

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»

Завідувач кафедри

Наталія АУШЕВА

«__» _____ 2026 р.

**Дипломна робота
на здобуття ступеня бакалавра**

За освітньою програмою «Цифрові технології в енергетиці»

Спеціальності 122 «Комп'ютерні науки»

на тему: «Вебсистема електронного обліку пацієнтів та медичних призначень»

Виконала: студентка 4 курсу, групи ТР-21

Батринчук Надія

(прізвище, ім'я, по батькові)

(підпис)

Керівник:

асистент каф. ЦТЕ, Рудик Володимир Іванович

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

Рецензент:

зав. каф. ТАЕ, к.т.н., доц. Пешко Віталій Анатолійович

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

Н.контроль старший викладач Беспала Ольга Миколаївна

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

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

Студентка _____

(підпис)

Київ — 2026

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра

ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти — перший (бакалаврський)

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Цифрові технології в енергетиці»

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

_____ Наталія АУШЕВА

«__» _____ 2026 р.

З А В Д А Н Н Я
на дипломну роботу студентці

Батринчук Надії

(прізвище, ім'я, по батькові)

1. Тема роботи «Вебсистема електронного обліку пацієнтів та медичних призначень».

керівник роботи Рудик Володимир Іванович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026р. № 1992-с

2. Термін подання роботи студентом 08.06.2026р.

3. Вихідні дані до роботи: мова програмування PHP, мова розмітки HTML, таблиці стилів CSS, мова сценаріїв JavaScript, система керування базами даних MySQL, архітектурний шаблон MVC, вебсервер Apache, система контролю версій Git, середовище розробки VS Code.

4. Перелік питань, які потрібно розробити: 1) проаналізувати наявні медичні інформаційні системи та визначити їхні переваги й недоліки; 2) обрати технології для розробки вебсистеми обліку пацієнтів та обґрунтувати їх використання; 3) спроектувати структуру бази даних для зберігання інформації про пацієнтів, лікування і документи; 4) реалізувати та протестувати вебсистему з ролями лікаря, адміністратора і пацієнта.

5. Орієнтовний перелік ілюстративного матеріалу: діаграма USE-CASE, схема архітектури системи, концептуальна модель даних, логічна модель даних, скріншот інтерфейсу користувача.

6. Дата видачі завдання 01.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Затвердження теми роботи	28.05.2026	Виконано
2	Дослідження існуючих рішень для електронного обліку пацієнтів та визначення вимог до системи	14–20.04.26	Виконано
3	Розробка UI/UX дизайну клієнтської частини	21–23.04.26	Виконано
4	Розробка архітектури системи та проектування бази даних	24–25.04.26	Виконано
5	Розробка клієнтської та серверної частини системи	26–27.04.26	Виконано
6	Програмна реалізація вебсистеми електронного обліку пацієнтів та медичних призначень	28.04–04.05.26	Виконано
7	Оформлення записки	05–13.05.26	Виконано
8	Захист програмного продукту	14.05.26	Виконано
9	Передзахист	03.06.26	Виконано
10	Захист	18.06.26	Виконано

Студентка

Керівник роботи

(підпис)

(підпис)

Надія БАТРИНЧУК

(ім'я, ПРИЗВИЩЕ)

Володимир РУДИК

(ім'я, ПРИЗВИЩЕ)

АНОТАЦІЯ

Робота містить 26 ілюстрацій, 3 таблиці, 4 додатки, 30 джерел у переліку посилань, загальних обсяг 67 сторінок.

Мета роботи: розробка вебсистеми електронного обліку пацієнтів та медичних призначень, орієнтованої на централізоване зберігання медичної інформації, ведення історії лікувань та забезпечення зручної взаємодії між лікарем і пацієнтом.

Методи та засоби: архітектурний шаблон MVC, мова програмування PHP, система керування базами даних MySQL, мови HTML, CSS та JavaScript, вебсервер Apache, система авторизації та розмежування доступу користувачів, середовище розробки VS Code.

Результат: вебсистема електронного обліку пацієнтів та медичних призначень із можливістю керування базою пацієнтів, створення записів лікування, прикріплення медичних документів і перегляду історії лікувань через вебінтерфейс.

Апробація результатів дипломної роботи: Батринчук Н. Вебсистема електронного обліку пацієнтів та медичних призначень. Scientific Method: Reality and Future Trends of Researching : Proceedings of the VIII International Scientific and Theoretical Conference, м. Монреаль, 29 трав. 2026 р. Монреаль, Канада, 2026. С. 158–160. URL: <https://doi.org/10.36074/scientia-29.05.2026>.

Ключові слова: ЕЛЕКТРОННИЙ ОБЛІК ПАЦІЄНТІВ, ВЕБСИСТЕМА, МЕДИЧНІ ПРИЗНАЧЕННЯ, ІСТОРІЯ ЛІКУВАНЬ, MVC.

ABSTRACT

The work contains 26 illustrations, 3 table, 4 appendices, 30 sources in the reference list, a total of 67 pages.

Objective of the work: development of a web-based electronic patient records and medical prescriptions management system focused on centralized storage of medical information, maintaining treatment history, and ensuring convenient interaction between doctors and patients.

Methods and tools: MVC architectural pattern, PHP programming language, MySQL database management system, HTML, CSS and JavaScript languages, Apache web server, user authorization and access control system, VS Code development environment.

Result: a web-based electronic patient records and medical prescriptions management system with the ability to manage a patient database, create treatment records, attach medical documents, and view treatment history through a web interface.

Approbation of the results of the thesis. Batrynychuk N. Web-Based System for Electronic Management of Patient Records and Medical Prescriptions. Scientific Method: Reality and Future Trends of Researching : Proceedings of the VIII International Scientific and Theoretical Conference, Montreal, May 29, 2026. Montreal, Canada, 2026. PP. 158–160. DOI: <https://doi.org/10.36074/scientia-29.05.2026.31>

Keywords: ELECTRONIC PATIENT RECORDS, WEB SYSTEM, MEDICAL PRESCRIPTIONS, TREATMENT HISTORY, MVC.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП.....	8
1 ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ЗАСОБІВ ДЛЯ РОЗРОБКИ ВЕБСИСТЕМИ ОБЛІКУ ПАЦІЄНТІВ	10
1.1 Постановка задачі, мета та завдання розробки	10
1.2 Функціональні та нефункціональні вимоги до системи.....	11
1.3 Аналіз існуючих медичних інформаційних систем.....	14
1.4 Порівняння з відомими платформами обліку пацієнтів.....	16
1.5 Визначення ключових переваг та недоліків існуючих рішень.....	18
1.6 Обґрунтування необхідності створення власної системи.....	21
2 ПРОЄКТУВАННЯ ТА ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ СИСТЕМИ ЕЛЕКТРОННОГО ОБЛІКУ ПАЦІЄНТІВ	24
2.1 Загальна архітектура системи	24
2.2 Вибір платформи та технологій.....	28
2.3 Проєктування бази даних	30
2.4 Модель взаємодії компонентів	32
2.5 Засоби розгортання системи на вебсервері	34
2.6 Забезпечення безпеки та збереження медичних даних	36
3 РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ ЕЛЕКТРОННОГО ОБЛІКУ ПАЦІЄНТІВ ТА МЕДИЧНИХ ПРИЗНАЧЕНЬ	40
3.1 Структура проєкту та логіка взаємодії модулів	40
3.2 Реалізація серверної частини	43
3.3 Реалізація бази даних MySQL.....	45
3.4 Реалізація користувацького інтерфейсу.....	47
3.5 Модуль керування користувачами	48
3.6 Модуль створення та перегляду записів лікування.....	50
3.7 Модуль додавання результатів прийому та документів	52

3.8 Модуль журналу дій.....	53
3.9 Тестування та перевірка працездатності.....	54
4 СЦЕНАРІЇ ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ ВЕБСИСТЕМИ	57
4.1 Системні вимоги.....	57
4.2 Сценарії роботи користувачів	58
4.3 Робота з ролями	61
4.4 Журнал подій	64
ВИСНОВКИ.....	66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68
ДОДАТОК А.....	71
ДОДАТОК Б	76
ДОДАТОК В	78
ДОДАТОК Г	82

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

API — Application Programming Interface.

