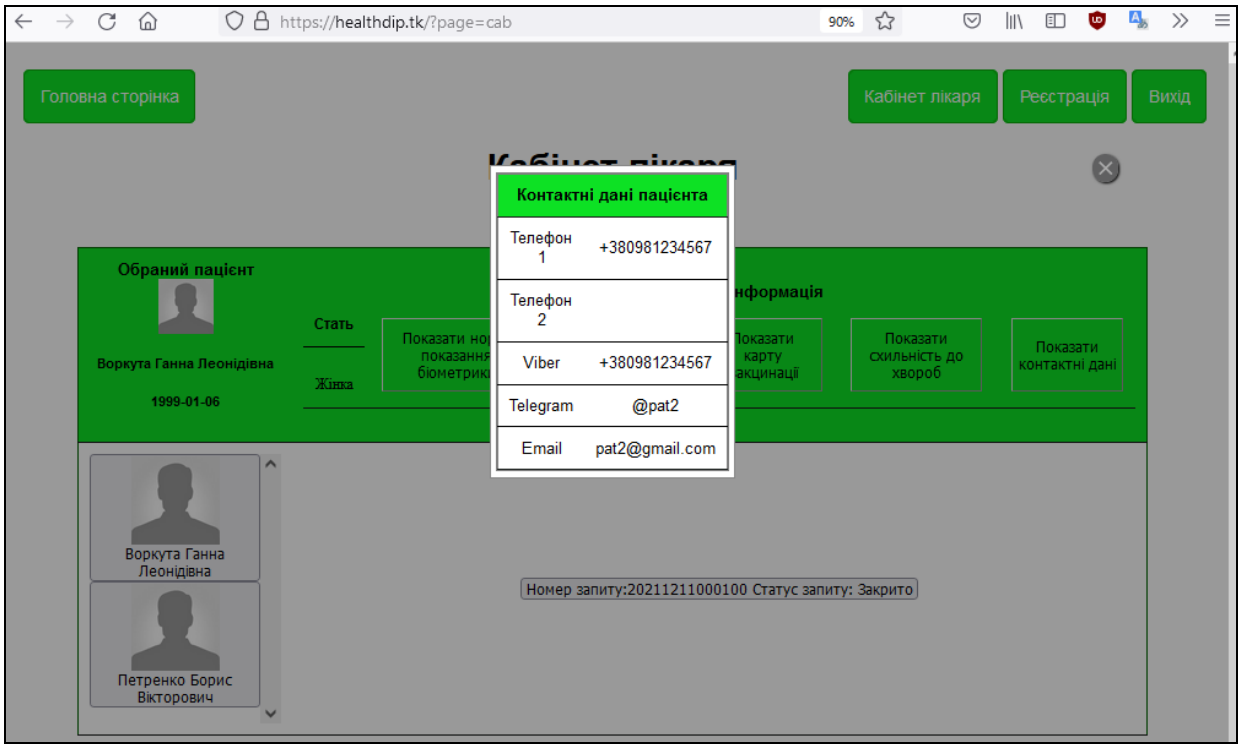


Рисунок 5.18 – Перегляд контактних даних пацієнта

На рисунку 5.19 зображено інтерфейс лікаря у момент відкриття звернень від пацієнта.

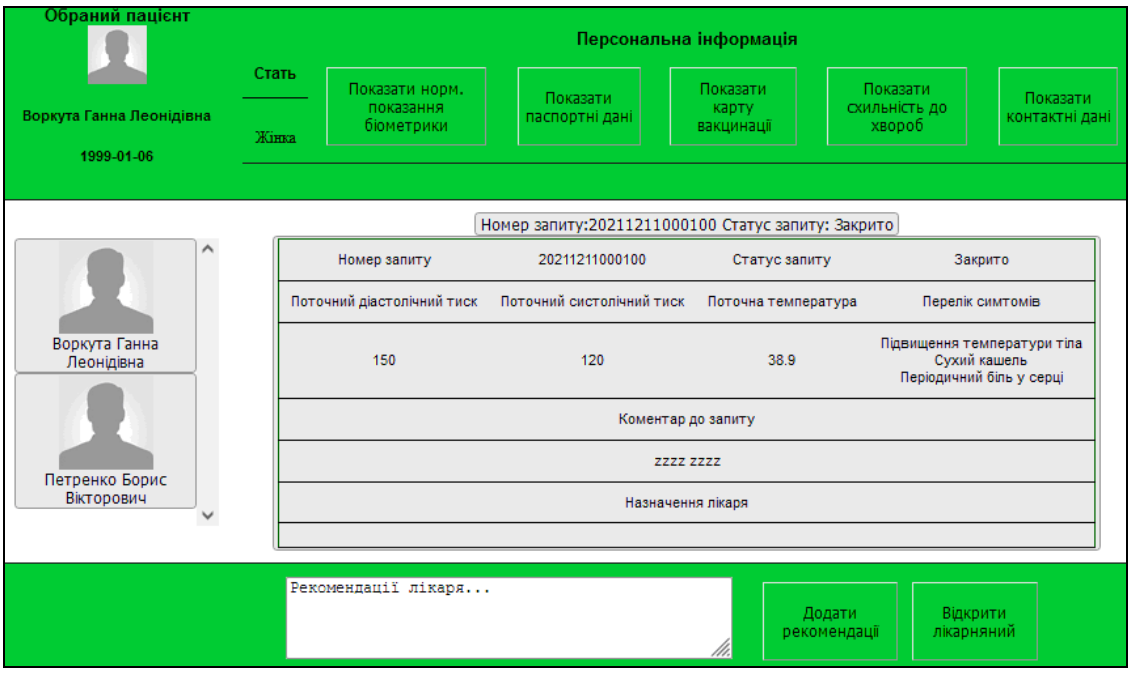


Рисунок 5.19 – Перегляд звернення та керування ним

5.3.3 Керівництво для адміністратора системи

Адміністратор використовує вебсайт для керування акаунтами користувачів. Адміністратор не може бути зареєстрований через сторінку реєстрації як пацієнт чи лікар. Сторінка «Адміністрування» має простий вигляд і дозволяє спростити частину роботи (рис. 5.20).

Рисунок 5.20 – Інтерфейс сторінки «Адміністрування»

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи магістра було проведено аналіз існуючих медичних інформаційних систем, визначено вимоги до проєктування, створено структурну та функціональну схеми дистанційної системи, розроблено спосіб обчислення рівня ургентності пацієнта, створено концептуальну та реляційну модель БД, виконано її нормалізацію та побудовано логічну схему бази даних, обрано середовище проєктування – СКБД MariaDB версії 10.6.4, розроблено фізичну модель бази даних з таблицями, які створюються за допомогою скриптів SQL, для створення вебінтерфейсів лікаря та пацієнта застосовано сучасні технології HTML 5, PHP, JavaScript, CSS та бібліотеку jQuery.

Розроблена дистанційна система дозволить виконувати такі функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації декількох біометричних даних, а також основних симптомів захворювань;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні.

Наукова новизна роботи полягає в запропонованому способі використання біометричних даних, зокрема температури і кров’яного тиску, для прискорення аналізу та класифікації стану пацієнта.

Практична цінність роботи полягає в тому, що досліджено та реалізовано спосіб обчислення рівня ургентності пацієнта в залежності від його схильностей до захворювань та поточного стану здоров’я. Цей спосіб можна буде запропонувати для використання у сучасних системах телемедицини.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Державні послуги онлайн [Електронний ресурс]. – Режим доступу: https://plan2.diia.gov.ua/projects?fbclid=IwAR1qj--MPvWg1SvPecO4P_uFxzGvBk0hxtOwlqNDNFxq17c0pk2kwVGi9M

2. Портал Національної служби здоров'я України [Електронний ресурс]. – Режим доступу: https://nszu.gov.ua/e-data/esoz

3. Медична інформаційна система. Вікіпедія. Вільна енциклопедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Медична_інформаційна_система

4. EMCiMED. Medical information system [Електронний ресурс]. – Режим доступу: https://emci.ua/statti/iak-vybraty-mis/

5. Медичні інформаційні системи: огляд можливостей і приклади використання [Електронний ресурс]. – Режим доступу: https://evergreens.com.ua/ua/articles/medical-information-systems.html

6. Офіційний сайт електронної системи охорони здоров'я eHealth [Електронний ресурс]. – Режим доступу: https://ehealth.gov.ua/pidklyucheni-do-ehealth-mis/

7. Офіційний Сайт Health24 [Електронний ресурс]. – Режим доступу: https://h24.ua/

8. Офіційний Сайт Helsi [Електронний ресурс]. – Режим доступу: https://helsi.me/

9. ЕМСіМЕД. Вікіпедія. Вільна енциклопедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ЕМСіМЕД

10. Офіційний сайт Dr. ELEKS. Medical information system [Електронний ресурс]. – Режим доступу: https://doctor.eleks.com/

11. Поліклініка без черг. Вікіпедія. Вільна енциклопедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Поліклініка_без_черг

12. Офіційний сайт MC Plus. Медична інформаційна система (uk) [Електронний ресурс]. – Режим доступу: https://mcplus.com.ua/

13. Підключені до eHealth MIC. eZdorovya (uk) [Електронний ресурс]. – Режим доступу: https://ehealth.gov.ua/pidklyucheni-do-ehealth-mis/

14. Офіційний Сайт Askep [Електронний ресурс]. – Режим доступу: https://askep.net/

15. Офіційний Сайт SimplexMed [Електронний ресурс]. – Режим доступу: http://simplex-med.com/

16. Офіційний Сайт MedElement [Електронний ресурс]. – Режим доступу: https://medelement.com/en

17. Офіційний Сайт МедІнфоСервіс [Електронний ресурс]. – Режим доступу: https://www.infomed.ck.ua/

18. Офіційний Сайт IMED [Електронний ресурс]. – Режим доступу: https://imed.co.ua/

19. Статус розробки функціоналу eHealth в Медичних Інформаційних Системах [Електронний ресурс]. – Режим доступу: https://ehealth.gov.ua/development-status/

20. Офіційний Сайт Міністерства охорони здоров'я України [Електронний ресурс]. – Режим доступу: https://moz.gov.ua/article/news/moz-rozpochav-robotu-nad-rozvitkom-telemedichnih-tehnologij-i-poslug-v-ukraini

21. Поточна статистика по коронавірусу в Україні. Офіційний Сайт Міністерства фінансів України [Електронний ресурс]. – Режим доступу: https://index.minfin.com.ua/reference/coronavirus/ukraine/

22. Концептуальне проєктування БД [Електронний ресурс]. – Режим доступу: https://studopedia.su/13_103235_kontseptualne-proektuvannya-bd.html

23. Осипов, Д.Л. Базы данных и Delphi. Теория и практика [Текст]/ Д.Л. Осипов. – СПб.: БХВ-Петербург, 2011. – 752 с.: ил.

24. Нормалізація баз даних [Електронний ресурс]. – Режим доступу: http://ukrbukva.net/92785-Normalizaciya-bazdannyh.html

25. Організація баз даних та знань. Тема 3 – Проєктування бази даних [Електронний ресурс]. – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151013153156/44233/index.html

26. Хансен, Г. Базы данных: разработка и управление: пер. с англ. [Текст] / Г. Хансен, Дж. Хансен. – М.: Издательство БИНОМ, 1999. – 196 с.

27. Васвани, В. MySQL: использование и администрирование = MySQL Database Usage & Administration [Текст] / В. Васвани. – М.: Питер, 2011. – 368 с.

28. Суэринг, С. PHP и MySQL. Библия программиста, 2-е издание = PHP 6 and MySQL 6 Bible [Текст] / С. Суэринг, Т. Конверс, Дж. Парк. – М.: Диалектика, 2010. – 912 с.

29. Документация к PostgreSQL 11 [Электронный ресурс]. – Режим доступа: https://postgrespro.ru/docs/postgresql/11/index

30. Зарезервовані змінні у PHP [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/en/reserved.variables.php

ДОДАТОК А

Технічне завдання

Вступ

Метою дипломної кваліфікаційної роботи магістра є розробка дистанційної системи консультацій сімейного лікаря, яка дозволить оперативно обчислювати рівень ургентності пацієнтів та виконувати сортування звернень до сімейного лікаря в залежності від тяжкості захворювання відповідного пацієнта.

Розроблювана дистанційна система дозволить виконувати наступні функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації поточної температури тіла та артеріального тиску, а також основних симптомів захворювань;

– зберігання показників схильності пацієнтів для подальшого обчислення рівню ургентності;

– можливість для пацієнтів залишати у зверненні коментарі щодо стану свого здоров’я;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні.

А.1 Підстава для розробки

Підставою для розробки послужило завдання на дипломну кваліфікаційну роботу магістра за темою: «Дослідження та програмна реалізація дистанційної системи консультацій сімейного лікаря з урахуванням рівня ургентності пацієнтів», затверджену наказом по Національному університету «Запорізька політехніка» № 435 від 05 листопада 2021 року.

А.2 Призначення розробки

Дистанційну систему консультацій сімейного лікаря призначено для створення пацієнтами не виходячи з дому звернень до свого сімейного лікаря, вказуючи поточні параметри температури тіла, артеріального тиску та основних симптомів захворювання, що дозволить лікарю звернути увагу по-перше саме на того пацієнта, який як найбільше потребує медичної допомоги.

А.3 Вимоги до програми

А.3.1 Вимоги до структури програми

Дистанційна система повинна мати такі структурні елементи:

– клієнтська частина (Пацієнт), що призначена для пацієнта, який вносить дані у звернення до лікаря;

– клієнтська частина (Лікар), що призначена для сімейного лікаря, який виконує обробку звернень пацієнтів до нього;

– серверна частина, яка виконує функцію віддаленого серверу для зберігання звернень та обчислення рівня ургентності пацієнтів;

– база даних, яка містить інформацію щодо звернень пацієнтів, їх симптомів захворювань, схильностей та основних біометричних показників таких як, поточна температура тіла і артеріальний тиск;

– модуль аналітики, який виконує обчислення рівня ургентності пацієнтів для впорядковування звернень.

А.3.2 Вимоги до функціональних характеристик

Виходячи з постановки задачі, було сформовано основні функціональні вимоги до програми.

Розроблювана дистанційна система має дозволяти виконувати наступні функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації поточної температури тіла та артеріального тиску, а також основних симптомів захворювань;

– зберігання показників схильності пацієнтів для подальшого обчислення рівню ургентності;

– можливість для пацієнтів залишати у зверненні коментарі щодо стану свого здоров’я;

– моніторинг лікарем як відкритих звернень пацієнтів, так і завершених;

– моніторинг пацієнтом як відкритих, так і завершених до лікаря звернень;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні.

Головною відмінністю розроблюваної дистанційної системи повинен бути спосіб обчислення рівня ургентності пацієнта в залежності від його схильностей до захворювань та поточного стану здоров’я.

А.3.3 Вимоги до зберігання даних

Створена дистанційна система дозволить зберігати та використовувати для обчислення рівня ургентності пацієнта:

– основні персональні дані пацієнта (П.І.Б. та інше);

– адреса реєстрації пацієнта;

– контактні телефони пацієнта;

– потенційно нормальна температура тіла;

– нормальний систолічний тиск у здоровому стані;

– нормальний діастолічний тиск у здоровому стані;

– дата та час звернення пацієнта до сімейного лікаря;

– показники поточної температури тіла пацієнта;

– показники поточного систолічного тиску пацієнта;

– показники поточного діастолічного тиску пацієнта;

– перелік симптомів захворювання;

– додатковий текст від пацієнта щодо стану здоров’я;

– рекомендації лікаря (або відповідь на звернення).

А.3.4 Вимоги до інтерфейсу програми

Дистанційна система повинна мати графічний інтерфейс для зручного використання. Графічний інтерфейс має бути ергономічним та дружнім для користувачів, із стартовою сторінкою. Взаємодія дистанційної системи із користувачем повинна відбуватись через поля для введення даних.

А.4 Обґрунтований вибір апаратно-технічних засобів, операційної системи і мови програмування

Для функціонування дистанційної системи необхідно мати будь-який комп’ютер, який під’єднано до всесвітньої мережі Інтернет із вбудованим або встановленим до будь-якої операційної системи браузером з останніми оновленнями.

Рекомендовані вимоги до програмного забезпечення серверу щодо найкращої швидкодії дистанційної системи:

– операційна система: Debian GNU Linux версії 9.x або вище чи Ubuntu 16.04 LTS або версіями вище;

– встановлена СКБД MariaDB версії 10.6.4;

– встановлене PHP версії 7.x;

– встановлене Apache 2.4.

Особливих вимог до експлуатації системи не пред’являється.

А.5 Основні обмеження на установку і використання програми

Особливих вимог до експлуатації дистанційної системи не пред’являється. За виключенням актуального встановлення останніх оновлень операційної системи та браузеру.

ДОДАТОК Б

Опис програми

Б.1 Загальні відомості

Створена дистанційна система консультацій дозволить зберігати інформацію щодо звернень пацієнтів, їх деяких біометричних показників, наявні симптоми захворювання, а також відповіді та рекомендації сімейного лікаря:

– основні персональні дані пацієнта (П.І.Б. та інше);

– адреса реєстрації пацієнта;

– контактні телефони пацієнта;

– потенційно нормальна температура тіла;

– нормальний систолічний тиск у здоровому стані;

– нормальний діастолічний тиск у здоровому стані;

– дата та час звернення пацієнта до сімейного лікаря;

– показники поточної температури тіла пацієнта;

– показники поточного систолічного тиску пацієнта;

– показники поточного діастолічного тиску пацієнта;

– перелік симптомів захворювання;

– додатковий текст від пацієнта щодо стану здоров’я;

– рекомендації лікаря (або відповідь на звернення).

Для створення бази даних дистанційної системи використовувалася СКБД MariaDB версії 10.6.4. Систему керування базами даних MariaDB встановлено на виділеному сервері для функціонування під операційною системою Linux.

Б.2 Функціональне призначення

Основним функціональним призначенням дистанційної системи є створення пацієнтами не виходячи з дому звернень до свого сімейного лікаря, вказуючи поточні параметри температури тіла, артеріального тиску та основних симптомів захворювання.

Дистанційна система дозволить виконувати такі функції:

– зберігання у базі даних звернень пацієнтів з можливістю реєстрації поточної температури тіла та артеріального тиску, а також основних симптомів захворювань;

– зберігання показників схильності пацієнтів для подальшого обчислення рівню ургентності;

– можливість для пацієнтів залишати у зверненні коментарі щодо стану свого здоров’я;

– моніторинг лікарем як відкритих звернень пацієнтів, так і завершених;

– моніторинг пацієнтом як відкритих, так і завершених до лікаря звернень;

– можливість для лікаря дистанційно зробити медичний висновок з рекомендаціями до лікування та призначення ліків;

– автоматичне сортування звернень пацієнтів для лікаря в залежності від тяжкості симптомів та найгіршості біометричних показників при захворюванні;

– можливість для лікаря проводити пошук у базі звернень за різними категоріями та біометричними даними пацієнтів.

Б.3 Опис логічної структури

Структура створеної дистанційної системи складається із трьох частин – клієнтська (Пацієнт), клієнтська (Лікар) та серверна частина. Серверна частина містить два компоненти:

– база даних;

– модуль аналітики.

Б.4 Використані технічні засоби та умови роботи програми

Б.4.1 Використані технічні засоби

Розробка дистанційної системи та створення бази даних відбувалися на комп’ютері, де було встановлено операційну систему Linux Debian версії 10. При проєктуванні бази даних у якості СКБД було застосовано MariaDB версії 10.6.4. Для створення вебінтерфейсів дистанційної системи було застосовано наступні мови та технології: HTML 5, PHP, JavaScript, технологію CSS, бібліотеку jQuery.

Б.4.2 Умови роботи програми

Для забезпечення роботи дистанційної системи необхідно мати будь-який комп’ютер, який під’єднано до всесвітньої мережі Інтернет із вбудованим або встановленим до будь-якої операційної системи браузером.

Для функціонування серверної частини найкращим вибором буде:

– операційна система: Debian GNU Linux версії 9.x або вище чи Red Hat Enterprise Linux версії 7.x або вище чи Ubuntu 16.04 LTS або версіями вище;

– встановлена СКБД MariaDB версії 10.6.4;

– встановлене PHP версії 7.x;

– встановлене Apache 2.4.

Б.5 Виклик, завантаження і звернення до програми

До виклику дистанційної системи лікарі та пацієнти можуть перейти через браузер за доменним ім’ям або за IP-адресою. Адміністратор бази даних (або системи) має можливість змінювати важливі налаштування тільки якщо він знаходиться в локальній мережі серверу.

ДОДАТОК В

Тексти скриптів SQL

В.1 Створення таблиці «ActiveDoc»

```
CREATE TABLE `ActiveDoc` (
  `ID` bigint(20) NOT NULL AUTO_INCREMENT,
  `DocId` bigint(20) NOT NULL,
  `IsActivated` tinyint(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (`ID`),
  KEY `ActiveDoc_FK` (`DocId`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.2 Створення таблиці «Admins»

```
CREATE TABLE `Admins` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `LFN` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'ПІБ Пацієнта',
  `Login` varchar(16) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Логін
адміністратора',
  `HashPassw` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Ґеш паролю',
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.3 Створення таблиці «Doctor»

```
CREATE TABLE `Doctor` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `LFN` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'П.І.Б. лікаря',
  `Photo` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT '/photo/nofoto.png' COMMENT
'Шлях до фото лікаря',
  `RecRegion` varchar(32) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Область пункту
прийому',
  `RecCitySett` varchar(32) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Місто або
село пункту прийому',
  `RecDepart` varchar(32) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Пункт прийому',
  `RecStreet` varchar(32) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Вулиця пункту
прийому',
  `RecHouse` varchar(4) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Дім пункту
прийому',
  `RecCorps` varchar(4) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Корпус пункту
прийому',
  `RecCab` varchar(4) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Кабінет пункту
прийому',
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.4 Створення таблиці «History»

```
CREATE TABLE `History` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `IdPatient` bigint(20) NOT NULL COMMENT 'Код пацієнта',
  `IdDoctor` bigint(20) NOT NULL COMMENT 'Код лікаря',
  `Numb` bigint(20) NOT NULL COMMENT 'Номер історії хвороби пацієнта',
  `StartTime` datetime NOT NULL COMMENT 'Дата відкриття історії хвороби',
  `HospCard` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Номер
лікарняного',
  `BegHCard` datetime DEFAULT NULL COMMENT 'Час початку лікарняного',
  `EndHCard` datetime DEFAULT NULL COMMENT 'Час закінчення лікарняного',
```

```
  `FinalDiagn` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Заключний
діагноз',
  `OpenFlag` tinyint(1) DEFAULT 0 COMMENT 'Прапорець відкриття історії хвороби',
  PRIMARY KEY (`Id`),
  KEY `History_FK_1` (`IdDoctor`),
  KEY `History_FK` (`IdPatient`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.5 Створення таблиці «Patient»

```
CREATE TABLE `Patient` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `LFN` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'ПІБ Пацієнта',
  `Photo` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Шлях до фото
пацієнта',
  `BirthDate` datetime NOT NULL COMMENT 'Дата народження',
  `Sex` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'Стать пацієнта False=Жінка',
  `NormTemp` float NOT NULL DEFAULT 36.6 COMMENT 'Нормальна температура тіла',
  `NormSistT` int(11) NOT NULL DEFAULT 120 COMMENT 'Нормальний артеріальний
сістолічний тиск',
  `NormDiaT` int(11) NOT NULL DEFAULT 80 COMMENT 'Нормальний артеріальний діастолічний
тиск',
  `UrgentPow` float NOT NULL DEFAULT 0 COMMENT 'Рівень ургентності',
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.6 Створення таблиці «Predispos»

```
CREATE TABLE `Predispos` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `IdPatient` bigint(20) NOT NULL COMMENT 'Код пацієнта',
  `Predis` text COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Код схильності
пацієнта',
  PRIMARY KEY (`Id`),
  KEY `Predispos_FK` (`IdPatient`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.7 Створення таблиці «PrivatDoc»

```
CREATE TABLE `PrivatDoc` (
  `Id` bigint(20) NOT NULL COMMENT 'Ключове поле',
  `Login` varchar(16) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Логін лікаря',
  `HashPassw` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Ґеш паролю',
  `Phone1` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Телефон 1',
  `Phone2` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Телефон 2',
  `Viber` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Viber',
  `Telegram` varchar(32) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Логін у TE',
  `Email` varchar(64) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Електронна
пошта',
  PRIMARY KEY (`Id`),
  UNIQUE KEY `PrivatDoc_UN_log` (`Login`),
  UNIQUE KEY `PrivatDoc_UN_hash` (`HashPassw`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.8 Створення таблиці «PrivatPat»

```
CREATE TABLE `PrivatPat` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `Login` varchar(16) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Логін пацієнта',
  `HashPassw` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Ґеш паролю',
  `PasspSerie` varchar(8) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Серія
паспорту',
  `PasspNumb` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Номер
паспорту',
  `PasspProduc` varchar(64) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Видано
паспорт',
  `RegCountry` varchar(32) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Країна
регістрації',
  `RegRegion` varchar(32) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Область
регістрації',
  `RegCitySett` varchar(32) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Місто або
село ',
  `RegStreet` varchar(32) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Вулиця
реєстрації',
  `RegHouse` varchar(4) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Дім
реєстрації',
  `RegCorps` varchar(4) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Корпус
реєстрації',
  `RegApart` varchar(4) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Квартира
реєстрації',
  `Phone1` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Телефон 1',
  `Phone2` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Телефон 2',
  `Viber` varchar(16) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Телефон 3',
  `Telegram` varchar(64) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Логін у
Телеграмі',
  `Email` varchar(32) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`Id`),
  UNIQUE KEY `PrivatPat_UN_log` (`Login`),
  UNIQUE KEY `PrivatPat_UN_hash` (`HashPassw`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.9 Створення таблиці «RecChart»

```
CREATE TABLE `RecChart` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `IdDoctor` bigint(20) NOT NULL COMMENT 'Код лікаря',
  `RecDate` datetime NOT NULL COMMENT 'Дата прийому',
  `BegTime` datetime NOT NULL COMMENT 'Час початку прийому',
  `EndTime` datetime DEFAULT NULL COMMENT 'Час закінчення прийому',
  `Cabinet` varchar(4) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Кабінет пункту
прийому',
  PRIMARY KEY (`Id`),
  KEY `RecChart_FK` (`IdDoctor`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.10 Створення таблиці «Session»

```
CREATE TABLE `Session` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `IdHistory` bigint(20) NOT NULL COMMENT 'Код історії хвороби',
  `CurrTime` datetime NOT NULL COMMENT 'Дата звернення пацієнта',
  `CurrTemp` float NOT NULL COMMENT 'Поточна температура тіла',
  `CurrSisT` int(11) NOT NULL COMMENT 'Поточний артеріальний систолічний тиск',
  `CurrDiaT` int(11) NOT NULL COMMENT 'Поточний артеріальний діастолічний тиск',
```

```
  `ApdxText` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT 'Додатковий
коментар від пацієнта',
  `DocRecom` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL COMMENT
'Рекомендації лікаря',
  PRIMARY KEY (`Id`),
  KEY `Session_FK` (`IdHistory`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.11 Створення таблиці «SimDoc»

```
CREATE TABLE `SimDoc` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT,
  `DocId` bigint(20) DEFAULT NULL,
  `PatId` bigint(20) DEFAULT NULL,
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.12 Створення таблиці «Symptom»

```
CREATE TABLE `Symptom` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `IdSession` bigint(20) NOT NULL COMMENT 'Код сеансу',
  `Symp` bigint(20) NOT NULL COMMENT 'Симптоми пацієнта',
  PRIMARY KEY (`Id`),
  KEY `Symptom_FK` (`IdSession`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.13 Створення таблиці «VaccCard»

```
CREATE TABLE `VaccCard` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `IdPatient` bigint(20) NOT NULL COMMENT 'Код пацієнта',
  `VaccDate` date NOT NULL COMMENT 'Дата вакцинації',
  `VaccDescr` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Опис
вакцинації',
  PRIMARY KEY (`Id`),
  KEY `VaccCard_FK` (`IdPatient`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.14 Створення таблиці «_Predis»

```
CREATE TABLE `_Predis` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `Name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Назва схильності',
  `Score` float NOT NULL COMMENT 'Коефіцієнт впливу',
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
```

В.15 Створення таблиці «_Symp»

```
CREATE TABLE `_Symp` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'Ключове поле',
  `Name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL COMMENT 'Назва симптому',
  `Score` float NOT NULL COMMENT 'Коефіцієнт впливу',
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

В.16 Поповнення таблиці «ActiveDoc»

```
INSERT INTO ActiveDoc (ID, DocId, IsActivated)
VALUES(1, 1, 1);
```

В.17 Поповнення таблиці «Admins»

```
INSERT INTO Admins (Id, LFN, Login, HashPassw)
VALUES(1, 'ADMIN', 'admin',
'pbkdf2_sha512$10000$LJ3yC2W6x4DGScwDre/zbg==$N2adHh0f7vI1NOF8vQyYwdUn16WCN7q1ur0sclAd
XYj4y1hIP29pC6FH1bbJ/1Q/RmiwzGmTAuwZoki+TSvPzw==');
```

В.18 Поповнення таблиці «Doctor»

```
INSERT INTO Doctor (Id, LFN, Photo, RecRegion, RecCitySett, RecDepart, RecStreet,
RecHouse, RecCorps, RecCab)
VALUES(1, 'Биков Андрій Євгенович', 'photo/3.png', 'Запорізька', 'Запоріжжя',
'Комунарський', 'Космічна', '99', NULL, '16');
```

В.19 Поповнення таблиці «History»

```
INSERT INTO History (Id, IdPatient, IdDoctor, Numb, StartTime, HospCard, BegHCard,
EndHCard, FinalDiagn, OpenFlag)
VALUES(1, 1, 1, 20211211000100, '2021-12-11 16:42:47.000', NULL, NULL, NULL, NULL, 1);
```

В.20 Поповнення таблиці «Patient»

```
INSERT INTO Patient (Id, LFN, Photo, BirthDate, Sex, NormTemp, NormSistT, NormDiaT,
UrgentPow)
VALUES(1, 'Петренко Борис Вікторович', '/photo/nofoto.png', '1995-12-23 00:00:00.000',
1, 36.6, 150, 100, 17.3864);
```

В.21 Поповнення таблиці «Predispos»

```
INSERT INTO Predispos
(Id, IdPatient, Predis)
VALUES(1, 1, '2');
```

В.22 Поповнення таблиці «PrivatDoc»

```
INSERT INTO PrivatDoc (Id, Login, HashPassw, Phone1, Phone2, Viber, Telegram, Email)
VALUES(1, 'doc1',
'pbkdf2_sha512$10000$2kCHIU5ysvzMq61EZ7AELA==$libGv45qjIf9YP+V/KTPG1Xr43vgmm9dI0y0Mydv
h0gyfQ1P1Gf6IJDlik2G+BDLYVka5pav04HywlX7GfwlIQ==', '+380661234567', NULL,
'+380661234567', '@doc1', 'doc1@gmail.com');
```

В.23 Поповнення таблиці «PrivatPat»

```
INSERT INTO PrivatPat (Id, Login, HashPassw, PasspSerie, PasspNumb, PasspProduc,
RegCountry, RegRegion, RegCitySett, RegStreet, RegHouse, RegCorps, RegApart, Phone1,
Phone2, Viber, Telegram, Email)
VALUES(1, 'pat1',
'pbkdf2_sha512$10000$N0qEjKtAVJyJEHX5K2VYOg==$A6X7nugS3jbyWzLLxMKObPPvM/z48LFoJLkAeb//
6YIhE4Ihrq6MXdBhIHM2y0tBGQLCMRDxpKBzUnItDITtyA==', 'СЮ', '123456', 'ДМС у Запорізькій
області', 'Україна', 'Запорізька', 'Запоріжжя', 'Космічна', '12', NULL, '1',
'+380991234567', NULL, '+380991234567', '@pat1', 'pat1@gmail.com');
```

В.24 Поповнення таблиці «Session»

```
INSERT INTO Session (Id, IdHistory, CurrTime, CurrTemp, CurrSisT, CurrDiaT, ApdxText,
DocRecom)
VALUES(1, 1, '2021-12-11 16:42:47.000', 38.9, 120, 150, 'zzzz zzzz', NULL);
```

В.25 Поповнення таблиці «SimDoc»

```
INSERT INTO SimDoc
(Id, DocId, PatId)
VALUES(1, 1, 1);
```

В.26 Поповнення таблиці «Symptom»

```
INSERT INTO Symptom
(Id, IdSession, Symp)
VALUES(1, 1, 1);
```

В.27 Поповнення таблиці «VaccCard»

```
INSERT INTO VaccCard
(Id, IdPatient, VaccDate, VaccDescr)
VALUES(1, 2, '2019-10-01', 'Щеплення №2019-10-01 АКДС');
```

В.28 Поповнення таблиці «_Predis»

```
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(1, 'Алергічні захворювання', 6.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(2, 'Неврологічні хвороби (у т.ч. інсульт)', 4.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(3, 'Гіпертонічні хвороби', 4.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(4, 'Легеневі захворювання', 4.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(5, 'Захворювання кровоносної системи', 3.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(6, 'Серцево-судинні хвороби', 3.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(7, 'Онкологічні захворювання', 2.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(8, 'Хвороби печінки', 1.0);
INSERT INTO `_Predis` (Id, Name, Score)
VALUES(9, 'Хвороби нирок', 1.0);
```

В.29 Поповнення таблиці «_Symp»

```
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(1, 'Підвищення температури тіла', 2.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(2, 'Сухий кашель', 2.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(3, 'Затруднення дихання', 2.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(4, 'Очна біль, почервоніння або подразнення', 2.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(5, 'Втрата координації руху та розрахунку чисел', 2.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(6, 'Нежить', 2.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(7, 'Різке змінення артеріального тиску', 2.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(8, 'Інші больові відчуття (найчастіше у шлунку)', 2.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(9, 'Діарея', 2.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(10, 'Головний біль', 2.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(11, 'Втрата нюхових або смакових відчуттів', 1.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(12, 'Спонтанний висип на різних місцях шкіри', 1.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(13, 'Зміна кольору шкіри (блідість обличчя)', 1.5);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(14, 'Зниження температури тіла', 1.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(15, 'Біль у горлі або в ухах', 1.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(16, 'Періодичний біль у серці', 1.0);
INSERT INTO `_Symp` (Id, Name, Score)
VALUES(17, 'Втомленість', 1.0);
```

ДОДАТОК Г

Тексти файлів вебсайту

Г.1 Файл index.php

```
<?php
ob_start();
session_start();
error_reporting(1);
include("connect.php");
include "process.php";
if (!isset($_SESSION['opt'])) {
  header('Location: login.php');}
$page = (isset($_GET['page']) ? $_GET['page'] : 'main');?>
<html>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="css/main.css" rel="stylesheet" />
<script type="text/javascript" src="jquery.min.js"></script>
<head>
  <title>Медична система</title>
</head>
<body>
  <header>
    <div class="navbar">
      <a href="/?page=main">Головна сторінка</a>
      <div class="enterbtn">
        <?php
        if (isset($_SESSION['opt'])) {
          switch ($_SESSION['opt']) {
            case 'adminform':
              echo '<a href="?page=admin">Адміністрування</a>';
              break;
            case 'docform':

              echo '<a href="?page=cab">Кабінет лікаря</a>';
              break;
            case 'patientform':

              echo '<a href="?page=card">Кабінет пацієнта</a>';
              break;       }        }
        ?>
        <a href="/register.php">Реєстрація</a>
        <?php
        if (!isset($_SESSION['use'])) {
          echo '<a href="?page=login" >Увійти</a>';
        } else {
          echo ' <a href="?page=logout">Вихід</a>';
        } ?>
      </div>
    </div>
  </header>
  <?php include basename($page) . '.php'; ?>
</body>
</html>
```

Г.2 Файл main.php

```
<div>
    <h1>Головна сторінка</h1>
</div>
<div class="faq">
    <h4> </h4>
    <br>
    <p><a href="?page=card">Кабінет пацієнта</a> Основна сторінка пацієнта.<strong> In
development</strong></p>
```

```
    <p><a href="?page=cab">Кабінет лікаря</a> Основна сторінка роботи лікаря.<strong>
In development</strong></p>
        <br>
    <br>
    <h1> Інші службові сторінки</h1>
    <br>
    <p><a href="?page=admin.php">Адміністрування</a>, через яку адміністратор може
видалити користувачів або змінити стан лікарів(активний/заблокований)  </p>
    <p><a href="login.php">Сторінка входу</a>, через яку відбувається вхід, як
пацієнтів, так і лікарів</p>
    <p><a href="register.php">Сторінка реєстрації</a> всі нові користувачі мають
необхідність у цій сторінці.</p>
    <p><a href="logout.php">Вихід</a> відключає користувача від системи.</p>
</div>
```

Г.3 Файл `register.php`

```
<?php
require('connect.php');
require('process.php');?>
<script>
  function changeOptions(selectEl) {
    let selectedValue = selectEl.options[selectEl.selectedIndex].value;
    let subForms = document.getElementsByClassName('regform')
    for (let i = 0; i < subForms.length; i += 1) {
      if (selectedValue === subForms[i].name) {
        subForms[i].setAttribute('style', 'display:block')
      } else {
        subForms[i].setAttribute('style', 'display:none')
      }    }  }</script>
<head>
  <title>Реєстрація</title>
  <link rel="stylesheet" href="css/reg.css">
  <link rel="stylesheet" href="css/main.css">
</head>
<body>
  <div class="container">
<form>      <h4>Реєстрація облікового запису </h4>
      <select name="form_select" onchange="changeOptions(this)">
        <option disabled value="" selected="selected">Оберіть зі списку</option>
        <option value="patientform">Пацієнта</option>
        <option value="docform">Лікаря</option>
      </select>
    </form>
    <form method="POST" action="" name="patientform" class="regform"
style="display:none">
      <div class="container>">
        <input type="text" hidden name="form_select" value="patientform" />
        <div class="row">
          <h4>Введіть ваші дані</h4>
          <div class="input-group">
            <div class="col-third">
              <input required type="text" name="last_name" placeholder="Прізвище" />
            </div>
            <div class="col-third input-2-3">
              <input required type="text" name="first_name" placeholder="Ім'я" />
            </div>
            <div class="col-third input-2-3">
              <input required type="text" name="middle_name" placeholder="По-батькові"
/>
            </div>
          </div>
          <div class="row">
            <h4>Введіть дату народження</h4>
            <div class="input-group">
              <div class="col-third">
                <select class="day" name="day" required>
```

```
                  <option selected disabled value="">День</option>
                  <option value="01">1</option>
                  <option value="02">2</option>
                  <option value="03">3</option>
                  <option value="04">4</option>
                  <option value="05">5</option>
                  <option value="06">6</option>
                  <option value="07">7</option>
                  <option value="08">8</option>
                  <option value="09">9</option>
                  <option value="10">10</option>
                  <option value="11">11</option>
                  <option value="12">12</option>
                  <option value="13">13</option>
                  <option value="14">14</option>
                  <option value="15">15</option>
                  <option value="16">16</option>
                  <option value="17">17</option>
                  <option value="18">18</option>
                  <option value="19">19</option>
                  <option value="20">20</option>
                  <option value="21">21</option>
                  <option value="22">22</option>
                  <option value="23">23</option>
                  <option value="24">24</option>
                  <option value="25">25</option>
                  <option value="26">26</option>
                  <option value="27">27</option>
                  <option value="28">28</option>
                  <option value="29">29</option>
                  <option value="30">30</option>
                  <option value="31">31</option>
                </select>
              </div>
              <div class="col-third input-2-3">
                <select class="month" name="month" required>
                  <option selected disabled value="">Місяць</option>
                  <option value="01">Січень</option>
                  <option value="02">Лютий</option>
                  <option value="03">Березень</option>
                  <option value="04">Квітень</option>
                  <option value="05">Травень</option>
                  <option value="06">Червень</option>
                  <option value="07">Липень</option>
                  <option value="08">Серпень</option>
                  <option value="09">Вересень</option>
                  <option value="10">Жовтень</option>
                  <option value="11">Листопад</option>
                  <option value="12">Грудень</option>
                </select>
              </div>
              <div class="col-third input-2-3">
                <input required type="text" name="year" pattern="[12]{1}[90]{1}[0-
9]{2}" placeholder="Рік" />
              </div>       </div>          </div>
          <h4>Оберіть стать</h4>
          <div class="input-group">
            <div class="row">
              <div class="form_radio_btn">
                <input requiered id="radio-1" type="radio" name="gen" value="0">
        <label for="radio-1">Чоловік</label>              </div>
              <div class="form_radio_btn">
                <input id="radio-2" type="radio" name="gen" value="1">
                <label for="radio-2">Жінка</label>
              </div>  </div>        </div>
          <h4>Введіть реєстраційні данні</h4>
<div class="input-group"><div class="row"><div class="col-half">
                <label for="passpserie">Серія паспорту</label>
                <input name="passpserie" type="text" pattern="[А-Я][ID]{2}">
</div>
              <div class="col-half">
```

```
              <label for="passpnumb">Номер паспорту</label>
              <input name="passpnumb" type="text" required>
            </div><div>
          <label for="passpproduc">Орган видачі паспорту</label>
          <input name="passpproduc" type="text" required>
          <label for="regcountry">Країна реєстрації</label>
          <input name="regcountry" type="text" required>
          <label for="regregion">Область реєстраціїї</label>
          <input name="regregion" type="text" required>
          <label for="regcity">Місто або село реєстрації</label>
          <input name="regcity" type="text" required>
          <label for="regstreet">Вулиця реєстрації</label>
          <input name="regstreet" type="text" required>
          <div class="col-third input-2-3">
            <label for="reghouse">Дім реєстрації</label>
            <input name="reghouse" type="text" required>
          </div>
          <div class="col-third input-2-3">
            <label for="regcorps">Корпус реєстрації</label>
            <input name="regcorps" type="text">
          </div>
          <div class="col-third input-2-3">
            <label for="regapart">Квартира реєстрації</label>
            <input name="regapart" type="text" required>
          </div>
          <label for="phone1">Телефон 1</label>
          <input name="phone1" type="text" pattern="+[0-9]{12}"
placeholder="+XXXXXXXXXXXX" required>
          <label for="phone2">Телефон 2</label>
          <input name="phone2" type="text" pattern="+[0-9]{12}"
placeholder="+XXXXXXXXXXXX">
          <label for="viber">Viber</label>
          <input name="viber" type="text" pattern="+[0-9]{12}"
placeholder="+XXXXXXXXXXXX">
          <label for="telegram">Логін у телеграм(нікнейм)</label>
          <input name="telegram" type="text" placeholder="@XXXXXXXXXXXX">
          <label for="email">E-mail</label>
          <input name="email" type="email" placeholder="XXXXXXXX@XXX.XXX" required>
        </div>
        <div class="input-group">
          <input required type="text" name="username" placeholder="Логін" />
        </div>
        <div class="row">
          <input required type="password" name="password" placeholder="Пароль" />
        </div>        </div>      </div>      <div class="row">
      <input style="margin-top:13px;" type="submit" name="register"
value="Зареєструватися" />
    </div>
  </form>
  <form method="POST" action="" id='docform' name="docform" class="regform"
style="display:none">
    <div class="container>">
      <input type="text" hidden name="form_select" value="docform" />
      <div class="row">
        <h4>Введіть ваші дані</h4>
        <div class="input-group">
          <div class="col-third">
            <input required type="text" name="last_name" placeholder="Прізвище" />
          </div>
          <div class="col-third input-2-3">
            <input required type="text" name="first_name" placeholder="Ім'я" />
          </div>
          <div class="col-third input-2-3">
            <input required type="text" name="middle_name" placeholder="По-батькові"
/>
          </div>
        </div>
        <h4>Введіть адресу місця роботи</h4>
        <div class="input-group">
          <label for="recregion">Область</label>
          <input name="recregion" type="text" required>
```

```
            <label for="reccity">Місто або село</label>
            <input name="reccity" type="text" required>
            <label for="recdepart">Клініка</label>
            <input name="recdepart" type="text" required>
            <label for="recstreet">Вулиця</label>
            <input name="recstreet" type="text" required>
            <div class="input-group">
              <div class="col-third input-2-3">
                <label for="rechouse">Дім</label>
                <input name="rechouse" type="text" required>
              </div>
              <div class="col-third input-2-3">
                <label for="reccorps">Корпус</label>
                <input name="reccorps" type="text">
              </div>
              <div class="col-third input-2-3">
                <label for="reccab">Кабінет</label>
                <input name="reccab" type="text" required>
              </div>
            </div>
            <div class="input-group">
              <label for="phone1">Телефон 1</label>
              <input name="phone1" type="text" pattern="+[0-9]{12}"
placeholder="+XXXXXXXXXXX" required>
              <label for="phone2">Телефон 2</label>
              <input name="phone2" type="text" pattern="+[0-9]{12}"
placeholder="+XXXXXXXXXXX">

              <label for="viber">Viber</label>
              <input name="viber" type="text" pattern="+[0-9]{12}"
placeholder="+XXXXXXXXXXX">

              <label for="telegram">Логін у телеграм(нікнейм)</label>
              <input name="telegram" type="text" placeholder="@XXXXXXXXXXX">

              <label for="email">E-mail</label>
              <input name="email" type="email" placeholder="XXXXXXXX@XXX.XXX"
required>
            </div>
            <div class="input-group">
              <input required type="text" name="username" placeholder="Логін">
</div>
            <div class="row">
              <input required type="password" name="password" placeholder="Пароль">
            </div>          </div>        </div>      </div>
      <div class="row">
        <input type="submit" name="register" value="Зареєструватися" />
      </div>    </form>
    <p>Вже маєте свій аккаунт? <a href="login.php"> Увійдіть</a></p>
    <?php
    if (isset($_POST['register'])) {
      if ($_POST['form_select'] == 'docform') {
        $password = make_password($_POST['password']);
        $LFN = $_POST['last_name'] . ' ' . $_POST['first_name'] . ' ' .
$_POST['middle_name'];

        $sql = 'INSERT INTO PrivatDoc Values(default,' . $_POST['username'] . ',
        ' . $password . ',' . $_POST['phone1'] . ',' . $_POST['phone2'] . ',
        ' . $_POST['viber'] . ',' . $_POST['telegram'] . ',' . $_POST['email'] . ');
        INSERT INTO Doctor Values(default,' . $LFN . ',default,' . $_POST['recregion']
. ',
        ' . $_POST['reccity'] . ',' . $_POST['recdepart'] . ',' . $_POST['recstreet']
. ',
        ' . $_POST['rechouse'] . ',' . $_POST['reccorps'] . ',' . $_POST['reccab'] .
');';
        mysqli_query($con, $sql1);
        $last_id = mysqli_insert_id($con);
        //echo $last_id;
        $sql3 = 'INSERT INTO ActiveDoc Values(default,"' . $last_id . '",default);';
        if (mysqli_query($con, $sql3)) {
```

```
          echo '<p style="color:green; font-size:larger">Ви успішно зареєструвалися.
Увійдіть за своїм логіном';
          header('Refresh:3; URL=/');
        } else {
          echo "Error: " . $sql . ":-";
        }
      } else {
        $password = make_password($_POST['password']);
        $LFN = $_POST['last_name'] . ' ' . $_POST['first_name'] . ' ' .
$_POST['middle_name'];
        $sql = 'INSERT INTO PrivatPat Values(default,' . $_POST['username'] . ',' .
$password . ',' . $_POST['passpserie'] . ',' . $_POST['passpnumb'] . ',' .
$_POST['passpproduc'] . ',' . $_POST['regcountry'] . ',' . $_POST['regregion'] . ',' .
$_POST['regcity'] . ',' . $_POST['regstreet'] . ',' . $_POST['reghouse'] . ',' .
$_POST['regcorps'] . ',' . $_POST['regapart'] . ',' . $_POST['phone1'] . ',' .
$_POST['phone2'] . ',' . $_POST['viber'] . ',' . $_POST['telegram'] . ',' .
$_POST['email'] . ');
INSERT INTO Patient Values(default, ' . $LFN . ', default, ' . $_POST['year'] . '-' .
$_POST['month'] . '-' . $_POST['day'] . ', ' . $_POST['gen'] . ', default, default,
default, default);';
        if (mysqli_query($con, $sql)) {
          echo '<p style="color:green; font-size:larger">Ви успішно зареєструвалися.
Увійдіть за своїм логіном';
          header('Refresh:3; URL=/');
        } else {
          echo "Error: " . $sql . ":-";
        }      }    };    ?>
  </div>
</body>
</html>
```