Apache — вебсервер для обробки HTTP-запитів та роботи вебдодатка.

CSS — Cascading Style Sheets.

CRUD — Create, Read, Update, Delete — базові операції роботи з даними.

DOM — Document Object Model.

HTML — HyperText Markup Language.

HTTP — Hypertext Transfer Protocol.

HTTPS — Hypertext Transfer Protocol Secure.

IP — Internet Protocol.

JS — JavaScript.

JSON — JavaScript Object Notation.

MVC — архітектурний шаблон побудови вебдодатків.

MySQL — система керування реляційними базами даних.

PHP — серверна мова програмування для створення вебдодатків.

SQL — Structured Query Language.

SSL — Secure Sockets Layer.

TLS — Transport Layer Security.

UI — User Interface.

URL — Uniform Resource Locator.

UTF-8 — формат кодування символів Unicode.

БД — база даних.

ВСТУП

Ефективна організація процесу обліку пацієнтів, їх лікування та медичних призначень є одним із ключових чинників підвищення якості надання медичних послуг у сучасній системі охорони здоров'я. Традиційні методи ведення пацієнтських записів — паперові журнали, неструктуровані таблиці або розрізнені електронні файли — часто призводять до дублювання інформації, втрати даних та ускладнюють взаємодію між лікарями. У результаті зростає ризик помилок, сповільнюється доступ до історії лікування пацієнта й знижується загальна ефективність медичного процесу.

Розвиток вебтехнологій відкриває нові можливості для створення інтегрованих рішень, що дозволяють централізовано зберігати, обробляти й аналізувати медичну інформацію. Вебсистеми забезпечують швидкий доступ до бази даних через браузер, мінімізують витрати на підтримку інфраструктури й можуть бути використані як у межах окремого закладу, так і на рівні мережі клінік. Особливої актуальності такі рішення набувають у контексті безперервності лікування — коли лікар повинен мати швидкий доступ до повної історії звернень пацієнта та результатів його попередніх прийомів.

Метою даної роботи є розробка вебсистеми електронного обліку пацієнтів та медичних призначень, що забезпечує зручне ведення історії лікувань, створення та редагування записів про прийоми, прикріплення результатів оглядів і документів, а також автоматичне ведення журналу дій користувачів. Система передбачає розмежування ролей між лікарями, адміністраторами та пацієнтами: лікар має можливість керувати всіма своїми записами, а пацієнт отримує персоналізований доступ до власної історії лікування. Важливим аспектом реалізації є захищене зберігання даних і контроль доступу до конфіденційної медичної інформації.

Для досягнення поставленої мети було поставлено наступні завдання:

- 1) проаналізувати наявні медичні інформаційні системи та визначити їхні переваги й недоліки;

2) обрати технології для розробки вебсистеми обліку пацієнтів та обґрунтувати їх використання;

3) спроектувати структуру бази даних для зберігання інформації про пацієнтів, лікування і документи;

4) реалізувати та протестувати вебсистему з ролями лікаря, адміністратора і пацієнта.

Запропонована система реалізована із застосуванням технологій PHP, MySQL та архітектурного підходу MVC, що забезпечує модульність, гнучкість і можливість подальшого розширення функціональності. Було спроектовано інтуїтивно зрозумілий користувацький інтерфейс, який дозволяє лікарю швидко працювати з даними, адміністратору створювати користувачів та переглядати журнал подій, а пацієнту — переглядати свою медичну історію в зручному форматі.

У процесі реалізації також приділено увагу питанням безпеки: передбачено систему авторизації, контроль доступу за ролями та захист персональних даних. Розроблене рішення може бути адаптоване для використання у невеликих клініках, приватних медичних кабінетах або освітніх медичних закладах як демонстраційна система управління записами пацієнтів.

Структура роботи складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі подано постановку задачі та аналіз існуючих засобів для розв'язання проблеми. У другому розділі наведено опис архітектури системи, структури бази даних і вибір технологій. Третій розділ присвячено реалізації вебдодатку, його основним модулям і механізмам безпеки. У четвертому розділі розглянуто сценарії взаємодії користувачів із системою, особливості роботи лікаря, адміністратора, пацієнта та системні вимоги.

1 ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ЗАСОБІВ ДЛЯ РОЗРОБКИ ВЕБСИСТЕМИ ОБЛІКУ ПАЦІЄНТІВ

У сучасних медичних закладах процес ведення обліку пацієнтів, фіксації історії їх лікування та управління медичними призначеннями залишається однією з найважливіших організаційних задач. Незважаючи на широке впровадження інформаційних технологій у сферу охорони здоров'я, багато клінік і приватних медичних установ досі використовують паперові або частково електронні форми обліку. Такий підхід створює низку проблем: ускладнюється пошук інформації про пацієнта, виникають дублювання даних, втрачається історія лікувань, а також підвищується ризик людських помилок при роботі з медичною інформацією.

Тому постає потреба у створенні зручної, безпечної та функціональної вебсистеми, яка забезпечить централізований електронний облік пацієнтів і дозволить лікарям вести детальну історію лікувань, прикріплювати результати обстежень та зберігати всю необхідну інформацію у єдиній базі даних. Така система має бути доступною через браузер, не вимагати складної установки та дозволяти одночасну роботу декількох користувачів із різними рівнями доступу.

1.1 Постановка задачі, мета та завдання розробки

Метою даної роботи є розробка вебсистеми електронного обліку пацієнтів та медичних призначень, яка забезпечує зберігання та структурований облік даних про пацієнтів, ведення історії лікувань із можливістю додавання записів і результатів прийомів, автоматичне фіксування дій користувачів у журналі подій, розмежування доступу між лікарями, адміністраторами та пацієнтами, а також захищене зберігання конфіденційної медичної інформації. Для досягнення поставленої мети в роботі необхідно провести аналіз предметної області та

визначити основні проблеми існуючих систем обліку пацієнтів, сформулювати функціональні та нефункціональні вимоги до вебдодатку, розробити архітектуру системи на основі моделі MVC, спроектувати реляційну базу даних для забезпечення логічної цілісності та зв'язності інформації між пацієнтами, адміністраторами, лікарями, лікуваннями й документами, реалізувати серверну частину мовою PHP з використанням MySQL для зберігання даних, створити інтуїтивно зрозумілий інтерфейс користувача для лікаря, адміністратора та пацієнта, впровадити механізм авторизації, розмежування ролей і контролю доступу до персональних даних, забезпечити можливість перегляду історії лікувань пацієнта у вигляді повного списку з інформацією про дати, скарги, діагнози та результати, а також провести тестування системи для перевірки її функціональності, зручності використання та безпеки [1].

Результатом виконання роботи повинен бути готовий вебдодаток, який дозволяє лікарю керувати базою пацієнтів, створювати записи лікувань, додавати документи з результатами обстежень, а пацієнту — переглядати історію своїх прийомів у зручному форматі. Система має модульну структуру, може бути розгорнута на будь-якому вебсервері та слугує прикладом інтеграції інформаційних технологій у медичну сферу для підвищення ефективності роботи персоналу та якості обслуговування пацієнтів.

1.2 Функціональні та нефункціональні вимоги до системи

Під час створення будь-якої інформаційної системи, особливо такої, що працює з конфіденційними медичними даними, важливим етапом є формування вимог до системи. Вимоги — це сукупність властивостей, характеристик та функцій, яким має відповідати розроблюваний програмний продукт. Вони визначають, що саме повинна робити система, як вона має це виконувати та в яких умовах працювати.

Коректно сформульовані вимоги дозволяють уникнути двозначностей під час розробки, забезпечують чітке розуміння цілей проекту, спрощують тестування готового рішення та підвищують якість кінцевого продукту. Крім того, вимоги є основою для оцінки ефективності, надійності й зручності майбутньої системи.