Г.4 Файл `login.php`

```
<?php
session_start();
require_once "connect.php";
require_once "process.php";
?>
<html>
<meta charset="utf-8" />


<head>
  <title>Вхід у систему</title>
  <!--<script src="jquery.min.js"></script>-->
</head>

<body>
  <div class="login-page">
    <div class="form">
      <form action="" method="post" class="login-form">
        <h4> Увійти як </h4>
        <select name="option" id="option" class="logsel">
          <option disabled value="" selected="selected">Оберіть зі списку</option>
          <option value="patientform">Пацієнт</option>
          <option value="docform">Лікар</option>
          <option value="adminform">Адміністратор</option>
        </select>
        <!--<input type="hidden" id='select' name='select' value=""/>-->
        <input type="text" name="user" placeholder="Логін" />
        <input type="password" name="pass" placeholder="Пароль" />
        <input type="submit" name="login" value="Увійти" />
        <p>Не маєте свого облікового запису? <a class="hypervis"
href="register.php">Зареєструйтеся</a></p>
      </form>
      <?php
```

```
      $user = $pass = $result = $row = "";
      if (isset($_POST['login'])) {
        $user = $_POST['user'];
        $pass = $_POST['pass'];
        switch ($_POST['option']) {
          case 'patientform':
            $query_user = "select Id,HashPassw from PrivatPat where Login='$user';";
            $result = mysqli_query($con, $query_user);
        $count = mysqli_num_rows($result);
        $row = mysqli_fetch_row($result);
        $db_user_id = $row['0'];
        $db_user_pass = $row['1'];  if ($user != NULL && $count == '1' && $pass != ""
&& $pass != NULL && verify_Password($db_user_pass, $pass)) {
          $_SESSION['use'] = $user;
          $_SESSION['id'] = $db_user_id;
          $_SESSION['opt'] = $_POST['option'];
          echo 'Доброго дня, вас переадресує на головну через 2 секунди';
          header('Location: /');
        } else {
          echo '<p style="color:red; font-size:larger">Не вірний пароль чи логін</p>';
        }
            break;
          case 'docform':
            $query_user = "select p.Id,HashPassw,IsActivated from PrivatDoc p join
ActiveDoc on ActiveDoc.DocId=p.Id where Login=\"".$user."\";";
            $result = mysqli_query($con, $query_user);
        $count = mysqli_num_rows($result);
        $row = mysqli_fetch_row($result);
        $db_user_id = $row['0'];
        $db_user_pass = $row['1'];
        $activated=$row['2'];
        if ($user != NULL && $count == '1' && $pass != "" && $pass != NULL &&
$activated !=0 && verify_Password($db_user_pass, $pass)) {
          $_SESSION['use'] = $user;
          $_SESSION['id'] = $db_user_id;
          $_SESSION['opt'] = $_POST['option'];
          echo 'Доброго дня, вас переадресує на головну через 2 секунди';
          header('Location: /');
        } else {
          echo '<p style="color:red; font-size:larger">Не вірний пароль чи логін</p>';
          echo '<p style="color:red; font-size:larger">Зверніться до адміністратора,
ваш акаунт можливо ще заблоковано</p>';
        }
        break;
          case 'adminform':
            $query_user = "select Id,HashPassw from Admins where Login='$user';";
            $result = mysqli_query($con, $query_user);
        $count = mysqli_num_rows($result);
        $row = mysqli_fetch_row($result);
        $db_user_id = $row['0'];
        $db_user_pass = $row['1'];
        if ($user != NULL && $count == '1' && $pass != "" && $pass != NULL &&
verify_Password($db_user_pass, $pass)) {
          $_SESSION['use'] = $user;
          $_SESSION['id'] = $db_user_id;
          $_SESSION['opt'] = $_POST['option'];
          echo 'Доброго дня, вас переадресує на головну через 2 секунди';
          header('Location: /');
        } else {
          echo '<p style="color:red; font-size:larger">Не вірний пароль чи логін</p>';
        }
            break;
        }

      }
      ?>
    </div>
  </div>

  <style>
    .login-page {
```

```
  width: 360px;
  padding: 8% 0 0;
  margin: auto;
}

.form {
  position: relative;
  z-index: 1;
  background: #FFFFFF;
  max-width: 360px;
  margin: 0 auto 100px;
  padding: 45px;
  text-align: center;
  box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}

.form input {
  font-family: sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 100%;
  border: 0;
  margin: 0 0 15px;
  padding: 15px;
  box-sizing: border-box;
  font-size: 14px;
}

.form input[type=submit] {
  font-family: sans-serif;
  text-transform: uppercase;
  outline: 0;
  background: rgb(13, 225, 36);
  width: 100%;
  border: 0;
  padding: 15px;
  color: #FFffFF;
  font-size: 14px;
  -webkit-transition: all 0.3 ease;
  transition: all 0.3 ease;
  cursor: pointer;
}

.form input[type=submit]:hover,
.form input[type=submit]:active,
.form input[type=submit]:focus {
  background: #43A047;
}

.form .message {
  margin: 15px 0 0;
  color: #b3b3b3;
  font-size: 12px;
}

.form .message a {
  color: #4CAF50;
  text-decoration: none;
}

.form select {
  width: 100%;
  padding: 1em 0em;
  margin: 1em auto;
  text-align: center;
}

.logsel {
  width: 38em;
}
```

```
    .logsel::after {
      content: "";
      display: table;
    }

    .hypervis,
    .hypervis:visited {
      color: green;
    }

    body {
      -webkit-font-smoothing: antialiased;
      -moz-osx-font-smoothing: grayscale;
    }
  </style>
  <script>
    $('.message a').click(function() {
      $('form').animate({
        height: "toggle",
        opacity: "toggle"
      }, "slow");
    });
    /*
        $(function() {
          $("#option").change(function(){
          var optionSel= $('option:selected', this).attr('v1');
          $('#select').val(optionSel);
       });
    });*/
  </script>
```

Г.5 Файл connect.php

```
<?php
$servername = "localhost";
$username = "admin";
$password = "post1";
$dbname="web";

$con =  mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if ($con->connect_error) {
  die("Connection failed: " . $con->connect_error);
}
```

Г.6 Файл cab.php

```
<?php
//echo make_password(12345);
//print_r($_SESSION);
//print_r($_POST);
/*echo $order_id=date('Ymd') .'000000';
echo $order_id+= $_SESSION['select']*1000+5;*/
if (!isset($_SESSION['opt']) || $_SESSION['opt']!='docform') {
    header('Location: /');}
if (isset($_SESSION['select'])) {
    $chosenId = $_SESSION['select'];
}
if (isset($_POST['select'])) {
    $_SESSION['select'] = $_POST['select'];
    $chosenId = $_SESSION['select'];
}
$clientQuery = "Select * from Patient where Id=" . $chosenId . ";";
```

```
$clientQuery2 = "Select * from PrivatPat where Id=" . $chosenId . ";";
$clientQuery3 = "Select * from VaccCard where IdPatient=" . $chosenId . ";";

$PatientResult = mysqli_query($con, $clientQuery);
while ($row = mysqli_fetch_assoc($PatientResult)) {
    $chosenName = $row['LFN'];
    $chosenPhoto = $row["Photo"];
    $chosenBirth = $row['BirthDate'];
    $chosenNormStatus = $row['NormTemp'] . " " . $row["NormSistT"] . " " .
$row["NormDiaT"];
    $chosenSex = $row['Sex'];
}
$PatientResult2 = mysqli_query($con, $clientQuery2);
while ($row = mysqli_fetch_assoc($PatientResult2)) {
    $chosenPassp = $row['PasspSerie'] . "|" . $row["PasspNumb"] . "|" .
$row["PasspProduc"] . "|" . $row["RegCountry"] . "|" . $row["RegRegion"] . "|" .
$row["RegCitySett"] . "|" . $row["RegStreet"] . "|" . $row["RegHouse"] . "|" .
$row["RegCorps"] . "|" . $row["RegApart"];
    $chosenContacts = $row['Phone1'] . " " . $row['Phone2'] . " " . $row['Viber'] . "
" . $row['Telegram'] . " " . $row['Email'];
}
$PatientResult3 = mysqli_query($con, $clientQuery3);


?>
<div>
    <h1> Кабінет лікаря </h1>
</div>
<div class="urgency">
    <?php //echo $chosenId; calc_urgency($chosenId);?>
</div>
<div class="doctable" style="margin:5%;text-align:center">
    <table class="doctable">
        <thead>
            <tr>
                <th style="width:20%">Обраний пацієнт
                    <div id="scores" class="chosenpat">
                        <img src="<?php echo $chosenPhoto; ?>" style="max-
width:50px;">
                        <h5><?php echo $chosenName; ?></h5>
                        <h5><?php $r = explode(" ", $chosenBirth) ?></h5>
                        <h5><?php echo $r[0] . "\n"?></h5>
                    </div>
                </th>
                <th style="width:80%">Персональна інформація
                    <class="details">
                        <div class="row">
                            <table style="border:none; border-collapse:collapse"
cellspacing="0" cellpadding="0">
                                <tr>
                                    <th>Стать</th>
                                    <th rowspan='2'><button class='listbutton'
id="normBio">Показати норм. показання біометрики</button></th>
                                    <th rowspan='2'><button class='listbutton'
id="showPass">Показати паспортні дані</button></th>
                                    <th rowspan='2'><button class='listbutton'
id="showVacc">Показати карту вакцинації</button></th>
                                    <th rowspan='2'><button class='listbutton'
id="showPredis">Показати схильність до хвороб</button></th>
                                    <th rowspan='2'><button class='listbutton'
id="showConts">Показати контактні дані</button></th>
                                </tr>
                                <td>
                                    <p><?php if ($chosenSex == '0') {
                                            echo "<p>Жінка</p>";
                                        } else {
                                            echo "<p>Чоловік</p>";
                                        }; ?></h5>
                                </td>
                            </table>
                        </div>
```

```
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>
                    <div class="listview" style="overflow-y:scroll"></div>
                </td>
                <td>
                    <div class="messages" style="overflow-y:auto">
                    <?php
    //$sqlHistory = "select h.StartTime,h.Numb,s.CurrSisT,s.CurrDiaT,s.CurrTemp,
s.ApdxText, n.Name from History h join Session s on h.Id=s.IdHistory join Symptom m on
m.IdSession=s.Id join _Symp n on m.Symp=n.Id where h.IdPatient=" . $_SESSION['id'] .
";";
    $sqlHistory = "select * from History join Session on History.Id=Session.IdHistory
where History.IdPatient=" . $_SESSION['id'] . ";";
    //print_r($sqlHistory);
    $resHistory = mysqli_query($con, $sqlHistory);
    while ($row = mysqli_fetch_assoc($resHistory)) {

        $sqlSymps = "select a.Name from _Symp a join Symptom b on b.Symp=a.Id join
Session c on c.Id=b.IdSession  join History d on d.Id=c.IdHistory where d.Numb=" .
$row['Numb'] . ';';
        $resSymps = mysqli_query($con, $sqlSymps);
        $symp = '';
        while ($row2 = mysqli_fetch_assoc($resSymps)) {
            $symp .= ($row2['Name']) . '<br>';
        }
        $flag = ($row['OpenFlag'] == "0") ? "Відкрито" : "Закрито";
        echo '<button class="shbtn" onclick="">Номер запиту:' . $row['Numb'] . '
Статус запиту: ' . $flag . ' </button>';
        echo '<div class="toogle_ " style="display:none">
        <button><table>
            <tr>
                    <td>Номер запиту</td>
                    <td>' . $row['Numb'] . '</td>
                    <td>Статус запиту</td>
                    <td>' . $flag . '</td>
                </tr>
                <tr>
                    <td>Поточний діастолічний тиск</td>
                    <td>Поточний систолічний тиск</td>
                    <td>Поточна температура</td>
                    <td>Перелік симтомів</td>
                </tr>
                <tr>
                    <td>' . $row['CurrDiaT'] . '</td>
                    <td>' . $row['CurrSisT'] . '</td>
                    <td>' . $row['CurrTemp'] . '</td>
                    <td>' . $symp . '</td>
                </tr>
                <tr>
                    <td colspan="4">Коментар до запиту</td>
                </tr>
                <tr>
                    <td colspan="4">' . $row['ApdxText'] . '</td>
                </tr>
                <tr>
                    <td colspan="4">Назначення лікаря</td>
                </tr>
                <tr>
                    <td colspan="4">' . $row['DocRecom'] . '</td>
                </tr>
            </table></button>
        </div>';
    }
    ?>
                    </div>
                </td>
            </tr>
```

```
        </tbody>
    </table>
</div>
<div id="bio" class="modal">
    <div class="modal-content">
        <span class="close" id="close1">&times;</span>
        <div class="modal-table">
            <table class="modal-table">
                <thead>
                    <th colspan='2'>Нормальні біометричні показники</th>
                </thead>
                <?php $bio = explode(" ", $chosenNormStatus); ?>
                <tr>
                    <td>Нормальна температура</td>
                    <td><?php echo $bio[0]; ?></td>
                </tr>
                <tr>
                    <td>Нормальний систолічний тиск</td>
                    <td><?php echo $bio[1]; ?></td>
                </tr>
                <tr>
                    <td>Нормальний діастолічний тиск</td>
                    <td><?php echo $bio[2]; ?></td>
                </tr>
            </table>
        </div>
    </div>

</div>
<div id="contacts" class="modal">
    <div class="modal-content">
        <span class="close" id="close2">&times;</span>
        <div class="modal-table">
            <table class="modal-table">
                <thead>
                    <th colspan='2'>Контактні дані пацієнта</th>
                </thead>
                <?php $cont = explode(" ", $chosenContacts); ?>
                <tr>
                    <td>Телефон 1</td>
                    <td><?php echo $cont[0]; ?></td>
                </tr>
                <tr>
                    <td>Телефон 2</td>
                    <td><?php echo $cont[1]; ?></td>
                </tr>
                <tr>
                    <td>Viber</td>
                    <td><?php echo $cont[2]; ?></td>
                </tr>
                <tr>
                    <td>Telegram</td>
                    <td><?php echo $cont[3]; ?></td>
                </tr>
                <tr>
                    <td>Email</td>
                    <td><?php echo $cont[4]; ?></td>
                </tr>
            </table>
        </div>
    </div>
</div>
<div id="pass" class="modal">
    <div class="modal-content">
        <span class="close" id="close3">&times;</span>
        <div class="modal-table">
            <table class="modal-table">
                <thead>
                    <th colspan='2'>Пасспортні дані пацієнта</th>
                </thead>
```

```
                <?php $pp = explode("|", $chosenPassp); ?>
                <tr>
                    <td>Серія паспорту</td>
                    <td><?php echo $pp[0]; ?></td>
                </tr>
                <tr>
                    <td>Номер паспорту</td>
                    <td><?php echo $pp[1]; ?></td>
                </tr>
                <tr>
                    <td>Видано паспорт</td>
                    <td><?php echo $pp[2]; ?></td>
                </tr>
                <tr>
                    <td>Країна реєстрації</td>
                    <td><?php echo $pp[3]; ?></td>
                </tr>
                <tr>
                    <td>Область реєстрації</td>
                    <td><?php echo $pp[4]; ?></td>
                </tr>
                <tr>
                    <td>Місто або село реєстрації</td>
                    <td><?php echo $pp[5]; ?></td>
                </tr>
                <tr>
                    <td>Вулиця реєстрації</td>
                    <td><?php echo $pp[6]; ?></td>
                </tr>
                <tr>
                    <td>Дім реєстрації</td>
                    <td><?php echo $pp[7]; ?></td>
                </tr>
                <tr>
                    <td>Корпус реєстрації</td>
                    <td><?php echo $pp[8]; ?></td>
                </tr>
                <tr>
                    <td>Квартира реєстрації</td>
                    <td><?php echo $pp[9]; ?></td>
                </tr>
            </table>
        </div>
    </div>
</div>
<div id="vacc" class="modal">
    <div class="modal-content">
        <span class="close" id="close4">&times;</span>
        <div class="modal-table">
            <table class="modal-table">
                <thead>
                    <tr>
                        <th colspan='4'>Картка вакцинувань</th>
                    </tr>
                    <tr>
                        <td>Дата вакцинування</td>
                        <td colspan='3'>Опис</td>
                    </tr>

                </thead>

                <?php
                while ($row = mysqli_fetch_assoc($PatientResult3)) {
                    $chosenVacc .= $row['VaccDate'] . " " . $row['VaccDescr'];
                    echo '<tr><td>' . $row['VaccDate'] . '</td>';
                    echo '<td>' . $row['VaccDescr'] . '</td>';
                    echo '<td><button type="button" class="dellVacc" name="dellVacc"
value="' . $row['Id'] . '" id="dellVacc" class="mybtn" />Видалити</td></tr>';
                }
                ?>
                </tr>
```

```
                </tr>
                <tfoot>
                    <form name="add_vacc" id="add_vacc" method="POST">
                        <div class="row">
                            <tr>
                                <td><input type="date" class="vacc" id="vaccdate"
name="vaccdate" placeholder="Дата вакцинування" /></td>
                                <td><input type="text" class="vacc" id="vaccdescribe"
name="vaccdescribe" placeholder="Опис вакцинації" /></td>
                                <td><button type="button" name="addVacc" id="addVacc"
class="enterbtn">Додати</button></td>
                            </tr>
                        </div>
                    </form>
                </tfoot>
            </table>
        </div>
    </div>
</div>
<div id="predis" class="modal">
    <div class="modal-content">
        <span class="close" id="close5">&times;</span>
        <div class="modal-table">
            <table class="modal-table">
                <thead>
                    <tr>
                        <th colspan='2'>Схильність до хвороб</th>
                    </tr>
                </thead>
                <tbody>
                    <?php
                    $clientQuery4 = "select Predispos.Id as row, Predispos.Predis,
_Predis.Name as p, _Predis.Score, Predispos.Id, Predispos.IdPatient as user from
Predispos join  _Predis on Predispos.Predis=_Predis.Id where IdPatient=" . $chosenId .
";";
                    $PatientResult4 = mysqli_query($con, $clientQuery4);
                    while ($row = mysqli_fetch_assoc($PatientResult4)) {
                        echo '<tr><td>' . $row['p'] . '</td><td><button type="button"
name="dellPredis" class="dellPredis" value="'.$row['row'].'"
class="mybtn"/>Видалити</td></tr>';
                    }
                    ?>
                </tbody>
                <tfoot>
                    <form name="add_redis" id="add_redis" method="POST">
                        <div class="row">
                            <tr>
                                <td><select id="addpredis" name="predis">
                                        <?php

                                        $queryPredis = 'select Id as idaddpredis,Name
from _Predis;';
                                        $allcodes = mysqli_query($con, $queryPredis);
                                        while ($row = mysqli_fetch_assoc($allcodes)) {
                                            echo '<option value="' .
$row["idaddpredis"] . '">' . $row["Name"] . ' </option>';
                                        }
                                        ?>
                                    </select></td>
                                <td><button type="button" name="addPredis"
id="addPredis" class="mybtn">Додати</button></td>
                            </tr>
                        </div>
                    </form>
                </tfoot>
            </table>
        </div>
    </div>
</div>
<script>
    (function update() {
```

```
    $.ajax({
        type: "POST",
        url: "ajax.php",
        data: {
            'action': 'select'
        },
        success: function(server_response) {
            $(".listview").html(server_response);
        }
    }).then(function() {
        setTimeout(update, 10000);
    });
})();
var modal1 = document.getElementById("bio");
var btn1 = document.getElementById("normBio");
var span1 = document.getElementById("close1");
var modal2 = document.getElementById("contacts");
var btn2 = document.getElementById("showConts");
var span2 = document.getElementById("close2");
var modal3 = document.getElementById("pass");
var btn3 = document.getElementById("showPass");
var span3 = document.getElementById("close3");
var modal4 = document.getElementById("vacc");
var btn4 = document.getElementById("showVacc");
var span4 = document.getElementById("close4");
var modal5 = document.getElementById("predis");
var btn5 = document.getElementById("showPredis");
var span5 = document.getElementById("close5");

btn1.onclick = function() {
    modal1.style.display = "block";
}
span1.onclick = function() {
    modal1.style.display = "none";
}
btn2.onclick = function() {
    modal2.style.display = "block";
}
span2.onclick = function() {
    modal2.style.display = "none";
}
btn3.onclick = function() {
    modal3.style.display = "block";
}
span3.onclick = function() {
    modal3.style.display = "none";
}
btn4.onclick = function() {
    modal4.style.display = "block";
}
span4.onclick = function() {
    modal4.style.display = "none";
}
btn5.onclick = function() {
    modal5.style.display = "block";
}
span5.onclick = function() {
    modal5.style.display = "none";
}
window.onclick = function(event) {
    if (event.target == modal1) {
        modal1.style.display = "none";
    }
    if (event.target == modal2) {
        modal2.style.display = "none";
    }
    if (event.target == modal3) {
        modal3.style.display = "none";
    }
    if (event.target == modal4) {
        modal4.style.display = "none";
```

```
    }
    if (event.target == modal5) {
        modal5.style.display = "none";
    }
}
$(document).on('click', '#addVacc', function() {
    var id_row = "<?php echo $chosenId; ?>";
    var date = $('#vaccdate').val();
    var descr = $('#vaccdescribe').val();
    $.ajax({
        type: "POST",
        url: "ajax.php",
        data: {
            'action': 'insertvacc',
            id_row: id_row,
            date: date,
            descr: descr,
        },
        success: function(data) {
            update();
        }
    })
});
$(document).on('click', '.dellVacc', function() {
    var id_r = $(this).val();
    if (confirm("Ви впевнені що хочете видалити запис?")) {
        $.ajax({
            type: "POST",
            url: "ajax.php",
            data: {
                'action': 'dellvacc',
                id_r: id_r,
            },
            dataType: "text",
            success: function(data) {
                update();
            }
        });
    }
});
$(document).on('click', '#addPredis', function() {
    var id_user = "<?php echo $chosenId; ?>";
    var id_predis=$('#addpredis').val();
    $.ajax({
        type: "POST",
        url: "ajax.php",
        data: {
            'action': 'insertpredis',
            id_user: id_user,
            id_predis: id_predis,
        },
        success: function(data) {
            update();
        }
    })
});
$(document).on('click', '.dellPredis', function() {
    var rowpredis = $(this).val();
    if (confirm("Ви впевнені що хочете видалити запис?")) {
        $.ajax({
            type: "POST",
            url: "ajax.php",
            data: {
                'action': 'dellpredis',
                rowpredis: rowpredis,
            },
            dataType: "text",
            success: function(data) {
                alert(data);
                update();
            }
```

```
            });
        }
    });
    $(function() {
        $(document).on('click', '.shbtn', function(e) {
            e.preventDefault();
            $(this).next('.toogle_').toggle();
        });
    });
</script>
```