Зазвичай усі вимоги поділяються на функціональні та нефункціональні. Функціональні вимоги описують конкретні можливості, дії та процеси, які реалізує система наприклад, авторизація користувачів, створення записів лікування, перегляд історії пацієнтів тощо. Нефункціональні вимоги визначають якісні характеристики системи: продуктивність, безпеку, надійність, зручність використання, масштабованість тощо.

Для створення ефективної вебсистеми електронного обліку пацієнтів необхідно визначити вимоги, яким вона має відповідати. Ці вимоги поділяються на функціональні, що описують конкретні можливості системи, і нефункціональні, які визначають її якісні характеристики — продуктивність, безпеку, зручність користування тощо [2].

Нижче наведено функціональні та нефункціональні вимоги до розроблюваної вебсистеми електронного обліку пацієнтів та медичних призначень. Система повинна забезпечувати авторизацію користувачів за допомогою логіна та пароля, після чого користувач отримує доступ до інтерфейсу відповідно до своєї ролі — лікаря або пацієнта. Лікар має можливість створювати, редагувати та видаляти записи лікування, а також створювати нових користувачів і керувати їхніми обліковими записами. Пацієнт, у свою чергу, має доступ лише до перегляду власної історії лікування. Адміністратор створювати користувачів, переглядати журнал подій.

Система повинна зберігати основну інформацію про пацієнтів і забезпечувати пошук за прізвищем або іншими критеріями. Лікар може створювати нові записи прийому пацієнта із зазначенням дати, скарг, діагнозу, опису лікування та статусу. Усі записи повинні зберігатися в базі даних та відображатися у списку лікувань. До кожного лікування передбачено можливість прикріплення результатів

аналізів, висновків, довідок та інших документів із зазначенням опису й зауважень лікаря.[3] Система автоматично фіксує створення, редагування та видалення записів і документів у журналі подій, де вказуються дата, користувач та опис виконаної дії. Пацієнт має можливість переглядати загальний список своїх прийомів із короткою інформацією про кожен запис, а також відкривати деталі лікування та прикріплені документи. Лікар може здійснювати пошук записів за пацієнтом, датою або статусом лікування. Система також повинна підтримувати додавання, редагування та видалення облікових записів, встановлення ролей під час створення користувача, завершення роботи через кнопку «Вихід» та обмеження доступу до внутрішніх сторінок без проходження авторизації.

До нефункціональних вимог належить забезпечення швидкого завантаження сторінок — до двох секунд при стандартному навантаженні, а також оптимізація роботи з базою даних для мінімізації часу виконання запитів. Дані користувачів повинні зберігатися у захищеному вигляді, а паролі — у зашифрованому форматі. Кожен користувач має унікальний логін, а сесії авторизації контролюються сервером. Уся інформація повинна зберігатися в реляційній базі даних MySQL, при цьому у випадку помилок або збоїв дані не повинні втрачатися.

Архітектура системи на основі моделі MVC має забезпечувати можливість подальшого розширення функціоналу, зокрема додавання нових модулів, ролей і звітів. Інтерфейс системи повинен бути інтуїтивно зрозумілим навіть для користувачів без технічних знань, із використанням чітких форм введення, таблиць, кнопок та інформативних повідомлень.[4] Вебсистема повинна коректно працювати у сучасних браузерх і не потребувати встановлення додаткового програмного забезпечення на клієнтському комп'ютері. Код системи має бути структурованим і організованим відповідно до принципів MVC для спрощення підтримки та оновлення. Усі конфігураційні параметри, зокрема налаштування підключення до бази даних та URL-шляхи, повинні зберігатися у централізованому файлі конфігурації.[5] Крім того, користувачі мають отримувати доступ лише до

інформації, що відповідає їхній ролі, а передача чутливих даних повинна здійснюватися через захищене HTTPS-з'єднання.

Таким чином, описані функціональні та нефункціональні вимоги визначають основні можливості та технічні обмеження системи. Вони формують основу для подальшого проєктування архітектури, структури бази даних і логіки роботи всіх модулів вебдодатку.

1.3 Аналіз існуючих медичних інформаційних систем

Сучасні медичні інформаційні системи є важливим елементом цифрової інфраструктури охорони здоров'я. Вони призначені для автоматизації процесів зберігання, обробки й передачі медичних даних, спрощення комунікації між лікарями та пацієнтами, а також підвищення ефективності управління медичними закладами. Аналіз існуючих рішень дозволяє визначити їхні можливості, переваги, недоліки та сформулювати вимоги до власної системи.

На сьогодні існує низка розповсюджених МІС, які частково або повністю реалізують електронний облік пацієнтів, формування історій хвороб і керування призначеннями лікарів. Нижче розглянуто кілька типових прикладів таких систем.

Doctor Eleks — одна з найпоширеніших українських медичних інформаційних систем, що використовується як у приватних клініках, так і в державних медичних закладах.

Основними можливостями системи є створення електронних карт пацієнтів із повною історією звернень, формування графіків прийомів та записів на консультації, інтеграція з лабораторними модулями для отримання результатів аналізів, автоматичне формування звітів і статистичних даних, а також надання доступу лікарям і адміністраторам відповідно до визначених рівнів прав доступу.

Переваги: широкий функціонал, можливість масштабування, підтримка державних стандартів обміну медичними даними.

Недоліки: висока вартість ліцензії, складність налаштування, потреба у спеціальному навчанні персоналу.

Helsi Pro — це онлайн-платформа, що забезпечує взаємодію пацієнтів і лікарів через вебінтерфейс. Система надає можливість онлайн-запису на прийом до лікаря, перегляду історії відвідувань, інтеграції з державною системою eHealth, а також використання електронних рецептів і направлень. До переваг платформи належать доступність через браузер, мобільність та інтеграція з національною медичною системою. Водночас серед недоліків можна виділити обмежені можливості кастомізації під потреби конкретного медичного закладу, а також залежність від центрального сервера та стабільного інтернет-з'єднання.

Medstar — ще одна українська система, орієнтована на ведення електронних медичних карток пацієнтів. Вона забезпечує ведення електронних медичних записів, підтримує модулі лабораторних аналізів і медичних призначень, дозволяє формувати звіти для адміністрації закладу та реалізує різні рівні доступу користувачів. Основними перевагами системи є структуроване зберігання даних і зручна робота з великими обсягами інформації. Недоліком є те, що інтерфейс орієнтований переважно на великі клініки, через що система може бути складною для використання у невеликих приватних медичних кабінетах.

eHealth — державна система, яка забезпечує обмін медичними даними між закладами охорони здоров'я, лікарями та пацієнтами на національному рівні. Система підтримує централізоване зберігання медичних записів, електронні декларації між лікарем і пацієнтом, а також взаємодію з аптеками та страховими компаніями. Переваги: державна підтримка, висока безпека, можливість інтеграції з приватними системами.

Недоліки: складна процедура підключення, обмежений функціонал для локального використання.

Аналіз показує, що більшість наявних систем орієнтовані на великі медичні установи або роботу в межах державної інфраструктури. Такі рішення мають широкий функціонал, але потребують значних ресурсів для впровадження,

налаштування та технічної підтримки. Для невеликих клінік чи приватних кабінетів вони є надмірно складними.

На відміну від цього, розроблена у межах даної роботи вебсистема має іншу концепцію, вона легка, автономна, проста у розгортанні та не потребує складних інтеграцій. Основний акцент зроблено на зручності роботи лікаря з базою пацієнтів, простому додаванні результатів прийомів і забезпеченні захищеного доступу до даних.

Таким чином, проведений аналіз дозволив визначити ключові недоліки існуючих систем і сформулювати вимоги до власного рішення, яке поєднує базову функціональність медичного обліку з простотою використання та гнучкістю налаштування.

1.4 Порівняння з відомими платформами обліку пацієнтів

Після аналізу існуючих медичних інформаційних систем доцільно провести порівняння їхніх ключових характеристик із розроблюваною вебсистемою. Це дозволяє визначити, наскільки нова система відповідає сучасним вимогам, у чому полягають її переваги, а також які функції запозичено або спрощено для підвищення ефективності та доступності.

Порівняння систем проводилося за такими критеріями, як функціональні можливості, зокрема ведення обліку пацієнтів, збереження історії лікування та результатів прийомів, складність упровадження, необхідність спеціального навчання персоналу й вимоги до технічного забезпечення, вартість і доступність системи для медичних закладів, зручність користування інтерфейсом, рівень безпеки та контролю доступу до медичної інформації, а також цільова аудиторія, на яку орієнтоване програмне забезпечення (таблиця 1.1).

Таблиця 1.1 — Порівняння систем

Система	Призначення	Переваги	Недоліки	Складність упровадження	Цільова аудиторія
Doctor Eleks	Комплексна автоматизація медзакладу	Повна історія пацієнта, інтеграція з лабораторіями, статистика	Висока вартість, складне налаштування	Висока	Великі лікарні, клініки
Helsi Pro	Онлайн- запис, eHealth інтеграція	Простий інтерфейс, онлайн-доступ	Обмежена кастомізація, залежність від сервера	Середня	Пацієнти та клініки, що працюють через eHealth
Medstar	Електронні медичні картки, лабораторія	Надійність, структурованість	Невелика гнучкість для приватних кабінетів	Середня	Великі клініки
eHealth	Державна система обліку медичних даних	Безпечна, централізована, інтегрована	Складне підключення, не автономна	Висока	Державні заклади, МОЗ
Розроблена вебсистема	Електронний облік пацієнтів і лікувань	Простота, швидкість, автономність, захист даних	Невеликий набір аналітики	Низька	Приватні кабінети, малі клініки

На відміну від них, розроблена вебсистема електронного обліку пацієнтів та медичних призначень має спрощену архітектуру, не потребує складних інсталяцій і може працювати на звичайному вебсервері. Її структура побудована за принципом MVC, що забезпечує зрозуміле розділення логіки, інтерфейсу та даних [6]. Система не залежить від зовнішніх сервісів і може функціонувати локально або на внутрішньому сервері клініки.

Основний акцент у системі зроблено на практичних задачах лікаря, зокрема на швидкому створенні картки пацієнта, веденні записів лікування, прикріпленні результатів прийомів.

Завдяки цьому система є оптимальним рішенням для невеликих медичних установ, приватних лікарських кабінетів або навчальних цілей. Вона поєднує базові функції обліку з простою адміністрацією, зрозумілим інтерфейсом і безпечним зберіганням медичних даних.

1.5 Визначення ключових переваг та недоліків існуючих рішень

Аналіз існуючих медичних інформаційних систем показав, що сучасний ринок пропонує низку комплексних програмних продуктів для автоматизації роботи медичних закладів. Вони здатні вирішувати широкий спектр задач — від електронного обліку пацієнтів до інтеграції з державними реєстрами, лабораторними модулями та системами звітності. Проте, незважаючи на розвинений функціонал, більшість рішень мають як переваги, так і суттєві обмеження, що ускладнюють їх використання у невеликих або приватних медичних практиках.

До переваг існуючих систем належать повнота функціоналу, централізоване зберігання даних, інтеграція з іншими програмними модулями, високий рівень

безпеки та контролю доступу, а також підтримка державних стандартів у сфері охорони здоров'я.

Сучасні медичні платформи охоплюють майже всі етапи роботи медичного закладу — від реєстрації пацієнтів до формування звітності для державних структур. Більшість систем забезпечують доступ до єдиної бази даних, що дає змогу лікарям швидко переглядати історію хвороби та результати обстежень.

Платформи на кшталт Doctor Eleks або Helsi Pro мають можливість обміну інформацією з лабораторіями, аптеками та страховими компаніями. Для користувачів передбачено різні рівні прав, що дозволяє запобігти несанкціонованому доступу до конфіденційних даних.

Частина систем наприклад, eHealth забезпечує відповідність нормативним вимогам МОЗ та міжнародним стандартам збереження медичних даних.

До недоліків існуючих рішень належать висока вартість впровадження, складність налаштування та використання, залежність від центральних серверів, надмірність функціоналу для невеликих медичних закладів, а також обмежена гнучкість у налаштуванні системи під індивідуальні потреби користувачів.

Більшість потужних медичних систем є комерційними продуктами, вартість ліцензії та технічної підтримки яких є значною для невеликих клінік. Для ефективної роботи з такими системами необхідно проходити навчання або залучати спеціалістів із впровадження. Хмарні рішення потребують постійного інтернет-з'єднання, а у разі технічних збоїв робота системи може бути тимчасово заблокована.

Часто платформи мають великий набір модулів, більшість з яких не використовується у невеликих закладах, але впливають на загальну складність і вартість продукту. Наявні рішення не завжди дозволяють адаптувати систему під специфічні потреби конкретного лікаря чи приватної практики без додаткової розробки.

Порівняння показало, що більшість наявних медичних інформаційних систем орієнтовані на масштабні інфраструктурні рішення (таблиця 1.2). Вони добре

підходять для великих медичних центрів, але виявляються надмірними для малих клінік, приватних кабінетів чи навчальних установ. Це створює нішу для простих, автономних і легких у розгортанні систем, які не потребують великих ресурсів, але забезпечують базову функціональність і захист даних.

Таблиця 1.2 — Аналіз переваг і недоліків

Назва системи	Основні переваги	Основні недоліки
Doctor Eleks	комплексний функціонал для великих клінік, інтеграція з лабораторіями та eHealth, автоматичні звіти та статистика	висока вартість ліцензії, складність налаштування та адміністрування, потребує навчання персоналу
Helsi Pro	онлайн-доступ для лікарів і пацієнтів, ручний запис на прийом, інтеграція з державною системою	обмежена можливість налаштувань, залежність від центрального сервера, не автономна робота
Medstar	структуроване зберігання медичних даних, підтримка звітів та аналітики, надійність у великих установах	недостатня гнучкість для малих клінік, високі технічні вимоги, обмежений інтерфейс
eHealth	державна підтримка та стандарти безпеки, централізована база даних, високий рівень захисту інформації	складне підключення, відсутність локальної автономії, не призначена для приватних кабінетів

Кінець таблиці 1.2

Розроблена вебсистема	простота у використанні та розгортанні, незалежність від зовнішніх сервісів, зрозумілий інтерфейс для лікаря, адміністратора та пацієнта, захищене зберігання даних	відсутність розширених аналітичних модулів, призначена для малих і середніх закладів
------------------------------	--	---

У цьому контексті розроблена вебсистема електронного обліку пацієнтів та медичних призначень покликана поєднати переваги професійних рішень із простотою локального використання. Її ключовими перевагами є незалежність від зовнішніх серверів і стабільна робота навіть у локальному середовищі, простий та зрозумілий інтерфейс без перевантаження зайвими функціями, мінімальні вимоги до апаратного забезпечення, швидке налаштування та можливість адаптації під потреби конкретного користувача, а також безпечне зберігання даних і авторизація користувачів із розмежуванням доступу за ролями.

Таким чином, зроблений аналіз підтверджує доцільність створення власного рішення, яке задовольняє потреби невеликих медичних практик і сприяє цифровізації процесів обліку пацієнтів без залучення складних корпоративних платформ.

1.6 Обґрунтування необхідності створення власної системи

Проведений аналіз існуючих медичних інформаційних систем показав, що попри наявність великої кількості готових рішень, значна частина з них орієнтована на великі медичні заклади з розгалуженою структурою, складною ієрархією користувачів та високими фінансовими можливостями. Такі продукти є

ефективними у масштабних мережах клінік, проте виявляються надмірними для невеликих медичних установ, приватних лікарських кабінетів або навчальних проєктів, де потрібна проста, автономна та легка у використанні система.

Ключовою проблемою існуючих рішень є надмірна складність і вартість впровадження. Для роботи з більшістю комерційних платформ потрібно проходити навчання, здійснювати інтеграцію з державними системами або купувати платні ліцензії. Крім того, хмарні сервіси вимагають постійного підключення до Інтернету, що не завжди зручно чи можливе у невеликих практиках. Це створює потребу в розробці більш гнучкого, доступного та незалежного рішення.