Г.7 Файл `card.php`

```
<?php
if (!isset($_SESSION['opt']) || $_SESSION['opt'] != 'patientform') {
    header('Location: /');
}
calc_urgency($_SESSION['id']);
//print_r($_POST);
//print_r($_SESSION);
$order_id = "";
$order_id = date('Ymd') . '000000';
$current_date = date('Y-m-d H:i');
$order_id += $_SESSION['id'] * 100;
if (isset($_POST['selectdoc'])) {
    $_SESSION['selectdoc'] = $_POST['selectdoc'];
    $chosenId = $_SESSION['selectdoc'];
}
$clientQuery = 'Select DocId from SimDoc where PatId="' . $_SESSION['id'] . '"';
$clientResult = mysqli_query($con, $clientQuery);
if ($row = mysqli_fetch_assoc($clientResult)) {
    $_SESSION['selectdoc'] = $row['DocId'];
}
$chosenId = $_SESSION['selectdoc'];
$clientQuery2 = "Select * from PrivatDoc where Id=" . $chosenId . ";";
$clientResult2 = mysqli_query($con, $clientQuery2);
while ($row = mysqli_fetch_assoc($clientResult2)) {
    $chosenContacts = $row['Phone1'] . "|" . $row['Phone2'] . "|" . $row['Viber'] .
"|" . $row['Telegram'] . "|" . $row['Email'];
}
$clientQuery3 = "Select * from Doctor where Id=" . $chosenId . ";";
$clientResult3 = mysqli_query($con, $clientQuery3);
while ($row = mysqli_fetch_assoc($clientResult3)) {
    $chosenPhoto = $row['Photo'];
    $chosenName = $row['LFN'];
    $chosenLocation = $row['RecRegion'] . "|" . $row['RecCitySett'] . "|" .
$row['RecDepart'] . "|" . $row['RecStreet'] . "|" . $row['RecHouse'] . "|" .
$row['RecCorps'] . "|" . $row['RecCab'];
}
?>
<div>
    <h1>Кабінет пацієнта</h1>
    <div>
        <div>
            <?php
            if (!isset($_SESSION['selectdoc'])) {
                echo '<h3>У вас не вибрано сімейного лікаря, будь ласка оберіть з
списку нижче</h3>';
                echo '<div style="display: inline-block;text-align: center;width:
100%;">';
                $clientQuery2 = "Select * from Doctor";
                $clientResult2 = mysqli_query($con, $clientQuery2);
                while ($row = mysqli_fetch_assoc($clientResult2)) {
                    echo '<div style="display:inline-block"><button name="setsim"
class="setsim" value=' . $row['Id'] . '><span></span><span><table><tr><td
rowspan="4"><img style="width:80px;" src=' . $row["Photo"] . '></td><td colspan="4">'
. $row['LFN'] . '</td></tr>
```

```
        <tr><td colspan="2">' . $row['RecRegion'] . ' область</td><td colspan="2">' .
$row['RecCitySett'] . '</td></tr>
        <tr><td colspan="4">Клініка "' . $row['RecDepart'] . '"</td></tr>
        <tr><td>вулиця ' . $row['RecStreet'] . '</td><td>будинок ' . $row['RecHouse']
. '</td><td> корпус ' . $row['RecCorps'] . '</td><td> кабінет ' . $row['RecCab'] .
'</td></tr></table></span></button>';
                }
                echo    '</div></div>';
            } ?>
        </div>
        <!-- Таблиця-->
        <div id="div3">
            <div style="width:20%">
                <div id="div1">
                    <h3>Сімейний лікар</h3>
                    <img style="width:80px" src="<?php echo $chosenPhoto ?>" /><br>
                    <p><?php echo $chosenName  ?></p>
                    <a id="doccont1" class="btn shbtn" onclick="">Контактні дані
лікаря</a>
                    <div id="doccont" class="toogle_form" style="display:none;margin-
top:25px;">
                        <table class="modal-table">
                            <thead>
                                <th colspan='2'>Контактні дані лікаря</th>
                            </thead>
                            <?php $cont = explode("|", $chosenContacts); ?>
                            <tr>
                                <td>Телефон 1</td>
                                <td><?php echo $cont[0]; ?></td>
                            </tr>
                            <tr>
                                <td>Телефон 2</td>
                                <td><?php echo $cont[1]; ?></td>
                            </tr>
                            <tr>
                                <td>Viber</td>
                                <td><?php echo $cont[2]; ?></td>
                            </tr>
                            <tr>
                                <td>Telegram</td>
                                <td><?php echo $cont[3]; ?></td>
                            </tr>
                            <tr>
                                <td>Email</td>
                                <td><?php echo $cont[4]; ?></td>
                            </tr>
                        </table>
                    </div><br><br>
                    <a id="docloc1" class="btn shbtn" onclick="">Міце прийому
лікаря</a>
                    <div id="docloc" class="toogle_form" style="display:none; margin-
top:25px;">
                        <table class="modal-table">
                            <thead>
                                <th colspan='2'>Місце прийому лікаря</th>
                            </thead>
                            <?php $loca = explode("|", $chosenLocation); ?>
                            <tr>
                                <td>Регіон, область</td>
                                <td><?php echo $loca[0]; ?></td>
                            </tr>
                            <tr>
                                <td>Місто</td>
                                <td><?php echo $loca[1]; ?></td>
                            </tr>
                            <tr>
                                <td>Клініка</td>
                                <td><?php echo $loca[2]; ?></td>
                            </tr>
                            <tr>
                                <td>Вулиця</td>
```

```
                                <td><?php echo $loca[3]; ?></td>
                            </tr>
                            <tr>
                                <td>Будинок</td>
                                <td><?php echo $loca[4]; ?></td>
                            </tr>
                            <tr>
                                <td>Корпус</td>
                                <td><?php echo $loca[5]; ?></td>
                            </tr>
                            <tr>
                                <td>Кабінет</td>
                                <td><?php echo $loca[6]; ?></td>
                            </tr>
                        </table>
                    </div>
                </div>
            </div>
            <div style="width:80%">
                <div id="div2">
                    <br>
                    <br>
                    <a href="javascript:void(0);" id="bnewreq1" class="btn shbtn"
onclick="">Відкрити/згорнути форму для подачі нового запиту</a>
                    <a href="javascript:void(0);" class="btn shbtn" onclick="">Архівні
звернення</a>
                    <div id="newreq" class="toogle_form " style="display:none">
                        <h2>Надіслати звернення</h2>
                        <form action="" method="POST">
                            <div style="float:left;width:min-content">
                                <div class="input-group">
                                    <table>
                                        <tr>
                                            <td><label
for="creationData">Дата</label></td>
                                            <td><label for name="CurrDiaT">Поточний
діас. тиск</label></td>
                                        </tr>
                                        <tr>
                                            <td><input name="creationData"
value="<?php echo date('Y-m-d H:i:s'); ?>"></td>
                                            <td><input name="CurrDiaT" type="text"
placeholder="XXX"></td>
                                        </tr>
                                        <tr>
                                            <td></td>
                                            <td><label for="CurrSisT">Поточний сист.
тиск</label></td>
                                        </tr>
                                        <tr>
                                            <td></td>
                                            <td><input name="CurrSisT" type="text"
placeholder="XXX"></td>
                                        </tr>
                                        <tr>
                                            <td><label for="cardNum">Номер
звернення</label></td>
                                            <td><label for="CurrTemp">Поточна
температура</label></td>
                                        </tr>
                                        <tr>
                                            <td><input name="cardNum" value="<?php
echo $order_id; ?>"></td>
                                            <td><input name="CurrTemp" type="text"
placeholder="XX,X">
                                            </td>
                                        </tr>
                                    </table>
                                </div>
                            </div>
                            <div style="float:left;width:max-content;">
```

```
                                <div class="input-group">
                                    <table id=symps>
                                        <thead>
                                            <tr>
                                                <th colspan='2'>
                                                    <label
for="symps">Симптоми</label>
                                                </th>
                                            </tr>
                                        </thead>
                                        <tbody>
                                            <div>
                                                <div id="field_wrapper">
                                                    <tr>
                                                        <td>
<select id="addsymps" name="symps[]">
<?php
$querySymp = 'select Id,Name from _Symp;';
$allcodes = mysqli_query($con, $querySymp);
while ($row = mysqli_fetch_assoc($allcodes)) {
echo '<option value="' . $row["Id"] . '">' . $row["Name"] . ' </option>';
}
?>
</select>
                                                        </td>
                                                    </tr>
                                                </div>
                                            </div>
                                        </tbody>
                                        <tfoot>
                                            <tr>
                                                <td>
                                                    <a class="btn"
href="javascript:void(0);" id="addSymp" name="addSymp">Додати симтом</a>
                                                </td>
                                            </tr>
                                        </tfoot>
                                    </table>
                                </div>
                            </div>
                            <div style="float:left;width:min-content">
                                <table>
                                    <thead>
                                        <tr>
                                            <th>Додатковий коментар</th>
                                        </tr>
                                    </thead>
                                    <tbody
style="display:block;width:400px;height:150px;">
                                        <tr>
                                            <td style="width:100%; height:100%">
                                                <textarea id="patcoment"
name="patcoment" rows="10" cols="45" style="width: 380px; height: 130px;"></textarea>
                                            </td>
                                        </tr>
                                    </tbody>
                                    <tfoot>
                                        <tr>
                                            <td><input type="Submit" class="btn"
value="Надіслати запит"></td>
                                        </tr>
                                    </tfoot>
                                </table>
                            </div>
                        </form>
                    </div><br><br>
                    <br>
                    <div class="toogle_form" style="display:block">
                        <?php
                        //$sqlHistory = "select
h.StartTime,h.Numb,s.CurrSisT,s.CurrDiaT,s.CurrTemp, s.ApdxText, n.Name from History h
```

```
join Session s on h.Id=s.IdHistory join Symptom m on m.IdSession=s.Id join _Symp n on
m.Symp=n.Id where h.IdPatient=" . $_SESSION['id'] . ";";
                        $sqlHistory = "select * from History join Session on
History.Id=Session.IdHistory where History.IdPatient=" . $_SESSION['id'] . ";";
                        //print_r($sqlHistory);
                        $resHistory = mysqli_query($con, $sqlHistory);
                        while ($row = mysqli_fetch_assoc($resHistory)) {
                            $sqlSymps = "select a.Name from _Symp a join Symptom b on
b.Symp=a.Id join Session c on c.Id=b.IdSession  join History d on d.Id=c.IdHistory
where d.Numb=" . $row['Numb'] . ';';
                            $resSymps = mysqli_query($con, $sqlSymps);
                            $symp = '';
                            while ($row2 = mysqli_fetch_assoc($resSymps)) {
                                $symp .= ($row2['Name']) . '<br>';
                            }
                            $flag = ($row['OpenFlag'] == "0") ? "Відкрито" :
"Закрито";
                            echo '<button class="shbtn" onclick="">Номер запиту:' .
$row['Numb'] . '                    Статус запиту: ' . $flag . ' </button>';
                            echo '<div class="toogle_ " style="display:none">
        <button><table>
            <tr>
                    <td>Номер запиту</td>
                    <td>' . $row['Numb'] . '</td>
                    <td>Статус запиту</td>
                    <td>' . $flag . '</td>
                </tr>
                <tr>
                    <td>Поточний діастолічний тиск</td>
                    <td>Поточний систолічний тиск</td>
                    <td>Поточна температура</td>
                    <td>Перелік симтомів</td>
                </tr>
                <tr>
                    <td>' . $row['CurrDiaT'] . '</td>
                    <td>' . $row['CurrSisT'] . '</td>
                    <td>' . $row['CurrTemp'] . '</td>
                    <td>' . $symp . '</td>
                </tr>
                <tr>
                    <td colspan="4">Коментар до запиту</td>
                </tr>
                <tr>
                    <td colspan="4">' . $row['ApdxText'] . '</td>
                </tr>
                <tr>
                    <td colspan="4">Назначення лікаря</td>
                </tr>
                <tr>
                    <td colspan="4">' . $row['DocRecom'] . '</td>
                </tr>
            </table></button>
        </div>';
                        }
                        ?>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <?php
    $sql_in_hist = 'INSERT INTO History VALUES(default,"' . $_SESSION['id'] . '","' .
$chosenId . '","' . $_POST['cardNum'] . '","' . $_POST['creationData'] .
'",default,default,default,default,default);';
    mysqli_query($con, $sql_in_hist);
    $sql_from_hist = 'SELECT Id from History where Numb="' . $_POST['cardNum'] . '";';
    $result = mysqli_query($con, $sql_from_hist);
    $result1 = mysqli_fetch_assoc($result);
    $hist_id = $result1['Id'];
```

```
    $sql_in_sess = 'INSERT INTO Session VALUES(default,"' . $hist_id . '","' .
$_POST['creationData'] . '","' . $_POST['CurrTemp'] . '","' . $_POST['CurrSisT'] .
'","' . $_POST['CurrDiaT'] . '","' . $_POST['patcoment'] . '",default);';
    mysqli_query($con, $sql_in_sess);
    $sql_from_sess = 'SELECT Id from Session where IdHistory="' . $hist_id . '";';
    $result = mysqli_query($con, $sql_from_sess);
    $result1 = mysqli_fetch_assoc($result);
    $sess_id = $result1['Id'];
    $symps_array = $_POST['symps'];
    foreach ($symps_array as $k) {
        $sql_to_symp = 'Insert into Symptom Values(default,"' . $sess_id . '","' . $k
. '");';
        mysqli_query($con, $sql_to_symp);
    }
    unset($k);
    ?>
    <script>
        $(function() {
        $(document).on('click', '.shbtn', function(e) {
            e.preventDefault();
            $(this).next('.toogle_').toggle();
        });
    });
        $(document).on('click', '.setsim', function() {
            let selectdoc = $(this).val();
            let pat = <?php echo $_SESSION['id']; ?>;
            if (confirm("Ви впевнені, що хочете саме цього лікаря?")) {
                $.ajax({
                    type: "POST",
                    url: "ajax.php",
                    data: {
                        'action': 'setsim',
                        selectdoc: selectdoc,
                        pat: pat,
                    },
                    dataType: "text",
                    success: function(data) {
                        alert(data);
                        location.reload();
                        return false;
                    }
                });
            }
        });
        $(document).ready(function(e) {
            var html = '<tr><td><select id="addsympschild" name="symps[]">
<?php $querySymp = "select Id,Name from _Symp;";
$allcodes = mysqli_query($con, $querySymp);
while ($row = mysqli_fetch_assoc($allcodes)) {
echo "<option value=\"" . $row["Id"] . "\">" . $row["Name"] . "</option>";
} ?>
</select></td><td><a href="javascript:void(0);" class="btn" id="delSymp">
Видалити</a></td></tr>';
            $('#addSymp').click(function(e) {
                $("#symps tbody").append(html);
            });
            $("#field_wrapper").on('click', '#delSymp', function(e) {
                $(this).parent('tr').remove();
            });
        });
        $(function() {
            $(document).on('click', '.shbtn', function(e) {
                e.preventDefault();
                $(this).next('.toogle_form').toggle();
            });
        });
    </script>;
```