Передумовами для створення власної системи стали необхідність забезпечення локальної автономної роботи, спрощення інтерфейсу та логіки взаємодії користувача із системою, забезпечення базового рівня безпеки та конфіденційності медичних даних, низькі вимоги до технічних ресурсів, а також можливість подальшого розширення функціоналу відповідно до потреб користувачів і медичного закладу.

Багато невеликих клінік і приватних лікарів не мають потреби у централізованому обміні даними, а тому надають перевагу системам, які можуть працювати автономно на локальному сервері або навіть одному комп'ютері.

Більшість готових систем перевантажені функціоналом, що ускладнює використання. Натомість приватним лікарям потрібен мінімалістичний, інтуїтивно зрозумілий інтерфейс, який дозволяє швидко вносити записи й переглядати історію лікувань. Навіть у спрощеній системі важливо реалізувати механізм авторизації користувачів, розмежування ролей і захист персональних даних пацієнтів, що відповідає сучасним стандартам обробки медичної інформації.

Наявні системи часто потребують окремих серверів, великого обсягу оперативної пам'яті чи специфічного програмного забезпечення. Розроблена система повинна бути легкою, швидкою й придатною для запуску на звичайному вебсервері без додаткових витрат.[7] Використання архітектури MVC забезпечує модульність і простоту подальшої модернізації. Це дозволяє поступово додавати

нові функції — наприклад, формування звітів, статистику відвідувань чи аналітику призначень.

Метою створення вебсистеми є розробка сучасного, зручного й захищеного інструменту для електронного обліку пацієнтів, який би поєднував у собі простоту використання, надійність і адаптивність до конкретних потреб лікаря.

На відміну від універсальних корпоративних систем, розроблене рішення реалізує лише ті функції, які безпосередньо необхідні у щоденній роботі медичного працівника, зокрема створення та редагування записів лікування, додавання результатів прийомів, перегляд історії пацієнтів і фіксацію дій користувачів у журналі подій. Система покликана підвищити ефективність роботи лікаря, мінімізувати час на заповнення даних та забезпечити швидкий доступ до повної історії лікувань пацієнтів.[8]

У результаті розробки буде створено вебсистему електронного обліку пацієнтів та медичних призначень, яка працює через браузер без необхідності складного встановлення додаткового програмного забезпечення, забезпечує захищене зберігання медичних даних у базі MySQL, має зрозумілий інтерфейс для лікаря, адміністратора та пацієнта, легко розгортається на будь-якому сервері та підтримує можливість подальшого розширення функціональності завдяки використанню архітектури MVC [9].

Таким чином, створення власної системи є обґрунтованою та доцільною відповіддю на обмеження існуючих рішень. Вона дозволяє забезпечити повний цикл обліку пацієнтів у компактному, автономному та безпечному середовищі, що відповідає потребам сучасної приватної медичної практики.

2 ПРОЄКТУВАННЯ ТА ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ СИСТЕМИ ЕЛЕКТРОННОГО ОБЛІКУ ПАЦІЄНТІВ

Розроблена вебсистема електронного обліку пацієнтів та медичних призначень побудована на основі архітектурної моделі MVC, яка передбачає поділ логіки програми на три незалежні частини: модель, представлення та контролер. Такий підхід дозволяє зробити структуру системи логічно впорядкованою, легко підтримуваною та розширюваною, що є важливим для будь-якого вебдодатку, який може з часом доповнюватися новими модулями.

2.1 Загальна архітектура системи

Архітектура системи має трьохрівневу структуру (рисунок 2.1). На першому рівні знаходиться інтерфейс користувача, який відображається у веббраузері та забезпечує взаємодію з користувачем. Інтерфейс реалізований засобами HTML, CSS і JavaScript, що дозволяє створити інтуїтивно зрозумілу форму взаємодії без потреби у додатковому програмному забезпеченні. Саме тут користувач виконує дії, такі як авторизація, створення записів лікування, перегляд історії пацієнтів або додавання документів.

Другий рівень становить логіка управління, що реалізується через контролери. Контролери відповідають за обробку запитів користувача, перевірку правильності введених даних, виклик необхідних моделей та формування результату, який потім передається у вигляді сторінки для відображення. Вони є своєрідною зв'язувальною ланкою між інтерфейсом користувача та базою даних.

Третій рівень — це модель даних, яка відповідає за роботу з базою даних MySQL. У ній зберігається вся інформація про користувачів, пацієнтів, лікування, документи та журнал дій. Моделі містять функції для створення, оновлення,

видалення та вибірки даних із таблиць. Завдяки використанню цього рівня забезпечується логічна цілісність бази та спрощується подальше обслуговування системи.

Взаємодія між компонентами відбувається у вигляді послідовного процесу. Коли користувач виконує певну дію в браузері, наприклад, відкриває сторінку або натискає кнопку, запит надходить до контролера. Контролер визначає, яка операція має бути виконана, і звертається до відповідної моделі для роботи з базою даних. Після отримання результату контролер передає його до частини представлення, де дані форматуються у вигляді вебсторінки. Користувач отримує оновлену інформацію без необхідності знати про внутрішні процеси, що відбуваються на сервері.

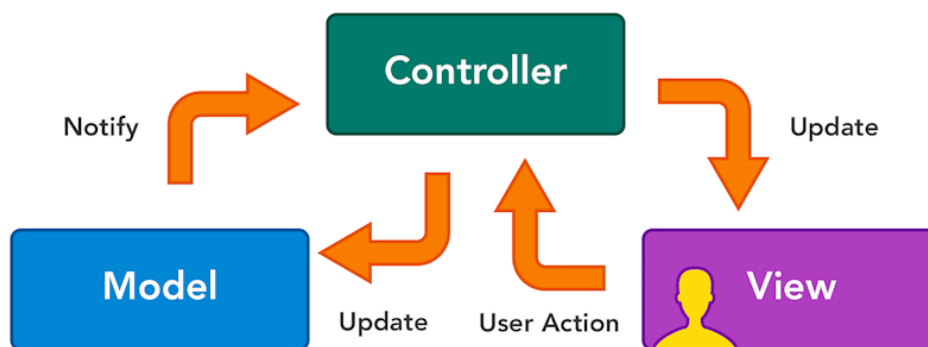


Рисунок 2.1 — MVC архітектура

Система має кілька основних модулів, які взаємодіють між собою. До них належать модуль авторизації, модуль управління пацієнтами, модуль ведення записів лікування, модуль збереження результатів прийомів і документів, а також журнал дій користувачів [10]. Усі ці частини об'єднані єдиною логікою, що забезпечує зручність роботи для лікаря, адміністратора та безпечний доступ для пацієнта.

Важливою особливістю архітектури є реалізація розмежування доступу (рисунок 2.2). У системі передбачено три типи користувачів: лікар, адміністратор і

пацієнт. Лікар має повний доступ до створення карток пацієнта, редагування та видалення записів, тоді як пацієнт може лише переглядати власну історію лікувань. Таке розмежування ролей реалізоване на рівні контролерів і сесій користувача, що забезпечує безпечне використання системи.

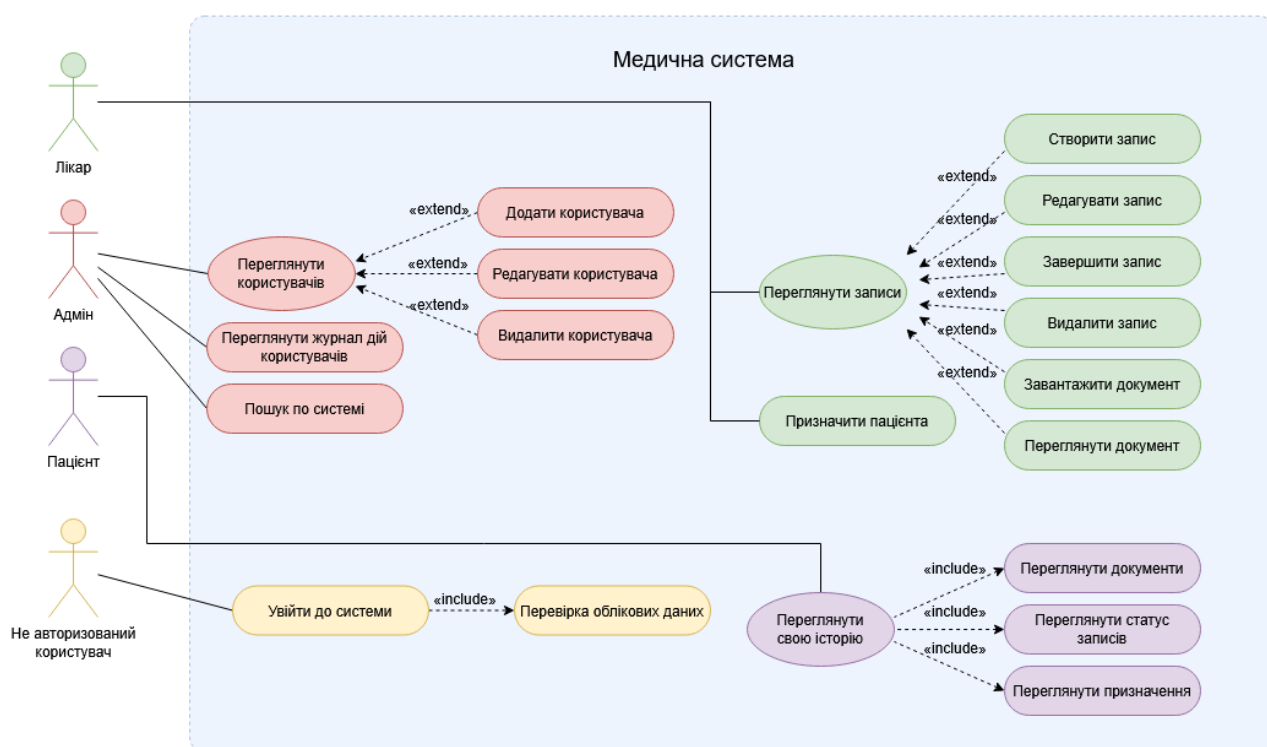


Рисунок 2.2 — Use Case діаграма

Окрім цього, у системі реалізовано механізми захисту даних. Користувачі проходять авторизацію перед входом у систему, а паролі зберігаються у зашифрованому вигляді. Доступ до бази даних із клієнтської частини відсутній, усі операції з інформацією виконуються лише через серверну логіку. Це дозволяє уникнути несанкціонованих змін і втручань у дані пацієнтів.

Завдяки модульності MVC-архітектури система може бути розширена без зміни її основної структури (рисунок 2.3). Наприклад, у майбутньому можна додати модуль аналітики або систему звітності для лікарів. Такий підхід робить проєкт гнучким, зручним для супроводу та придатним до масштабування.

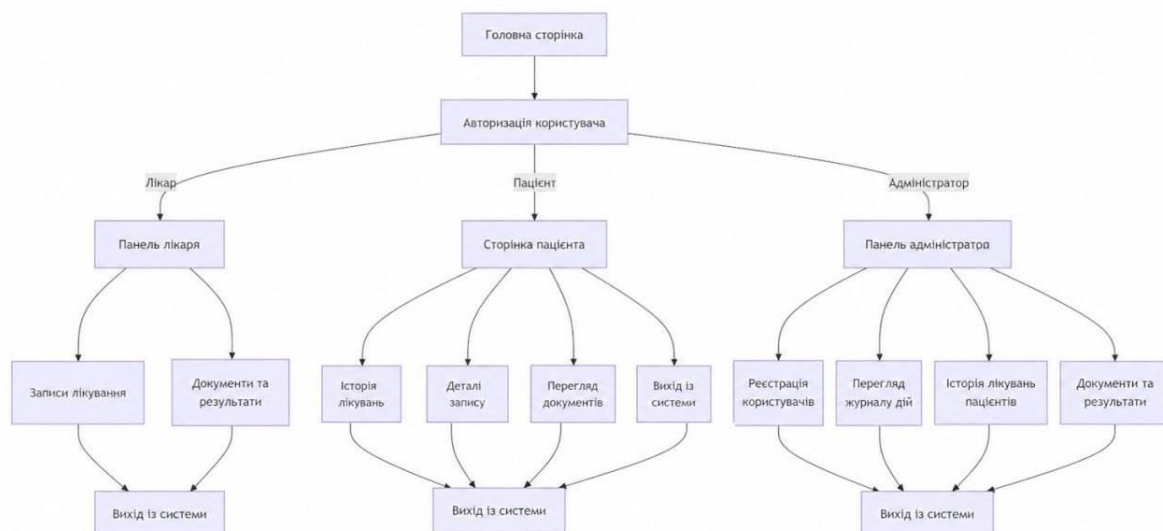


Рисунок 2.3 — Архітектура системи

У межах архітектури системи виділено кілька функціональних модулів. Модуль авторизації забезпечує перевірку логіна та пароля користувача, визначення його ролі та відкриття відповідного інтерфейсу. Модуль управління пацієнтами реалізує створення нових користувачів, редагування даних, пошук і видалення записів. Модуль лікувань дозволяє створювати, переглядати та редагувати записи лікування для кожного пацієнта. Модуль документів забезпечує додавання файлів, зауважень і описів до кожного лікування. Модуль журналу дій автоматично фіксує всі зміни, створення та видалення записів, відображаючи історію активності користувачів.

Система реалізує декілька типів користувачів. Адміністратор відповідає за керування профілями користувачів та налаштуваннями системи. Лікар має повний доступ до функціоналу, пов'язаного з лікуванням пацієнтів, створенням записів і роботою з документами. Пацієнт може переглядати лише власну історію лікувань та пов'язані з нею дані.

Для забезпечення безпеки передбачено серверну перевірку сесій користувачів, шифрування паролів, контроль доступу відповідно до ролей користувачів та захищене зберігання конфіденційної медичної інформації. Завдяки

MVC-архітектурі можливо легко додати нові функції наприклад, звітність або графіки результатів без зміни основної структури коду.

Таким чином, загальна архітектура вебсистеми є трирівневою, побудованою на принципах MVC, що забезпечує незалежність компонентів, простоту супроводу, безпеку обробки даних та можливість подальшого розширення функціональності.

2.2 Вибір платформи та технологій

Під час розробки вебсистеми електронного обліку пацієнтів важливим етапом є вибір технологічного середовища, мови програмування та допоміжних інструментів, що забезпечують стабільну, безпечну та ефективну роботу програмного продукту. Вибір конкретних технологій був здійснений з урахуванням вимог до системи, можливостей її подальшого розширення, простоти розгортання та підтримки.

Основою серверної частини системи є мова програмування PHP, що традиційно використовується для створення динамічних вебдодатків [11]. PHP має широкий набір готових бібліотек, підтримує роботу з базами даних, легко інтегрується з HTML-кодом і може бути розгорнутий на більшості вебсерверів. Однією з переваг цієї мови є її сумісність з великою кількістю систем управління базами даних і підтримка об'єктно-орієнтованого підходу, що дозволяє реалізувати архітектуру MVC[12].

Крім того, PHP характеризується високою швидкістю обробки запитів та невисокими вимогами до серверних ресурсів, що робить його оптимальним вибором для невеликих і середніх проєктів, зокрема для локальних медичних систем [13].

У якості серверного середовища використовується Apache HTTP Server, який забезпечує стабільну роботу вебдодатку та підтримує стандартні механізми маршрутизації запитів. Завдяки простоті налаштування, безкоштовній ліцензії та

високій надійності Apache є одним із найпоширеніших рішень для запуску PHP-додатків. Разом із PHP він входить до складу популярного пакета XAMPP, який застосовувався під час розробки і тестування системи.