Г.8 Файл admin.php

```
<?php
include 'connect.php';

if ((!isset($_SESSION['use'])) && $_SESSION['opt'] != 'adminform') {
    header('Location: login.php');
}
$queryPatients = "Select Id,LFN from Patient;";
$PatientResTable = mysqli_query($con, $queryPatients);
$queryDocs= "Select d.Id,d.LFN,ad.IsActivated as activated from Doctor d inner join
ActiveDoc ad on d.Id=ad.DocId;";
$DocsResTable = mysqli_query($con, $queryDocs);

?>
<div class="patientsTable">
<h4>Таблиця пацієнтів</h4>
    <table style="border:solid 2px grey;">
        <thead>
            <th>ID</th><th>Пацієнт</th><th>Керування</th>
        </thead>
        <tbody>
            <?php

            while ($r = mysqli_fetch_assoc($PatientResTable)) {
                $output = '<tr id="patient-'. $r["Id"] . '"><td>' .$r["Id"].
'</td><td>' .$r["LFN"]. '</td>';
                $output .='<td><button name="delPat" class="delPat"
value="'.$r['Id'].'">Видалити пацієнта</button></td></tr>';
                echo $output;


            }
            ?>
        </tbody>
    </table>
</div>
<div class="docsTable">
    <h4>Таблиця лікарів</h4>
    <table >
        <thead>
            <th>ID</th><th>Лікар</th><th>Статус аккаунту</th><th
colspan="2">Керування</th>
        </thead>
        <tbody>
            <?php

            while ($row = mysqli_fetch_assoc($DocsResTable)) {
                $output = '<tr id="doc-'. $row["Id"] . '"><td>' .$row["Id"].
'</td><td>' .$row["LFN"]. '</td><td>';
                if($row["activated"]=='1'){ $output.= "Активний"; } else {$output .=
"Заблоковано";}
                $output.='</td><td><button name="changeDoc" class="changeDoc"
value="'.$row['Id'].'">Змінити статус</button></td>';
                $output.='</td><td><button name="delDoc" class="delDoc"
value="'.$row['Id'].'">Видалити лікаря</button></td></tr>';
                echo $output;

            }
            ?>
        </tbody>
    </table>
</div>
<script>
    $(document).on('click', '.delDoc', function() {
        var doctodel = $(this).val();
        if (confirm("Ви впевнені, що хочете видалити запис?")) {
            $.ajax({
```

```
                type: "POST",
                url: "ajax.php",
                data: {
                    'action': 'delete_doc',
                    doctodel: doctodel,
                },
                dataType: "text",
                success: function(data) {
                    alert(data, doctodel);
                }
            });
        }
    });
    $(document).on('click', '.changeDoc', function() {
        var docactive = $(this).val();
        if (confirm("Ви впевнені, що хочете змінити запис?")) {
            $.ajax({
                type: "POST",
                url: "ajax.php",
                data: {
                    'action': 'change_activ',
                    docactive: docactive,
                },
                dataType: "text",
                success: function(data) {
                    alert(data, docactive);
                }
            });
        }
    });
    $(document).on('click', '.delPat', function() {
        var pattodel = $(this).val();
        if (confirm("Ви впевнені, що хочете видалити запис?")) {
            $.ajax({
                type: "POST",
                url: "ajax.php",
                data: {
                    'action': 'delete_pat',
                    pattodel: pattodel,
                },
                dataType: "text",
                success: function(data) {
                    alert(data, pattodel);
                }
            });
        }
    });
</script>
```

Г.9 Файл `ajax.php`

```
<?php
require 'connect.php';
if (isset($_POST['action'])) {
    switch ($_POST['action']) {
        case 'insertvacc':
            insertvacc($con);
            break;
        case 'select':
            select($con);
            break;
        case 'dellvacc':
            dellvacc($con);
            break;
        case 'dellpredis':
            dellpredis($con);
            break;
```

```
        case 'insertpredis':
            insertpredis($con);
            break;
        case 'change_activ':
            change_activ($con);
            break;
        case 'delete_pat':
            delete_pat($con);
            break;
        case 'delete_doc':
            delete_doc($con);
            break;
        case 'setsim':
            setsim($con);
            break;
        case 'auto_calc_urgency':
            auto_calc_urgency($con);
            break;
    }
}
function select($con) {
    auto_calc_urgency($con);
    $sql = "SELECT * FROM Patient order by UrgentPow Desc";
    $result = mysqli_query($con, $sql);
    if (mysqli_num_rows($result) > 0) {
        echo '<form action="" method="POST">';
        while ($row = mysqli_fetch_assoc($result)) {
            echo  '<button name="select" type="submit"  value="' . $row["Id"] .
'"><tr><img style="width:80px" src="' . $row["Photo"] . '"/></tr><tr>' . $row["LFN"] .
'</tr></button><br>';
        }
        echo '</form>';
    }
}
function insertvacc($con)
{
    $vacc_date = $_POST['date'];
    $vacc_descr = $_POST['descr'];
    $vaccId = $_POST['id_row'];
    $ql = "INSERT INTO VaccCard VALUES(default,\"" . $vaccId . "\",\"" . $vacc_date .
"\",\"" . $vacc_descr . "\")";
    if (mysqli_query($con, $ql)) {
        echo 'Запис внесено';
    } else {
        die("sql " . $ql . ' vd' .  $_POST['vaccdate'] . ' d ' . $_POST['date']);
    }
}
function dellvacc($con)
{
    $sql = "DELETE FROM VaccCard WHERE id=\"" . $_POST["id_r"] . "\"";
    if (mysqli_query($con, $sql)) {
        echo 'Запис видалено ' . $sql . ' ';
    } else {
        die($sql);
    }
}
function insertpredis($con)
{
    $predis_id = $_POST['id_predis'];
    $predisPatId = $_POST['id_user'];
    $ql = "INSERT INTO Predispos VALUES(default,\"" . $predisPatId . "\",\"" .
$predis_id . "\")";
    if (mysqli_query($con, $ql)) {
        echo 'Запис внесено';
    } else {
        die("sql " . $ql);
    }
}
function dellpredis($con)
{
    $sql = "DELETE FROM Predispos WHERE Id=\"" . $_POST["rowpredis"] . "\"";
```

```
    if (mysqli_query($con, $sql)) {
        echo 'Запис видалено ' . $sql . ' ' . $_POST['rowpredis'];
    } else {
        die($sql . ' ' . $_POST['rowpredis']);
    }
}
function change_activ($con)
{
    $act = $_POST['docactive'];
    $sread = "select IsActivated from ActiveDoc where DocId=\"" . $act . "\";";
    $readresult = mysqli_query($con, $sread);
    if ($readresult) {
        $act_check = mysqli_fetch_assoc($readresult);
        if ($act_check['IsActivated'] == '0') {
            $acts = '1';
        } else {
            $acts = '0';
        }
        $swrite = "UPDATE ActiveDoc SET IsActivated = '" . $acts . "' where DocId='" .
$act . "';";
        mysqli_query($con, $swrite);
    } else {
        die($sread . " " . $readresult);
    }
}
function delete_pat($con)
{
    $sql = "DELETE from VaccCard where IdPatient = \"" . $_POST['pattodel'] .
"\";DELETE from Patient where Id = \"" . $_POST['pattodel'] . "\";DELETE from History
where IdPatient = \"" . $_POST['pattodel'] . "\";DELETE from PrivatPat where Id = \""
. $_POST['pattodel'] . "\";DELETE from Predispos where IdPatient = \"" .
$_POST['pattodel'] . "\";";
    print_r($sql);
    if (mysqli_query($con, $sql)) {
        echo "Записи видалені";
    } else {
        die("не вийшло" . $sql);
    }
}
function delete_doc($con)
{
    $sql = "DELETE from History where IdDoctor = \"" . $_POST['doctodel'] . "\";";
    $sql .= "DELETE from Doctor where Id = \"" . $_POST['doctodel'] . "\";";
    $sql .= "DELETE from History where IdDoctor = \"" . $_POST['doctodel'] . "\";";
    $sql .= "DELETE from PrivatDoc where Id = \"" . $_POST['doctodel'] . "\";";
    $sql .= "DELETE from RecChart where IdDoctor = \"" . $_POST['doctodel'] . "\";";
    if (mysqli_query($con, $sql)) {
        echo "Записи видалені";
    } else {
        die($sql);
    }
}
function setsim($con)
{
    $doc_id = $_POST['selectdoc'];
    $pat_id = $_POST['pat'];
    $ql = "INSERT INTO SimDoc VALUES(default,\"" . $doc_id . "\",\"" . $pat_id .
"\")";
    if (mysqli_query($con, $ql)) {
        echo 'Запис внесено';
    } else {
        die("sql " . $ql);
    }
}
function auto_calc_urgency($con){
    $sql="Select Id from Patient";
    $result=mysqli_query($con,$sql);
    while($row=mysqli_fetch_assoc($result)){
        calc_urgency($row['Id']);
    }
}
```

```
function calc_urgency($patientid)
{
    global $con;
    $y_Sql = "select floor(datediff(curdate(),BirthDate) / 365) from Patient where
Id=" . $patientid . ";";
    $Y = mysqli_fetch_row(mysqli_query($con, $y_Sql));
    $sql1 = 'Select * from Patient where Id="' . $patientid . '";';
    $query1 = mysqli_query($con, $sql1);
    while ($row = mysqli_fetch_assoc($query1)) {
        $NT = $row['NormTemp'];
        $NDT = $row['NormDiaT'];
        $NST = $row['NormSistT'];
    }
    $sql2 = 'select h.Id , s.CurrTemp as CT,s.CurrDiaT as CDT, s.CurrSisT as CST, s.Id
as cIS from History h join Session s on h.Id=s.IdHistory where OpenFlag=0 and
IdPatient="' . $patientid . '";';
    $query2 = mysqli_query($con, $sql2);
    while ($row = mysqli_fetch_assoc($query2)) {
        $CT = $row['CT'];
        $CDT = $row['CDT'];
        $CST = $row['CST'];
        $IS = $row['cIS'];
        //$check = $row;
    }
    $sql3 = 'Select SUM(Score) as sum_score from Predispos join _Predis on
_Predis.Id=Predispos.Predis where IdPatient=' . $patientid . ';';
    $query3 = mysqli_query($con, $sql3);
    while ($row = mysqli_fetch_assoc($query3)) {
        $SPS = $row['sum_score'];
    }
    $sql4 = 'Select Symp,Score from Symptom join _Symp on _Symp.Id=Symptom.Symp where
Symptom.IdSession=' . $IS . ';';
    $query4 = mysqli_query($con, $sql4);
    while ($row = mysqli_fetch_assoc($query4)) {
        $symps[] = $row['Symp'];
        $ss[] = $row['Score'];
    }
    $sql5 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 1 and 6';
    $query5 = mysqli_query($con, $sql5);
    $Sa = $row = mysqli_fetch_row($query5);
    if ($row[0] == NULL) {
        $Sa = 0;
    } else {
        $Sa = $row[0];
    }
    $sql6 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 7 and 10';
    $query6 = mysqli_query($con, $sql6);
    $Sb = $row = mysqli_fetch_row($query6);
    if ($row[0] == NULL) {
        $Sb = 0;
    } else {
        $Sb = $row[0];
    }
    $sql7 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 11 and 13';
    $query7 = mysqli_query($con, $sql7);
    $Sc = $row = mysqli_fetch_row($query7);
    if ($row[0] == NULL) {
        $Sc = 0;
    } else {
        $Sc = $row[0];
    }
    $sql8 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 14 and 17';
    $query8 = mysqli_query($con, $sql8);
```

```
    $Sd = $row = mysqli_fetch_row($query8);
    if ($row[0] == NULL) {
        $Sd = 0;
    } else {
        $Sd = $row[0];
    }
    $Ky = pow(1.05, $Y[0]);
    $dt = $CT - $NT;
    if ($dt >= 2.6) {
        $Kbi = 4;
    } elseif ($dt >= 2.0) {
        $Kbi = 3;
    } elseif ($dt >= 1.4) {
        $Kbi = 2;
    } elseif ($dt >= 0.8) {
        $Kbi = 1;
    } else {
        $Kbi = 0;
    }
    if (($CST - $NST) >= 15.0) {
        $Kbi += 1.5;
    };
    if (($CDT - $NDT) >= 10.0) {
        $Kbi += 2.0;
    };
    $Sacheck = array("1", "2", "3");

//Для коефіцієнта множення Ka
    $koef_array_1_check = array_intersect($symps, $Sacheck);

//Отримуємо перетин массивів перевірки та реальних даних
    if (count(array_diff($Sacheck, $koef_array_1_check)) == 0) {
        $Ka = 2.2;
    } else {
        $Ka = 1.0;
    }; //встановлення коєфіцієнту
    $Sbcheck = array("7", "8", "9"); //Для коефіцієнта множення Kb
    $koef_array_2_check = array_intersect($symps, $Sbcheck);
    if (count(array_diff($Sacheck, $koef_array_2_check)) == 0) {
        $Kb = 1.7;
    } else {
        $Kb = 1.0;
    };
    $Sccheck = array("11", "12", "13"); //Для коефіцієнта множення Kc
    $koef_array_3_check = array_intersect($symps, $Sccheck);
    if (count(array_diff($Sacheck, $koef_array_3_check)) == 0) {
        $Kc = 1.4;
    } else {
        $Kc = 1.0;
    };
    $Sdcheck = array("14", "15", "16", "17"); //Для коефіцієнта множення Kd
    $koef_array_4_check = array_intersect($symps, $Sdcheck);
    if (count(array_diff($Sdcheck, $koef_array_4_check)) == 0) {
        $K = 1.2;
    } else {
        $Kd = 1.0;
    };
    $K_urgent = $Ky + $Kbi + $SPS + $Ka * $Sa + $Kb * $Sb + $Kc * $Sc + $Kd * $Sd;
    $sql9 = 'UPDATE Patient Set UrgentPow="' . $K_urgent . '" where Id="' . $patientid
. '";';
    mysqli_query($con, $sql9);
})
```

Г.10 Файл process.php

```
<?php
require('connect.php');
function make_password($password)
{
    $hash_coding = "pbkdf2_sha512";
    $repeats = 10000;
    $createSalt = random_bytes(16);
    $createSalt = base64_encode($createSalt);
    $hash = hash_pbkdf2("SHA512", $password, $createSalt, $repeats, 0, true);
    $toDBStr = $hash_coding . "$" . $repeats . "$" . $createSalt . "$" .
base64_encode($hash);
    // This string is to be saved into DB.
    //    echo $toDBStr;
    return $toDBStr;
}
function verify_Password($dbString, $password)
{
    $chunks = explode('$', "$dbString");
    $repeats = $chunks[1];
    $salt = $chunks[2];
    $old_hash = $chunks[3];
    $string_hash = hash_pbkdf2("SHA512", $password, $salt, $repeats, 0, true);
    $string_hash = base64_encode($string_hash);
    if ($string_hash == $old_hash) {
        // login ok.
        return true;
    } else {
        //login fail
        return false;
    }
}
function getPII($id)
{
    //here php get from DB
    //PII from database
}
function sendMessageDP($id)
{
    //here php insert info from site to db
    //Doc sends message to patient
}
function sendMessagePD($id)
{
    //here php insert info from site to db
    //Patient sends message to Doc
}
function calc_urgency($patientid)
{
    global $con;
    $y_Sql = "select floor(datediff(curdate(),BirthDate) / 365) from Patient where
Id=" . $patientid . ";";
    $Y = mysqli_fetch_row(mysqli_query($con, $y_Sql));
    $sql1 = 'Select * from Patient where Id="' . $patientid . '";';
    $query1 = mysqli_query($con, $sql1);
    while ($row = mysqli_fetch_assoc($query1)) {
        $NT = $row['NormTemp'];
        $NDT = $row['NormDiaT'];
        $NST = $row['NormSistT'];
    }
    $sql2 = 'select h.Id , s.CurrTemp as CT,s.CurrDiaT as CDT, s.CurrSisT as CST, s.Id
as cIS from History h join Session s on h.Id=s.IdHistory where OpenFlag=0 and
IdPatient="' . $patientid . '";';
    $query2 = mysqli_query($con, $sql2);
    while ($row = mysqli_fetch_assoc($query2)) {
        $CT = $row['CT'];
        $CDT = $row['CDT'];
```

```
        $CST = $row['CST'];
        $IS = $row['cIS'];
        //$check = $row;
    }
    $sql3 = 'Select SUM(Score) as sum_score from Predispos join _Predis on
_Predis.Id=Predispos.Predis where IdPatient=' . $patientid . ';';
    $query3 = mysqli_query($con, $sql3);
    while ($row = mysqli_fetch_assoc($query3)) {
        $SPS = $row['sum_score'];
    }
    $sql4 = 'Select Symp,Score from Symptom join _Symp on _Symp.Id=Symptom.Symp where
Symptom.IdSession=' . $IS . ';';
    $query4 = mysqli_query($con, $sql4);
    while ($row = mysqli_fetch_assoc($query4)) {
        $symps[] = $row['Symp'];
        $ss[] = $row['Score'];
    }
    $sql5 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 1 and 6';
    $query5 = mysqli_query($con, $sql5);
    $Sa = $row = mysqli_fetch_row($query5);
    if ($row[0] == NULL) {
        $Sa = 0;
    } else {
        $Sa = $row[0];
    }
    $sql6 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 7 and 10';
    $query6 = mysqli_query($con, $sql6);
    $Sb = $row = mysqli_fetch_row($query6);
    if ($row[0] == NULL) {
        $Sb = 0;
    } else {
        $Sb = $row[0];
    }
    $sql7 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 11 and 13';
    $query7 = mysqli_query($con, $sql7);
    $Sc = $row = mysqli_fetch_row($query7);
    if ($row[0] == NULL) {
        $Sc = 0;
    } else {
        $Sc = $row[0];
    }
    $sql8 = 'select SUM(a.Score) as custom_sum from (Select Symp,Score from Symptom
join _Symp on _Symp.Id=Symptom.Symp where Symptom.IdSession=' . $IS . ') a where
a.Symp between 14 and 17';
    $query8 = mysqli_query($con, $sql8);
    $Sd = $row = mysqli_fetch_row($query8);
    if ($row[0] == NULL) {
        $Sd = 0;
    } else {
        $Sd = $row[0];
    }
    $Ky = pow(1.05, $Y[0]);
    $dt = $CT - $NT;
    if ($dt >= 2.6) {
        $Kbi = 4;
    } elseif ($dt >= 2.0) {
        $Kbi = 3;
    } elseif ($dt >= 1.4) {
        $Kbi = 2;
    } elseif ($dt >= 0.8) {
        $Kbi = 1;
    } else {
        $Kbi = 0;
    }
    if (($CST - $NST) >= 15.0) {
```

```
        $Kbi += 1.5;
    };
    if (($CDT - $NDT) >= 10.0) {
        $Kbi += 2.0;
    };
    $Sacheck = array("1", "2", "3");
    $koef_array_1_check = array_intersect($symps, $Sacheck);
    if (count(array_diff($Sacheck, $koef_array_1_check)) == 0) {
        $Ka = 2.2;
    } else {
        $Ka = 1.0;
    }; //встановлення коєфіцієнту
    $Sbcheck = array("7", "8", "9"); //Для коефіцієнта множення Kb
    $koef_array_2_check = array_intersect($symps, $Sbcheck);
    if (count(array_diff($Sacheck, $koef_array_2_check)) == 0) {
        $Kb = 1.7;
    } else {
        $Kb = 1.0;
    };
    $Sccheck = array("11", "12", "13"); //Для коефіцієнта множення Kc
    $koef_array_3_check = array_intersect($symps, $Sccheck);
    if (count(array_diff($Sacheck, $koef_array_3_check)) == 0) {
        $Kc = 1.4;
    } else {
        $Kc = 1.0;
    };
    $Sdcheck = array("14", "15", "16", "17"); //Для коефіцієнта множення Kd
    $koef_array_4_check = array_intersect($symps, $Sdcheck);
    if (count(array_diff($Sdcheck, $koef_array_4_check)) == 0) {
        $K = 1.2;
    } else {
        $Kd = 1.0;
    };
    $K_urgent = $Ky + $Kbi + $SPS + $Ka * $Sa + $Kb * $Sb + $Kc * $Sc + $Kd * $Sd;
    $sql9 = 'UPDATE Patient Set UrgentPow="' . $K_urgent . '" where Id="' . $patientid
. '";';
    mysqli_query($con, $sql9);
}
```

Г.11 Файл `logout.php`

```
<?php

   session_destroy();
   echo "<h1 style=\"color:light-blue\">Ваш сеанс завершено, переадресую на початкову
сторінку </h1>";
   //header('Location: ' . $_SERVER['HTTP_REFERER']);
   header('Refresh: 1.5; URL=/');
?>
<script>
sessionStorage.clear();
</script>
```