Для зберігання даних обрано реляційну систему управління базами даних MySQL. Вона забезпечує швидку обробку запитів, підтримує транзакції, зв'язки між таблицями, а також має зручний синтаксис SQL для створення, оновлення та пошуку інформації. У базі даних MySQL зберігаються всі основні сутності системи: користувачі, пацієнти, записи лікувань, результати прийомів і журнал дій.

MySQL також має високу сумісність із PHP, що дозволяє легко реалізувати обмін даними між моделями і сервером через вбудовані розширення, такі як PDO [14]. Це забезпечує захищеність запитів і стійкість системи до SQL-ін'єкцій.

Архітектурно система побудована за моделлю MVC. Завдяки цьому забезпечується чітке структурування проєкту, спрощується тестування і розширення, а зміни в одному компоненті не впливають на інші. Наприклад, можливо оновити дизайн сторінок без втручання у серверну логіку або змінити структуру бази даних без порушення роботи інтерфейсу [15].

Клієнтська частина системи реалізована засобами HTML5, CSS3 та JavaScript, що дозволяє створити зрозумілий, легкий і швидкий інтерфейс без використання зовнішніх фреймворків. Таке рішення є доцільним, оскільки система орієнтована на простоту використання і не вимагає складних анімацій або динамічного контенту [16]. Усі стилі оформлення створені вручну, без застосування бібліотек на кшталт Bootstrap, що дозволило досягти максимальної легкості та швидкодії інтерфейсу.

Для зберігання файлів документів використовується структура каталогів на сервері. Кожен документ зберігається з унікальним ідентифікатором, що прив'язаний до конкретного запису лікування у базі даних [17]. Це спрощує пошук і гарантує відповідність між фізичним файлом і записом у системі.

Загалом вибір технологій зумовлений потребою створити гнучку, надійну й водночас легку у використанні систему. Використання зв'язки PHP + MySQL

дозволило реалізувати повноцінний динамічний вебдодаток, а архітектура MVC забезпечила логічну організацію коду і можливість подальшого розвитку проєкту[18]. Клієнтська частина, створена на базі стандартних вебтехнологій, гарантує сумісність із усіма сучасними браузерами та стабільну роботу на різних пристроях.

Отже, обрана технологічна платформа відповідає вимогам до функціональності, безпеки, продуктивності та простоти супроводу. Вона забезпечує необхідний баланс між технічною складністю реалізації, швидкістю розробки та зручністю користування, що є ключовими факторами для створення ефективної вебсистеми медичного призначення.

2.3 Проєктування бази даних

Одним із ключових етапів розробки вебсистеми електронного обліку пацієнтів є проєктування бази даних. Від структури бази залежить швидкість роботи програми, узгодженість даних та зручність реалізації логіки застосунку. Проєктування бази даних передбачає визначення основних сутностей, зв'язків між ними, типів полів та обмежень цілісності.

На етапі аналізу вимог було виявлено, що система повинна забезпечувати облік користувачів лікарів, адміністраторів і пацієнтів, зберігання історії лікувань, прикріплених документів, а також журнал усіх дій, що виконуються у системі.

База даних спроектована з дотриманням принципів нормалізації, що дозволяє уникнути дублювання інформації [19]. Кожна таблиця відповідає окремій сутності, а зв'язки між таблицями встановлюються за допомогою зовнішніх ключів (рисунок 2.4). Такий підхід забезпечує логічну цілісність даних та мінімізує ризик помилок при оновленні або видаленні записів.

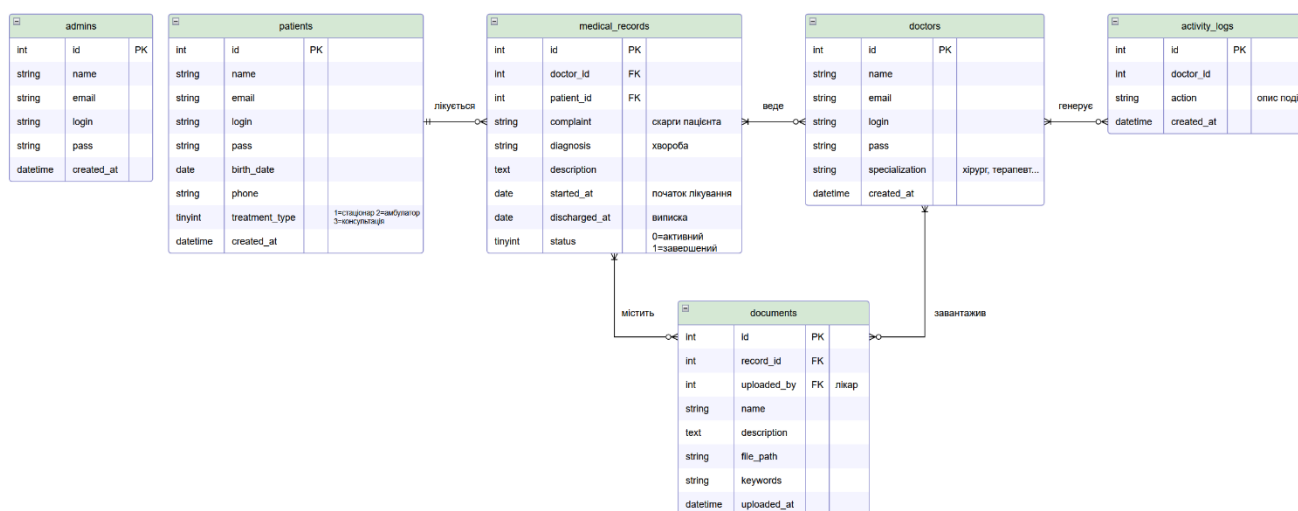


Рисунок 2.4 — Структурна схема бази даних системи

У системі передбачено кілька основних таблиць бази даних. Таблиця *users* містить інформацію про користувачів системи, зокрема логін, пароль та роль користувача. Таблиця *patients* призначена для зберігання даних про пацієнтів, таких як прізвище, ім'я, контактні відомості, дата народження та інша загальна інформація. Таблиця *treatments* містить записи про лікування або прийоми пацієнтів, включно з описом скарг, діагнозом, рекомендаціями та датою створення запису. Таблиця *documents* використовується для зберігання файлів, прикріплених до конкретного лікування або медичного запису. Таблиця *logs* виконує функцію журналу дій, у якому фіксуються події, що відбуваються в системі, зокрема створення, редагування та видалення записів, із зазначенням користувача, дати та опису виконаної дії.

Зв'язки між таблицями побудовані таким чином, щоб кожен пацієнт міг мати кілька записів лікування, кожне лікування — кілька документів, а всі дії користувачів фіксуються в журналі [20]. Усі зовнішні ключі налаштовані з каскадним оновленням або видаленням, що забезпечує узгодженість даних при зміні або видаленні записів.

Структура бази даних дозволяє лікарю швидко отримати доступ до історії

лікувань пацієнта, додавати нові записи або результати прийомів. Така організація даних забезпечує простоту розширення — у майбутньому можна додати нові таблиці, наприклад, для лабораторних результатів або планування прийомів, без порушення існуючої логіки.

2.4 Модель взаємодії компонентів

У процесі роботи вебсистеми електронного обліку пацієнтів усі запити користувачів проходять через чітко визначений ланцюг взаємодії між компонентами серверної логіки (рисунок 2.5).