Г.12 Файл `reg.css`

```
    *,
    *:before,
    *:after {
        box-sizing: border-box;
        text-align: center;
    }
    body {
        padding: 2em;
```

```
    font-family: 'Open Sans', 'Helvetica Neue', Helvetica, Arial, sans-serif;
    font-size: 15px;
    color: #b9b9b9;
    background-color: #e3e3e3;
}
h4 {
    color: rgb(13, 225, 36);
}
input[type=submit] {
    font-family: sans-serif;
    outline: 0;
    background: rgb(13, 225, 36);
    width: 100%;
    border: 0;
    padding: 15px;
    color: #FFffFF;
    font-size: 14px;
    -webkit-transition: all 0.3 ease;
    transition: all 0.3 ease;
    cursor: pointer;
}
input,
input[type="radio"]+label,
input[type="checkbox"]+label:before,
select option,
select {
    color: gray;
    text-align: center;
    text-align-last: center;
    width: 100%;
    padding: 1em;
    line-height: 1.4;
    background-color: #f9f9f9;
    border: 1px solid #e5e5e5;
    border-radius: 3px;
    -webkit-transition: 0.35s ease-in-out;
    -moz-transition: 0.35s ease-in-out;
    -o-transition: 0.35s ease-in-out;
    transition: 0.35s ease-in-out;
    transition: all 0.35s ease-in-out;
}
select {
    font-family: 'Font Awesome', sans-serif;
}
input:focus {
    outline: 0;
    border-color: #64ac15;
}
input:focus+.input-icon i {
    color: #7ed321;
}
input:focus+.input-icon:after {
    border-right-color: #7ed321;
}
.input-group {
    margin-bottom: 1em;
    zoom: 1;
}
.input-group:before,
.input-group:after {
    content: "";
    display: table;
}
.input-group:after {
    clear: both;
}
.input-group-icon {
    position: relative;
}
.input-group-icon input {
    padding-left: 3.4em;
```

```
}
.input-group-icon .input-2-3 input {
    padding-left: 0.6em;
}
.input-group-icon .input-icon {
    position: absolute;
    top: 17;
    left: 0;
    width: 3.4em;
    height: 3.4em;
    line-height: 3.4em;
    text-align: center;
    pointer-events: none;
}
.input-group-icon .input-icon:after {
    position: absolute;
    top: 0;
    bottom: 0.6em;
    left: 3.4em;
    display: inline-block;
    content: "";
    -webkit-transition: 0.35s ease-in-out;
    -moz-transition: 0.35s ease-in-out;
    -o-transition: 0.35s ease-in-out;
    transition: 0.35s ease-in-out;
    transition: all 0.35s ease-in-out;
}
.input-group-icon .input-icon i {
    -webkit-transition: 0.35s ease-in-out;
    -moz-transition: 0.35s ease-in-out;
    -o-transition: 0.35s ease-in-out;
    transition: 0.35s ease-in-out;
    transition: all 0.35s ease-in-out;
}
.container {
    max-width: 38em;
    padding: 1em 3em 2em 3em;
    margin: 0em auto;
    background-color: #fff;
    border-radius: 4.2px;
    box-shadow: 0px 3px 10px -2px rgba(0, 0, 0, 0.2);
}
.row {
    zoom: 1;
}
.row:before,
.row:after {
    content: "";
    display: table;
}
.row:after {
    clear: both;
}
.col-half {
    padding-right: 10px;
    float: left;
    width: 50%;
}
.col-half:last-of-type {
    padding-right: 0;
}
.col-third {
    padding-right: 0;
    margin-right: 2%;
    float: left;
    width: 32%;
}
.col-third:last-of-type {
    padding-right: 0;
    margin-right: 0;
}
```

```
    @media only screen and (max-width: 540px) {
        .col-half {
            width: 100%;
            padding-right: 0;
        }
    }
    .form_radio_btn {
        display: inline-block;
        margin-right: 10px;
    }
    .form_radio_btn input[type=radio] {
        display: none;
    }
    .form_radio_btn label {
        display: inline-block;
        cursor: pointer;
        padding: 0px 15px;
        line-height: 34px;
        border: 1px solid #999;
        border-radius: 6px;
        user-select: none;
    }
    /* Checked */
    .form_radio_btn input[type=radio]:checked+label {
        background: #ffe0a6;
    }
    /* Hover */
    .form_radio_btn label:hover {
        color: rgb(167, 39, 39);
        background-color: #ffe0a6;
    }
    /* Disabled */
    .form_radio_btn input[type=radio]:disabled+label {
        background: #efefef;
        color: #666;
    };
```

Г.13 Файл `main.css`

```
p {
    color: black;
    font-weight: normal;
    letter-spacing: normal;
    word-spacing: normal;
    font-family: 'Times New Roman', Times, serif;
}
nav {
    color: white;
    background-color: darkblue;
    font-size: larger;
    font-weight: bold;
    font-family: Arial, Helvetica, sans-serif;
}
h1 {
    font-family: Arial, Helvetica, sans-serif;
    text-align: center;
}
footer {
    position: relative;
    color: black;
    text-align: center;
    opacity: 0.5;
    font-size: smaller;
    bottom: 0px;
    right: auto;
    left: auto;
    width: 98%;
```

```
}
select {
    width: min-content;
}
html,
body {
    height: 100%;
    padding: 0;
    margin: 10;
}
/* Navbar container */
.navbar {
    overflow: hidden;
    font-family: Arial;
}
/* Links inside the navbar */
.navbar a {
    float: left;
    font-size: 16px;
    color: white;
    text-align: center;
    padding: 14px 14px;
    text-decoration: none;
    border: solid 2px rgb(11, 175, 11);
    border-radius: 5px;
    margin: 5 0 0 5;
    background-color: rgb(13, 225, 36);
}
/* The dropdown container */
.dropdown {
    float: left;
    margin: 5 0 0 5;
    overflow: hidden;
}
/* Dropdown button */
.dropdown .dropbtn {
    font-size: 16px;
    border: solid 2px darkgreen;
    border-radius: 5px;
    margin: 5 0 0 5;
    outline: none;
    color: white;
    padding: 14px 16px;
    background-color: rgb(18, 148, 87);
    font-family: inherit;
    /* Important for vertical align on mobile phones */
    margin: 0;
    /* Important for vertical align on mobile phones */
}
/* Add a red background color to navbar links on hover */
.navbar a:hover,
.dropdown:hover .dropbtn {
    background-color: rgba(105, 155, 97, 0.509);
}
/* Dropdown content (hidden by default) */
.dropdown-content {
    display: none;
    position: absolute;
    min-width: 10px;
    z-index: 1;
}
/* Links inside the dropdown */
.dropdown-content a {
    float: none;
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;
    background-color: white;
}
```

```
/* Add a grey background color to dropdown links on hover */
.dropdown-content a:hover {
    background-color: rgb(131, 126, 126);
}
/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {
    display: block;
}
/* Button used to open the contact form - fixed at the bottom of the page */
.open-button {
    font-size: 16px;
    overflow: hidden;
    background-color: #333;
    font-family: Arial;
    color: white;
    padding: 16px 20px;
    border: none;
    cursor: pointer;
    position: relative;
    float: right;
    top: auto;
    max-width: 180px;
}
.enterbtn {
    float: right;
}
.btn {
    background-color: rgb(63, 204, 63);
    width: 100%;
    height: 100%;
    color: white;
    text-decoration: none;
    border: none;
    margin: 0;
    padding: 8;
}
.btn:disabled {
    display: none;
    /*
  opacity: 0.3;
  cursor: not-allowed;*/
}
/* CHART*/
.graphics {
    display: flex;
    justify-content: center;
}
.chart-container {
    height: 200;
    width: 400;
    display: inline-block;
    margin-right: 5px;
    margin-top: 5px;
}
.counter-div {
    height: 200;
    width: 400;
    display: inline-block;
    margin-right: 5px;
    margin-top: 5px;
}
.counter-div p {
    text-align: center;
    color: #535353;
}
.counter-div .count-p {
    color: red;
    font-size: 48px;
    font-weight: 500;
}
.faq a {
```

```
    font-size: 24px;
    ;
    color: crimson;
    font-style: bold;
    font-family: 'Courier New', Courier, monospace;
}
.faq p {
    color: #000;
    font-size: larger;
    font-family: Arial, Helvetica, sans-serif
}
.year-switch {
    text-decoration: none;
    border: solid 2px darkgreen;
    background-color: rgb(255, 255, 255);
    color: darkgreen;
    margin: 5 5 1 1;
    padding: 20;
    border-radius: 40px;
}
.year-switch:hover {
    background-color: #45c779;
}
.hide-button {
    width: full-width;
    border: solid 2px black;
    text-decoration: none;
    padding: 10 10;
    margin: 20;
    color: black;
    background-color: lightgoldenrodyellow;
}
table {
    /*margin: 5%;*/
    width: 100%;
    border: solid 2px darkgreen;
    font-size: 0.9em;
    font-family: sans-serif;
    /*min-width: 400px;*/
    border-collapse: collapse;
    /*box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);*/
}
th {
    background-color: rgb(13, 225, 36);
}
thead tr {
    background-color: rgb(13, 225, 36);
    /*color: #ffffff;*/
    text-align: center;
}
th,
td,
tr {
    padding: 10px;
    text-align: center;
    /*border: solid 1px lightgrey;*/
}
tr {
    border-bottom: 1px solid #000000;
}
tr:last-of-type {
    border-bottom: 2px solid #1a1e1d;
}
*/ .error {
    color: red;
}
.div_form {
    display: none;
}
.modal-body {
    background: white;
```

```
}
.manager-success {
    text-decoration: none;
    text-align: center;
    color: white;
    background-color: orange;
    font-size: larger;
    width: 100%;
    margin: 0 0 0 0;
    border: none;
}
.manager-success a {
    text-decoration: none;
    color: white;
}
.editModal {
    position: fixed;
    font-family: Arial, Helvetica, sans-serif;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    background: rgba(0, 0, 0, 0.8);
    z-index: 99999;
    opacity: 0;
    -webkit-transition: opacity 400ms ease-in;
    -moz-transition: opacity 400ms ease-in;
    transition: opacity 400ms ease-in;
    pointer-events: none;
}
.editModal:target {
    opacity: 1;
    pointer-events: auto;
}
.editModal>div {
    width: 400px;
    position: relative;
    margin: 10% auto;
    padding: 5px 20px 13px 20px;
    border-radius: 10px;
    background: #fff;
}
.close {
    background: #606061;
    color: #FFFFFF;
    line-height: 25px;
    position: absolute;
    right: -12px;
    text-align: center;
    top: -10px;
    width: 24px;
    text-decoration: none;
    font-weight: bold;
    -webkit-border-radius: 12px;
    -moz-border-radius: 12px;
    border-radius: 12px;
    -moz-box-shadow: 1px 1px 3px #000;
    -webkit-box-shadow: 1px 1px 3px #000;
    box-shadow: 1px 1px 3px #000;
}
.close:hover {
    background: #00d9ff;
}
.listbutton th {
    height: 85%;
    width: 100%;
    background-color: #45c779;
    border: none;
    border-bottom: #000000;
}
.listbutton {
```

```
    background: none;
    width: 95%;
    height: 5em;
    margin: 5px;
    padding: 5px;
}
.listbutton th:hover {
    background-color: #c7eb26;
}
.listbutton::after {
    content: " ";
    border-bottom: #373f3a;
}
.modal {
    display: none;
    position: fixed;
    z-index: 1;
    padding-top: 10%;
    left: 0;
    top: 0;
    width: 100%;
    height: 85%;
    overflow: auto;
    background-color: rgb(0, 0, 0);
    background-color: rgba(0, 0, 0, 0.4);
}
.modal-table {
    max-width: max-content;
    border: solid 1px grey;
    border-collapse: collapse;
    background-color: #ffffff;
    margin: auto;
}
/* Modal Content */
.modal-content {
    background-color: #fefefe;
    margin: auto;
    padding: 5px;
    border: 1px solid #888;
    width: min-content;
}
#card_table {
    display: block;
    margin-top: 10%;
    position: relative;
    width: 100%;
}
#div3 {
    margin: 0% 10% 0 10%;
    width: 100%;
    text-align: center;
    float: left;
}
h3 {
    text-align: center;
}
#div1 {
    float: left;
    width: 100%;
    height: 100%;
    text-align: center;
    margin: auto;
    background-color: rgba(15, 112, 15, 0.549);
}
#div2 {
    text-align: center;
    margin: auto;
    top: auto;
    left: auto;
    background-color: rgba(38, 182, 38, 0.549);
    float: left;
```

```
    width: 75%;
    height: 100%;
}
img {
    display: block;
    margin-left: auto;
    margin-right: auto;
}
/* The Close Button */
.close {
    margin: 10%;
    color: #aaaaaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}
.close:hover,
.close:focus {
    color: #000;
    text-decoration: none;
    cursor: pointer;
}
.patientsTable,
.docsTable table {
    border: 2px grey;
    text-align: center;
    min-width: 400px;
    max-width: 80%;
    padding: 2em;
    margin: 5em;
    overflow-y: auto;
}
```

ДОДАТОК Д

Слайди презентації

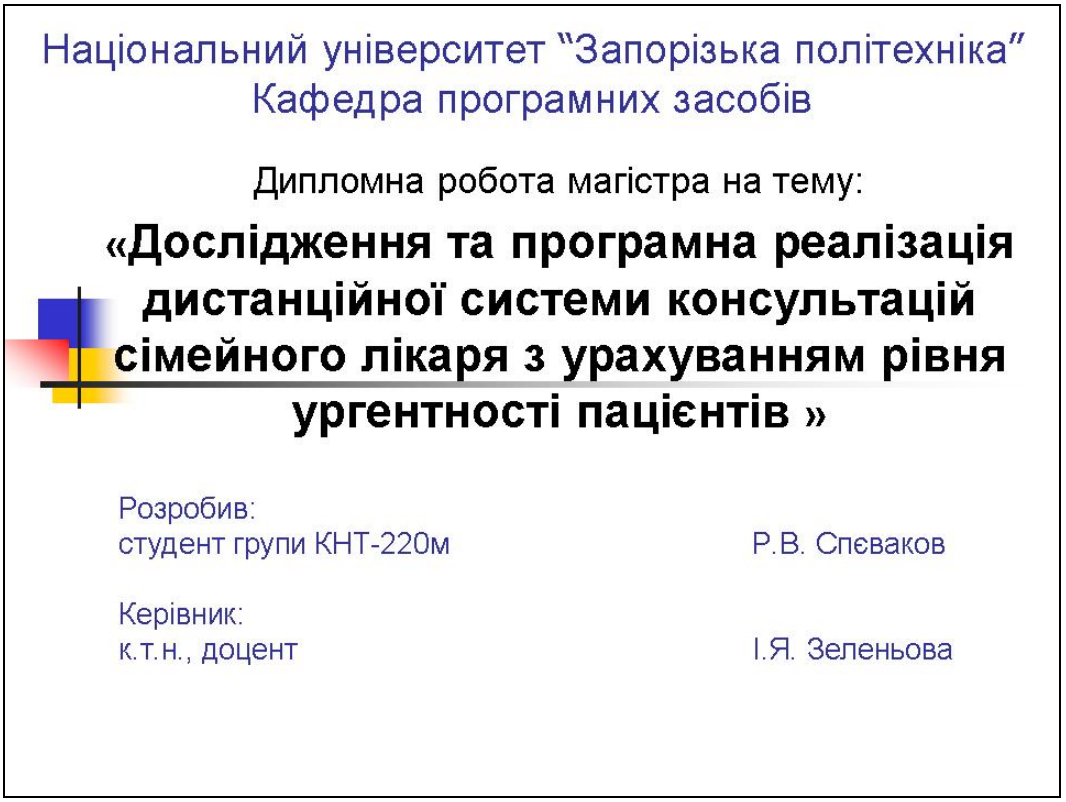


Рисунок Д.1 – Слайд № 1

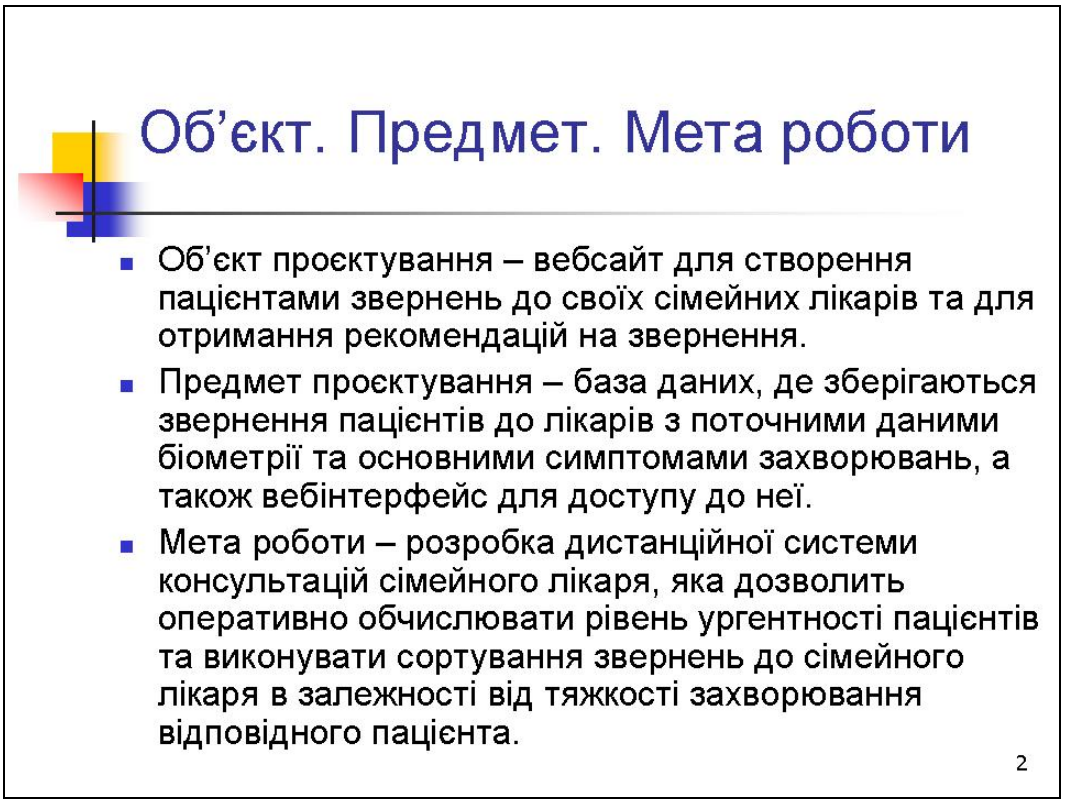


Рисунок Д.2 – Слайд № 2

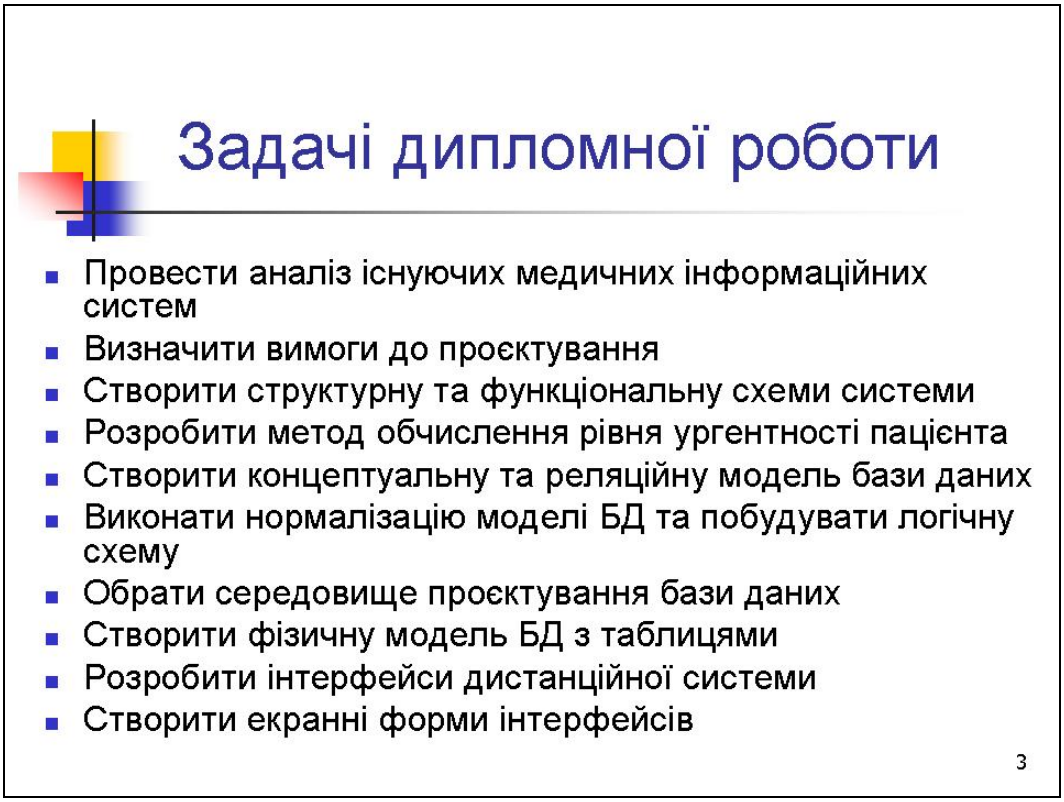


Рисунок Д.3 – Слайд № 3

Рисунок Д.4 – Слайд № 4

Рисунок Д.5 – Слайд № 5

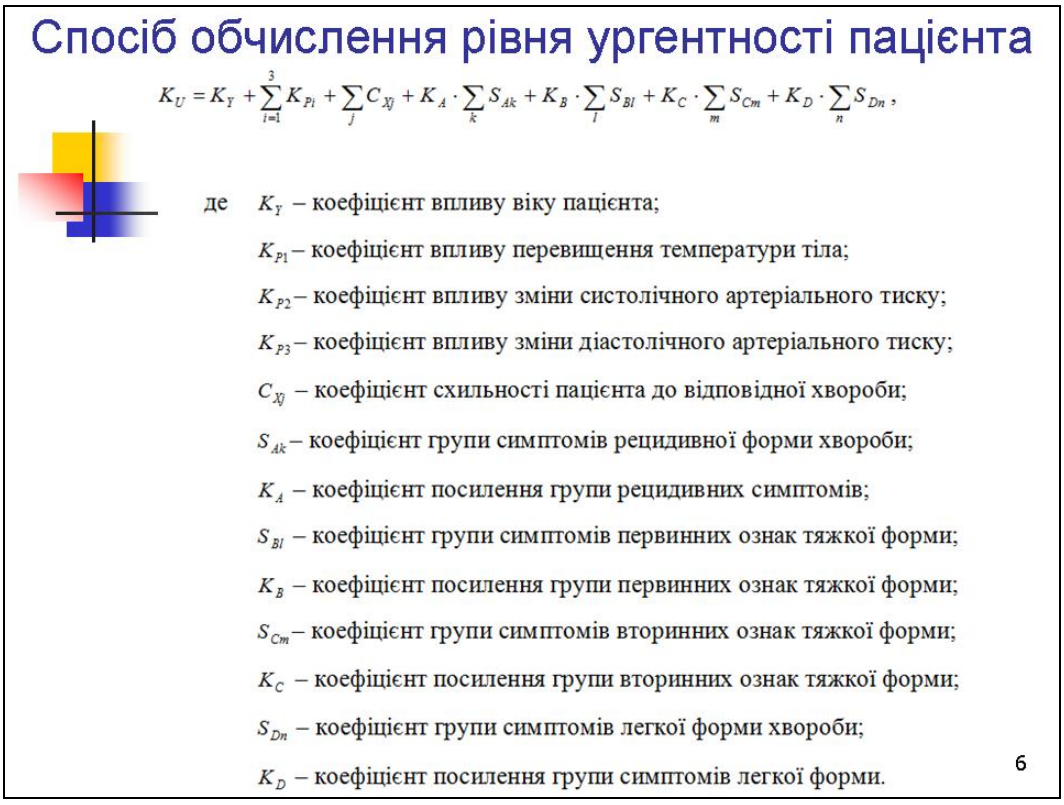


Рисунок Д.6 – Слайд № 6

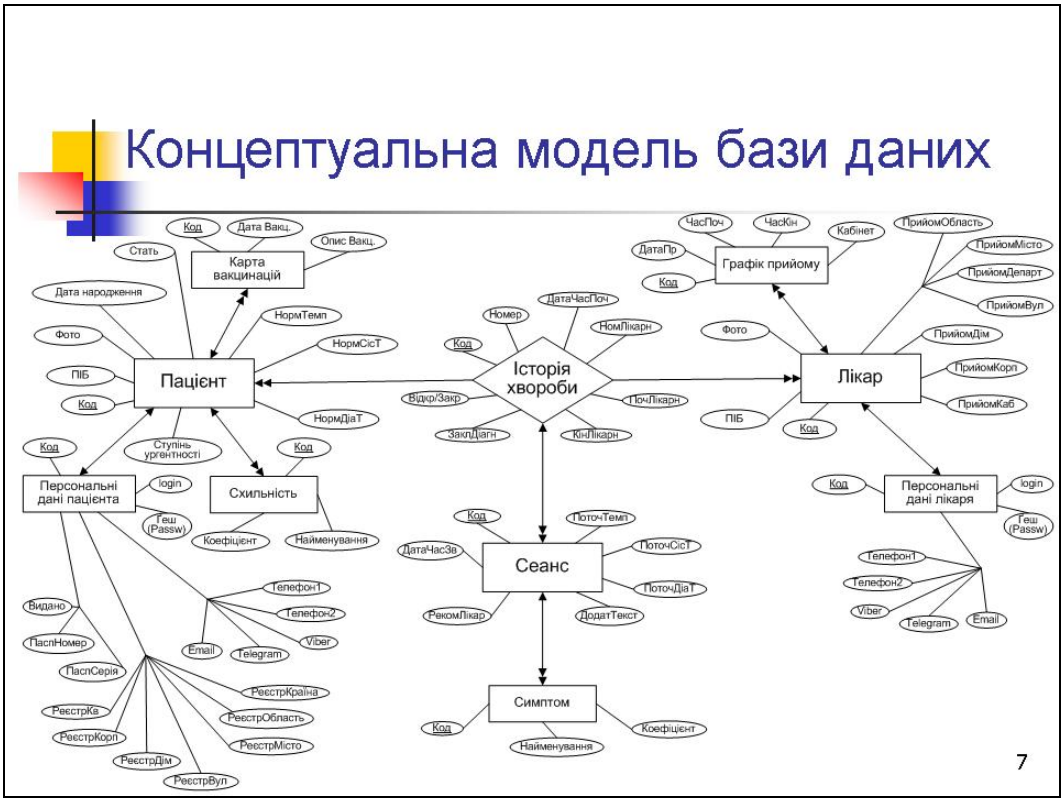


Рисунок Д.7 – Слайд № 7

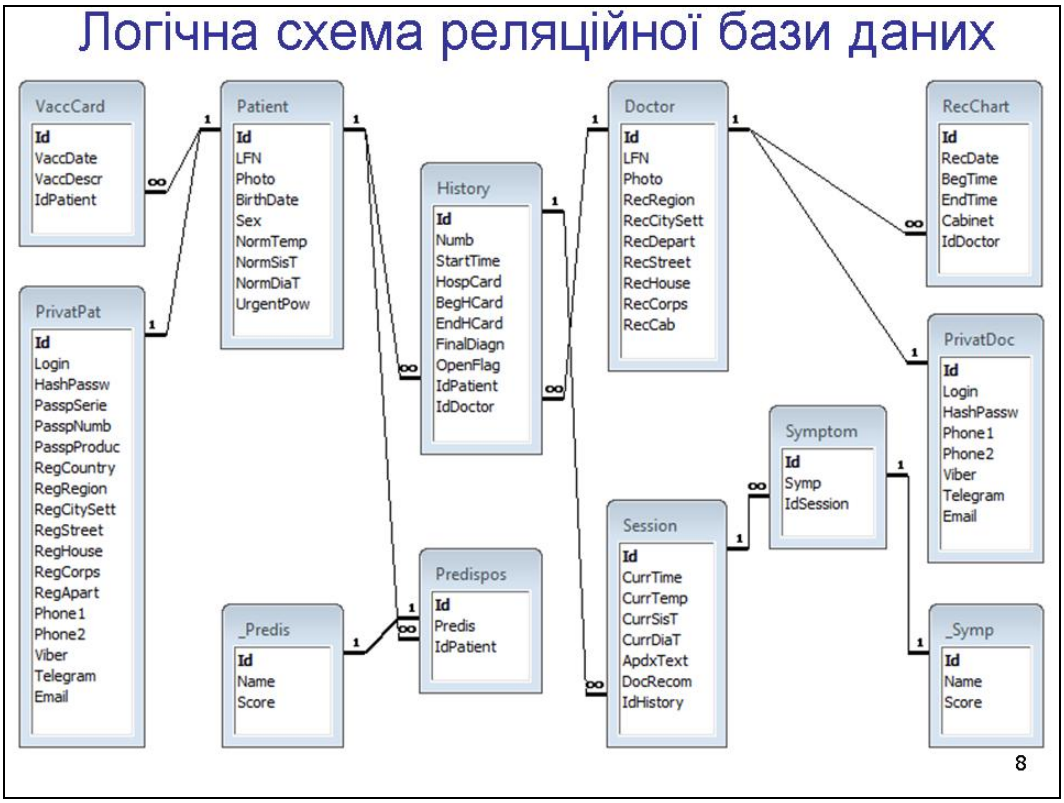


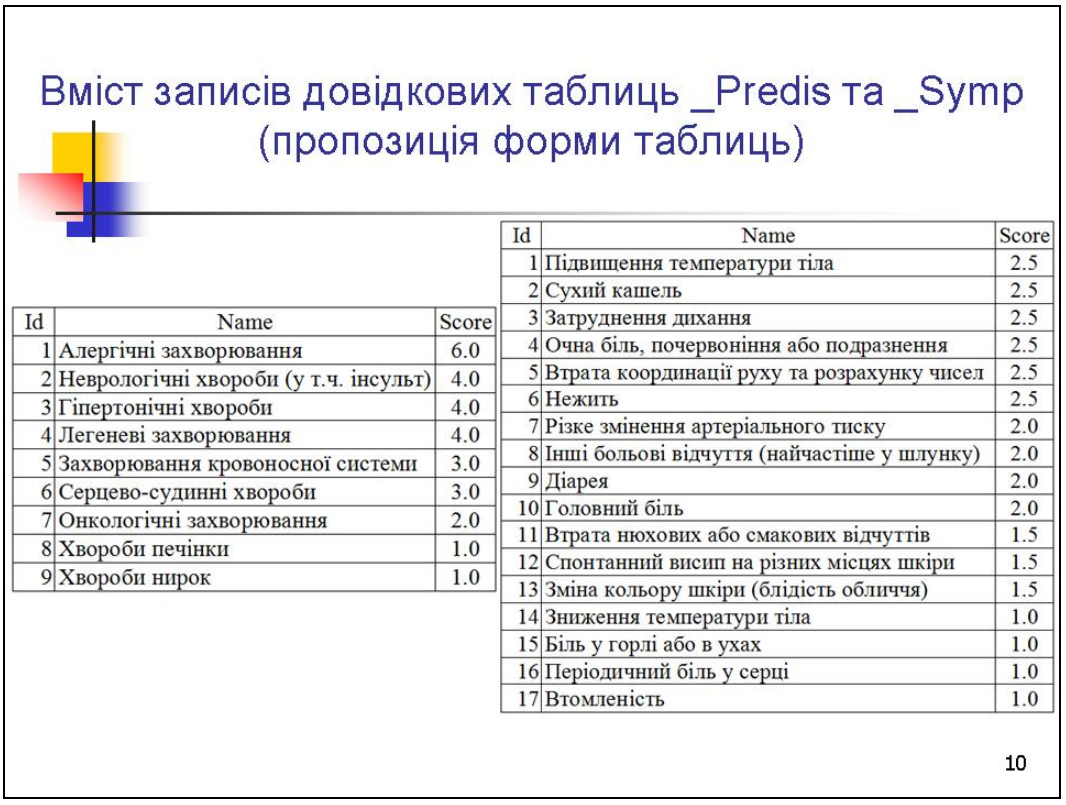
Рисунок Д.8 – Слайд № 8

Порівняльний аналіз особливостей різних СКБД

Назва СКБД	Розробник	Тип	Операційна система	Початковий код	Ліцензія	Підтримка	Популяр-ність	Стабіль-ність
SQL Server	Microsoft	Реляційна	Linux, Microsoft Windows	Закритий	Комерційна	Безкоштовна	+	+
MySQL	Oracle Corporation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Відкритий	GNU GPL та комерційна	За гроші	+	+
PostgreSQL	PostgreSQL Global Development Group	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX, macOS, QNX	Відкритий	Вільне та відкрите програмне забезпечення, дозвільна ліцензія	За гроші	+	+
Oracle Database	Oracle Corporation	Мульти-модельна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX	Закритий	Комерційна	За гроші	+	+
SAP HANA	SAP SE	Реляційна, in-memory	Linux	Закритий	Комерційна	За гроші	–	+
MongoDB	MongoDB Inc.	Документо-орієнтована	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Відкритий	GNU AGPL (СКБД) та Apache License (драйвери)	За гроші	+	+
DB2	IBM	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Закритий	Пропрієтарна EULA	Безкоштовна	–	+
MariaDB	**MariaDB Corporation AB, MariaDB Foundation**	**Реляційна**	**Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD**	**Відкритий**	**GNU GPL**	**За гроші**	–	–

9

Рисунок Д.9 – Слайд № 9



Вміст записів довідкових таблиць _Predis та _Symp (пропозиція форми таблиць)

Id	Name	Score
1	Алергічні захворювання	6.0
2	Неврологічні хвороби (у т.ч. інсульт)	4.0
3	Гіпертонічні хвороби	4.0
4	Легеневі захворювання	4.0
5	Захворювання кровоносної системи	3.0
6	Серцево-судинні хвороби	3.0
7	Онкологічні захворювання	2.0
8	Хвороби печінки	1.0
9	Хвороби нирок	1.0

Id	Name	Score
1	Підвищення температури тіла	2.5
2	Сухий кашель	2.5
3	Затруднення дихання	2.5
4	Очна біль, почервоніння або подразнення	2.5
5	Втрата координації руху та розрахунку чисел	2.5
6	Нежить	2.5
7	Різке змінення артеріального тиску	2.0
8	Інші больові відчуття (найчастіше у шлунку)	2.0
9	Діарея	2.0
10	Головний біль	2.0
11	Втрата нюхових або смакових відчуттів	1.5
12	Спонтанний висип на різних місцях шкіри	1.5
13	Зміна кольору шкіри (блідість обличчя)	1.5
14	Зниження температури тіла	1.0
15	Біль у горлі або в ухах	1.0
16	Періодичний біль у серці	1.0
17	Втомленість	1.0

10

Рисунок Д.10 – Слайд № 10

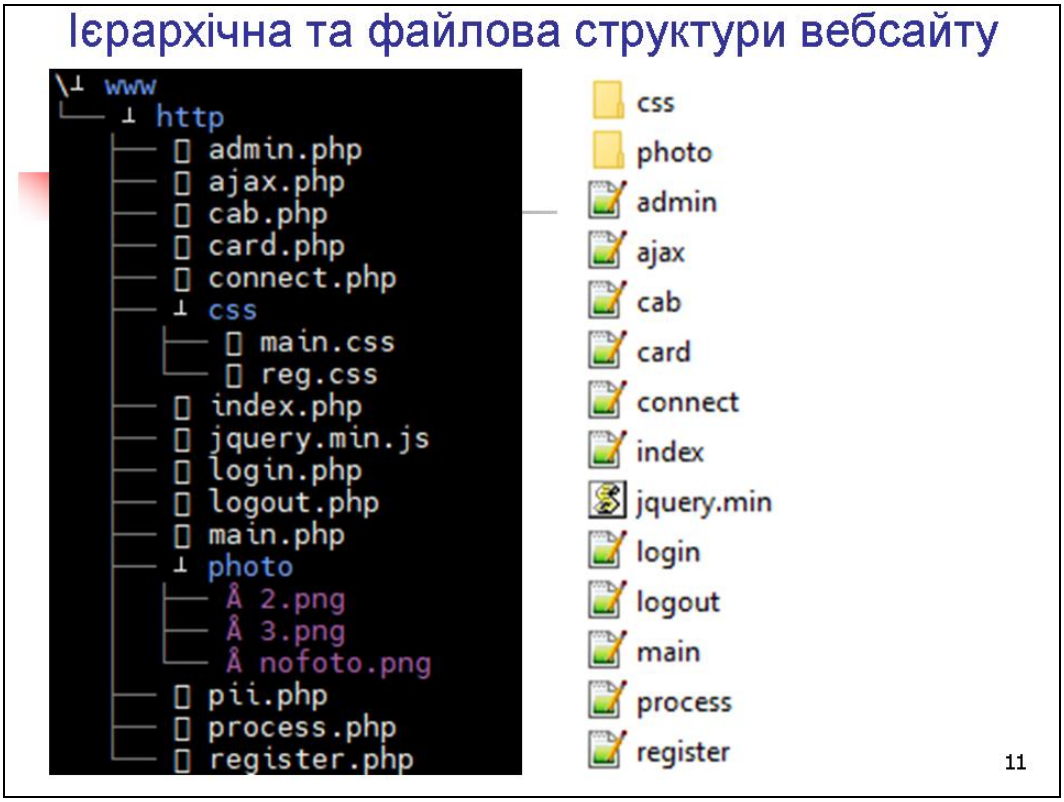


Рисунок Д.11 – Слайд № 11

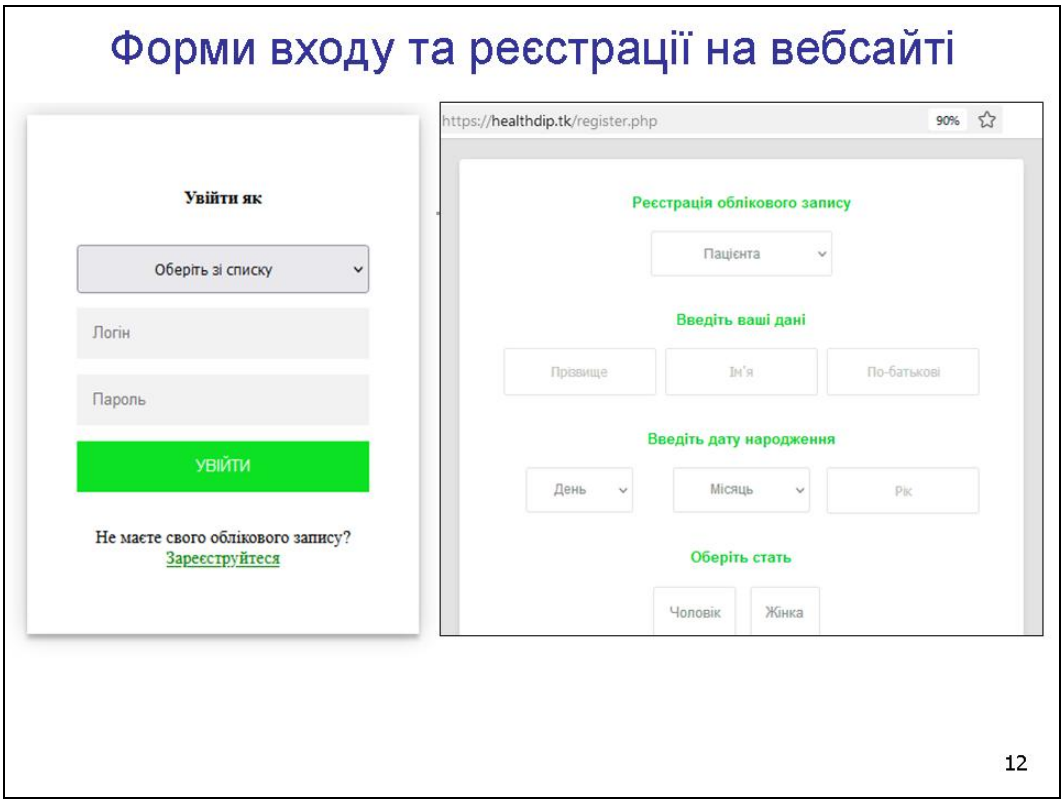


Рисунок Д.12 – Слайд № 12

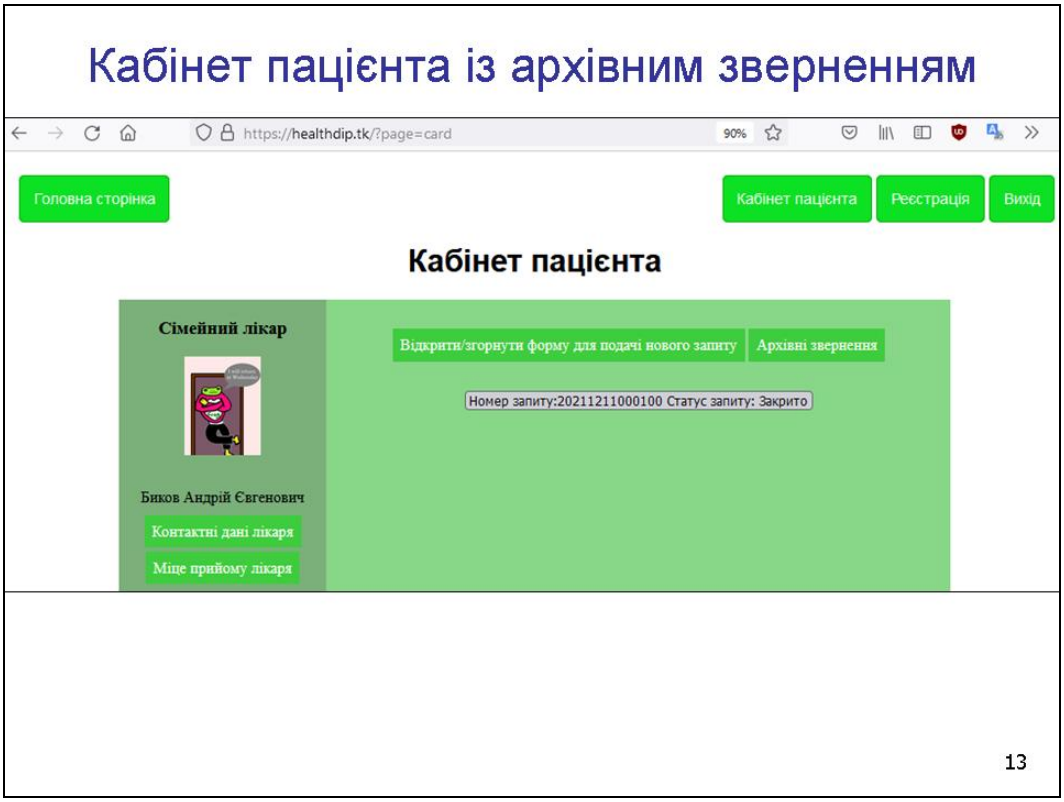


Рисунок Д.13 – Слайд № 13

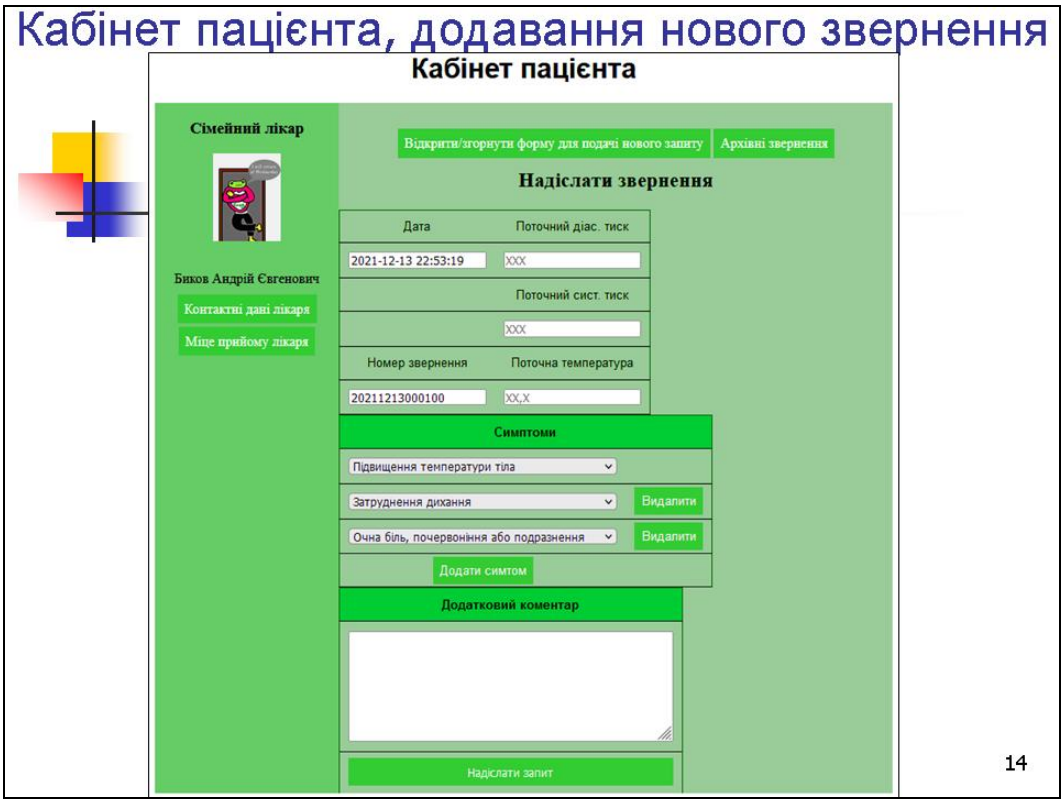


Рисунок Д.14 – Слайд № 14

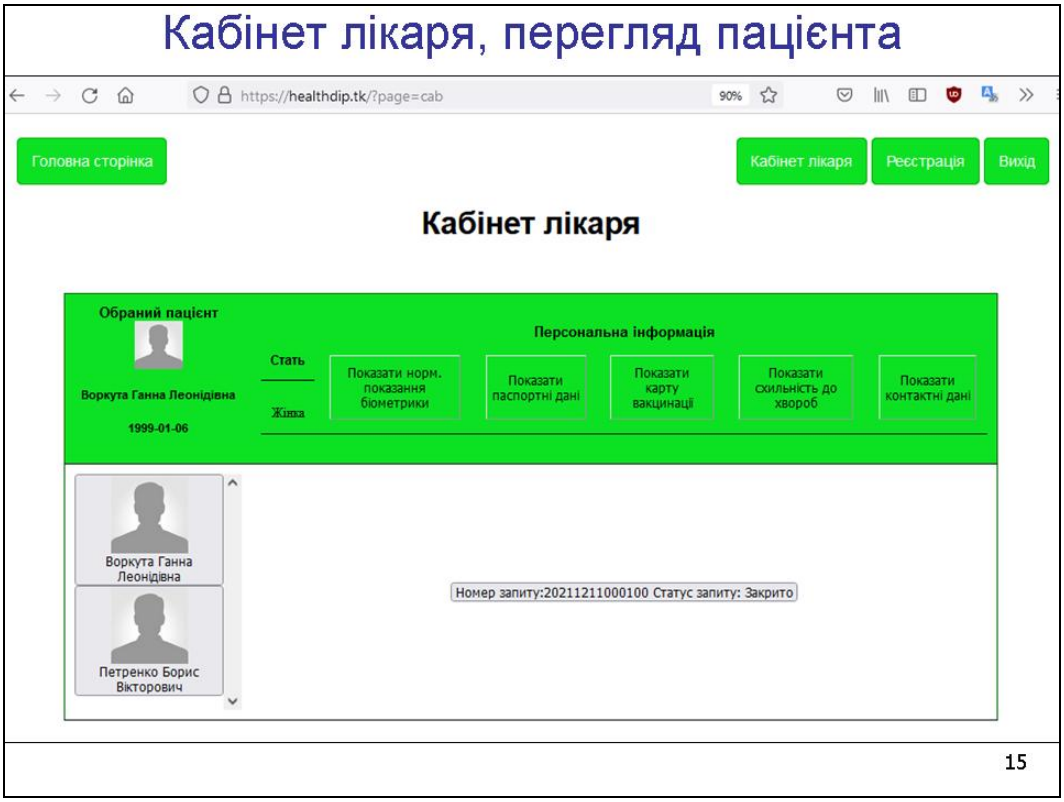


Рисунок Д.15 – Слайд № 15

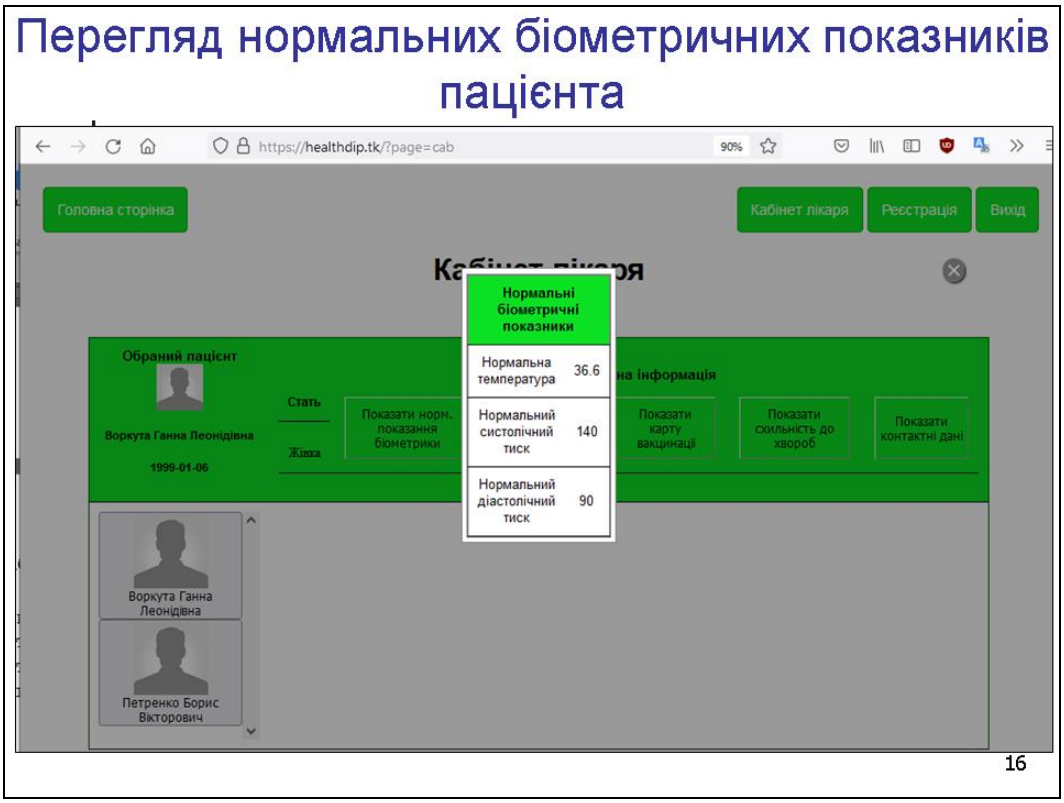


Рисунок Д.16 – Слайд № 16

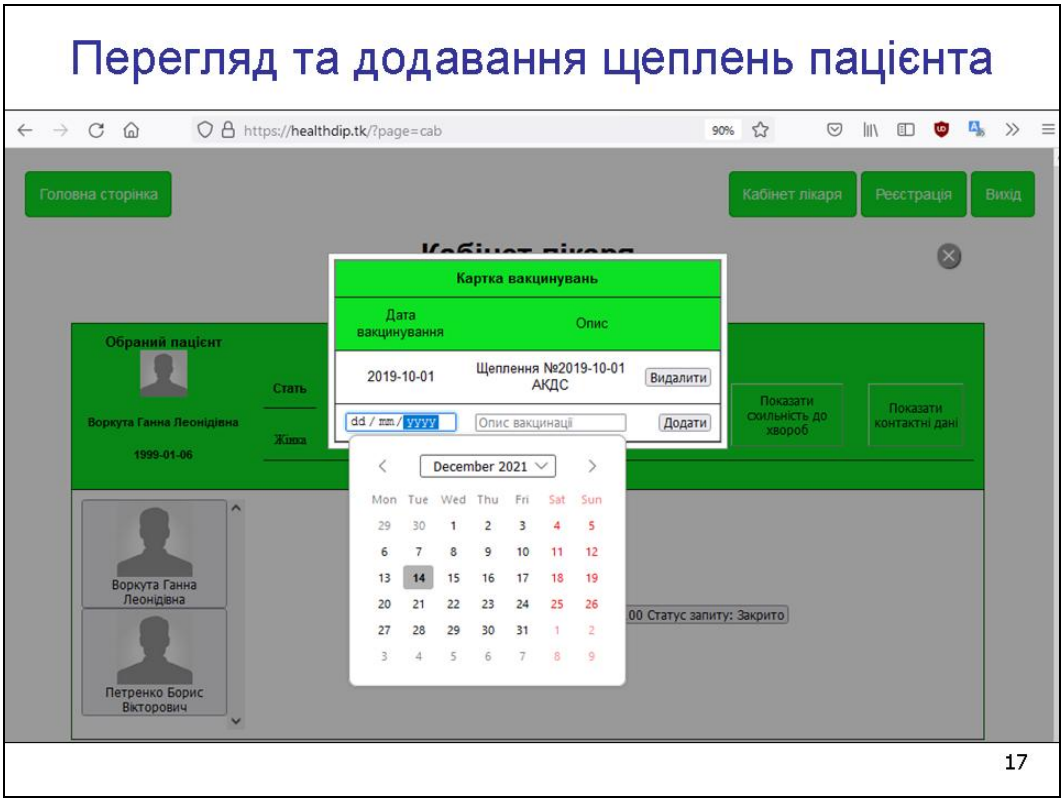


Рисунок Д.17 – Слайд № 17

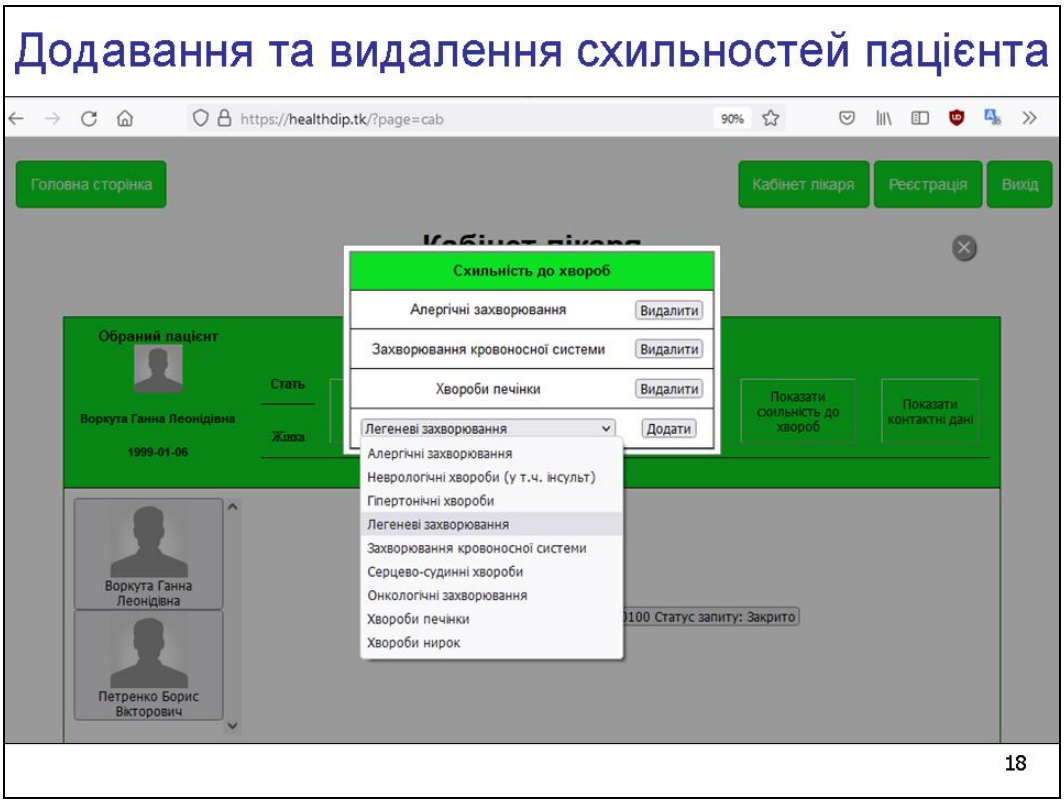


Рисунок Д.18 – Слайд № 18

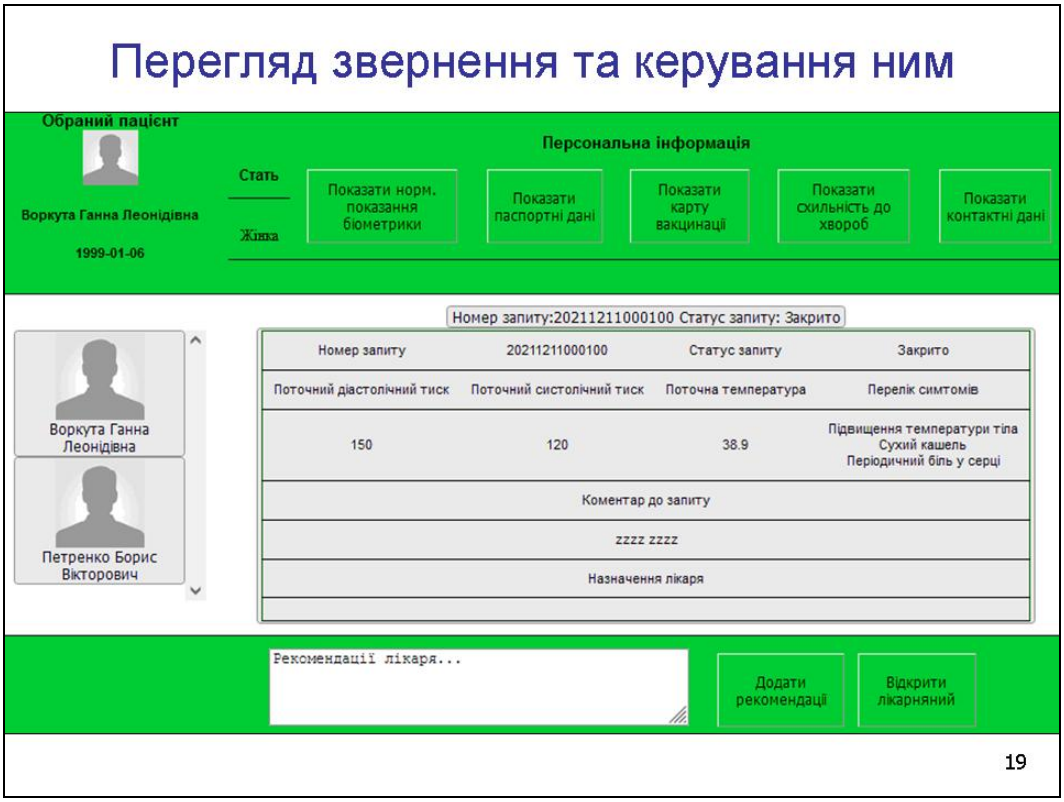


Рисунок Д.19 – Слайд № 19

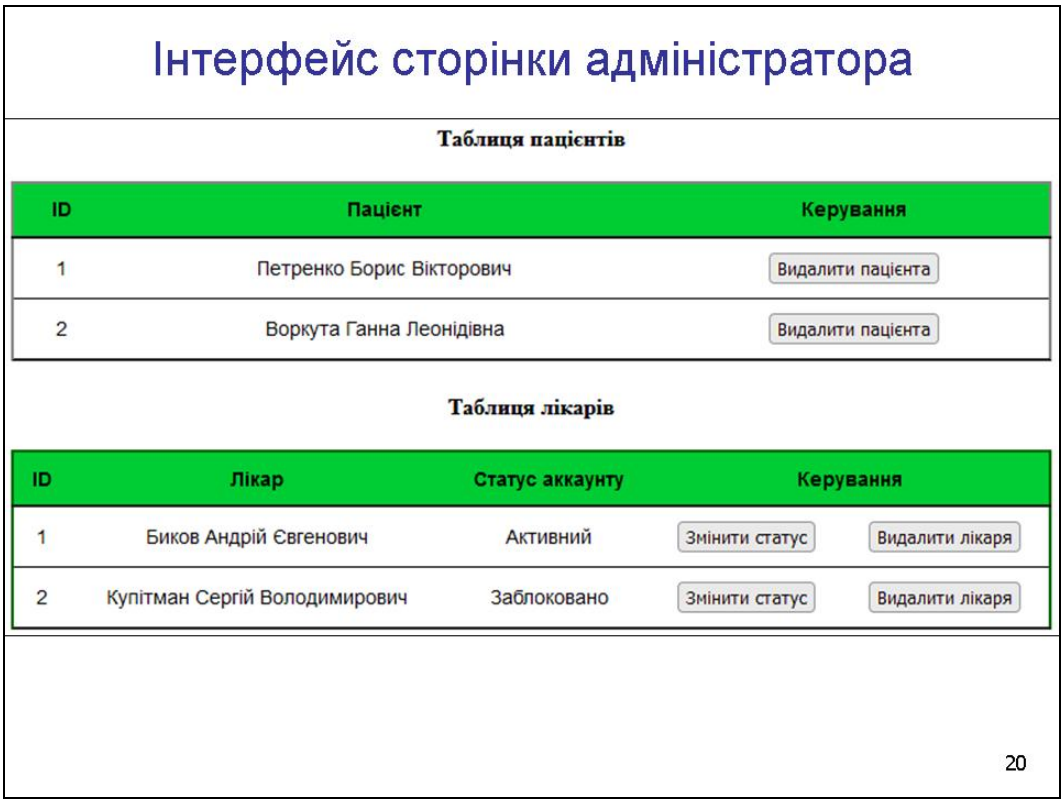


Рисунок Д.20 – Слайд № 20

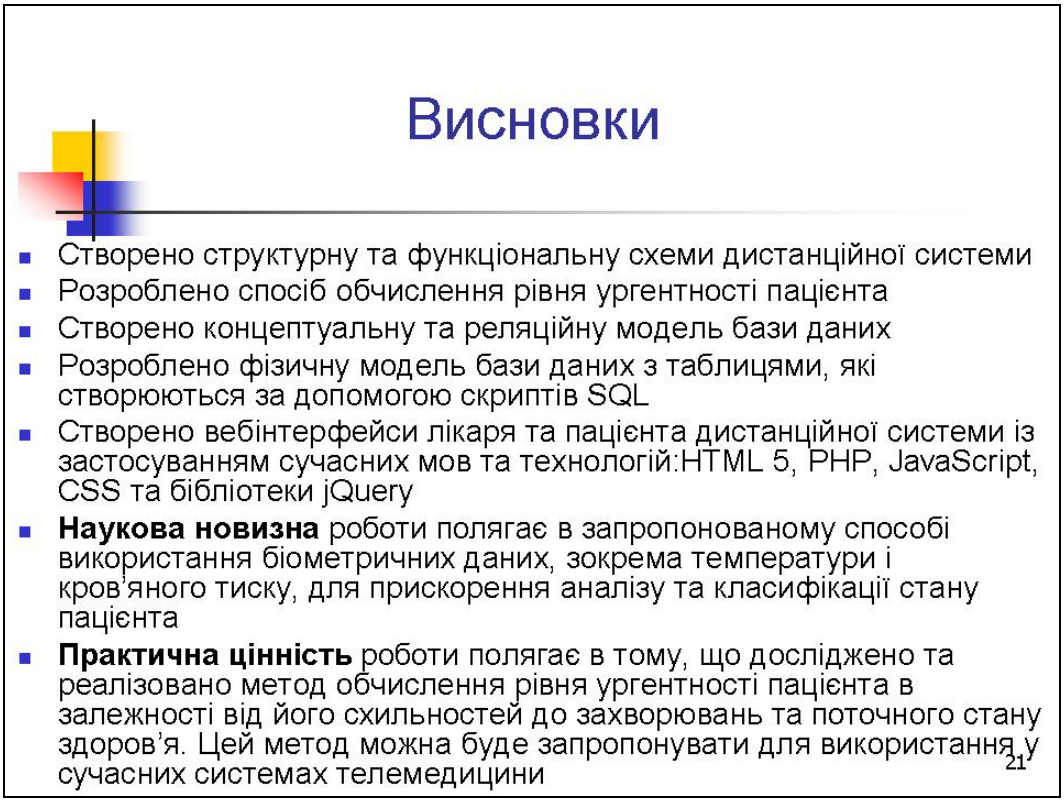


Рисунок Д.21 – Слайд № 21