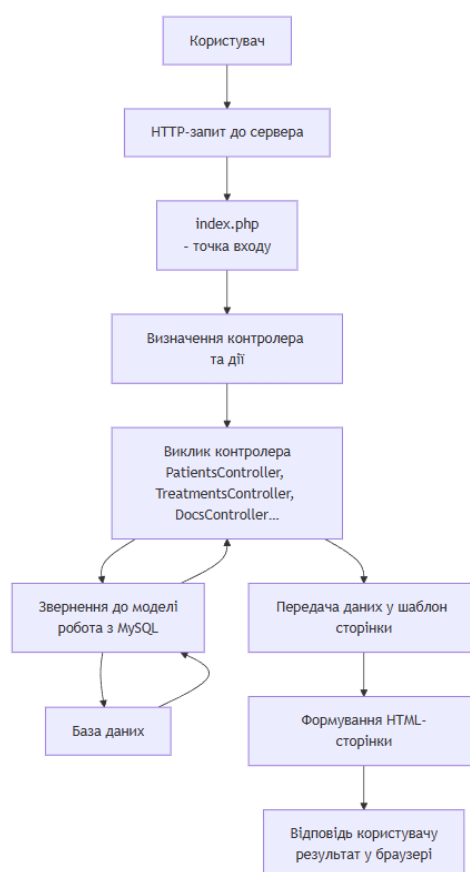


Рисунок 2.5 — Взаємодії компонентів системи

Кожен елемент виконує окрему функцію — приймає запит, обробляє його, звертається до бази даних, отримує результат і формує сторінку для відображення. Такий механізм забезпечує керованість коду, логічну послідовність обробки даних і підвищує безпеку системи.

При відкритті будь-якої сторінки користувач надсилає HTTP-запит до веб-сервера Apache, який перенаправляє його у центральний файл `index.php`. Цей файл виконує роль точки входу в систему — він визначає, який контролер і яку дію потрібно викликати, виходячи з параметрів запиту [21]. Наприклад, якщо користувач відкриває сторінку списку пацієнтів, запит спрямовується до `PatientsController`, який відповідає за роботу з відповідним модулем.

Контролер завантажує потрібні моделі, виконує перевірку прав користувача, і звертається до бази даних через відповідний метод. Наприклад, при додаванні нового лікування контролер викликає функцію `createTreatment()` у моделі, передає їй отримані з форми дані, а модель уже записує цю інформацію в таблицю `treatments`. У випадку перегляду історії пацієнта контролер викликає метод `getTreatmentsByPatientId()` і отримує список усіх записів лікування.

Після отримання даних із бази контролер формує масив або об'єкт з результатами і передає його до шаблону сторінки. Кожна сторінка представлена окремим файлом, який містить HTML-розмітку з динамічними вставками PHP-змінних. Таким чином, користувач бачить актуальну інформацію, сформовану сервером у момент запиту.

Усі дії користувачів фіксуються в таблиці `logs`. Це відбувається автоматично під час виконання ключових операцій, таких як створення нового пацієнта, додавання документа або зміна лікування. Завдяки цьому система зберігає історію активності й дозволяє адміністратору переглядати дії користувачів через інтерфейс «Журнал дій».

Роль контролерів є центральною, вони координують роботу всієї системи. Контролери не взаємодіють безпосередньо між собою, проте користуються спільними допоміжними функціями — наприклад, системою авторизації, класом

для роботи з базою даних або функцією завантаження файлів. Усі компоненти розташовані у відповідних директоріях controllers, models, views, що забезпечує чітку структуру проєкту й полегшує його супровід [22].

Така логіка дозволяє розділити обов'язки між частинами програми, зменшити ймовірність помилок і спростити масштабування системи.

2.5 Засоби розгортання системи на вебсервері

Після завершення етапу розробки вебсистеми електронного обліку пацієнтів необхідно забезпечити її працездатність у реальному середовищі вебсерверу. Розгортання системи є заключним етапом життєвого циклу програмного продукту, під час якого вебдодаток переноситься із локального середовища розробки на віддалений сервер, де ним можуть користуватися лікарі, адміністратори та пацієнти.

У процесі розробки використовувалося локальне середовище XAMPP, яке містить компоненти Apache, PHP і MySQL. Це рішення забезпечувало можливість швидкого тестування всіх модулів системи без підключення до Інтернету. У локальному режимі перевірялися робота контролерів, моделі бази даних, система авторизації користувачів і механізми завантаження документів.

XAMPP виконував роль проміжного середовища для налагодження логіки, структури запитів та перевірки функціональності. Саме у цьому середовищі була створена початкова база даних і відпрацьована модель взаємодії компонентів.

Після завершення тестування систему може бути легко перенесено на віддалений вебсервер, що працює під керуванням операційної системи Linux і має встановлений вебстек LAMP. Ця конфігурація є стандартною для більшості сучасних хостингів і забезпечує стабільну роботу вебдодатків, створених на PHP.

Процес розгортання системи на сервері виконується у декілька етапів. Спочатку на сервер завантажується структура проєкту — каталоги controllers,

models, views, папки з файлами стилів, сценаріїв JavaScript і зображень. Для передачі файлів використовується SFTP-з'єднання або панель керування хостингом.

Далі створюється база даних через панель адміністратора MySQL або за допомогою інструменту phpMyAdmin, після чого імпортується SQL-файл, який містить структуру таблиць users, patients, treatments, documents, logs. На цьому етапі також створюється користувач бази даних і надаються відповідні права доступу.

У конфігураційному файлі системи config.php вказуються параметри підключення до бази даних. Після оновлення конфігурації система готова до запуску. Для перевірки працездатності у браузері відкривається домен або IP-адреса сервера, де автоматично завантажується сторінка авторизації. Успішне завантаження сторінки підтверджує коректну роботу PHP-інтерпретатора, серверної логіки та з'єднання з базою даних.

На рівні вебсервера налаштовано віртуальний хост, який спрямовує всі HTTP-запити до головного файлу index.php. Цей файл є точкою входу у систему й забезпечує маршрутизацію запитів до відповідних контролерів. У разі необхідності система може бути розгорнута у підкаталозі, що дозволяє інтегрувати її у вже існуючий сайт.

Для підвищення безпеки вхід до адміністративної частини обмежено авторизацією, а доступ до системних файлів заборонено на рівні Apache-директив .htaccess. Додатково на сервері активовано SSL-сертифікат, що забезпечує передавання даних між клієнтом і сервером по захищеному протоколу HTTPS. Це особливо важливо при роботі з медичною інформацією.

Таким чином, розгортання вебсистеми відбувається у два етапи — локальне тестування у середовищі XAMPP і подальше перенесення на віддалений сервер із конфігурацією LAMP. Такий підхід забезпечує стабільність, сумісність і безпечну експлуатацію системи в реальному середовищі. У результаті створюється готовий до використання вебдодаток, який може працювати як у межах внутрішньої медичної мережі, так і на публічному сервері з обмеженим доступом.

2.6 Забезпечення безпеки та збереження медичних даних

Збереження конфіденційності та захист персональних медичних даних є одним із ключових завдань при створенні вебсистеми. Оскільки система працює з інформацією, що містить персональні відомості про пацієнтів, до її безпеки висуваються підвищені вимоги. Для цього передбачено комплекс організаційних і технічних заходів, спрямованих на запобігання несанкціонованому доступу, втраті або зміні даних.

Основу безпеки системи становлять три головні складові: контроль доступу користувачів, захист даних під час їх зберігання та передавання, а також резервне копіювання й відновлення інформації. Контроль доступу та авторизація користувачів забезпечують розмежування прав доступу відповідно до ролі користувача та запобігають несанкціонованому доступу до конфіденційних медичних даних [23].

Першим етапом захисту є авторизація (рисунки 2.6). Усі користувачі системи повинні проходити перевірку логіна та пароля, які зберігаються у базі даних у зашифрованому вигляді використовується хешування `password_hash()`.

Після входу система створює сесію, у якій зберігається роль користувача — лікар, адміністратор або пацієнт. Відповідно до ролі визначається доступ до сторінок та функцій: лікар має можливість редагувати записи лікування, адміністратор створювати користувачів, а пацієнт — лише переглядати їх. Доступ до адміністративних сторінок обмежено перевіркою сесій, а спроби неавторизованого переходу перенаправляються на сторінку входу.

На рівні передачі даних використовується протокол HTTPS, який забезпечує шифрування всього трафіку між клієнтом і сервером. Це унеможливорює перехоплення логінів, паролів або інших конфіденційних відомостей.

Усі запити до бази даних виконуються через підготовлені оператори, що запобігає SQL-ін'єкціям. Також контролери перевіряють усі вхідні дані на наявність небезпечних символів або спроб виконати скрипти XSS-захист.

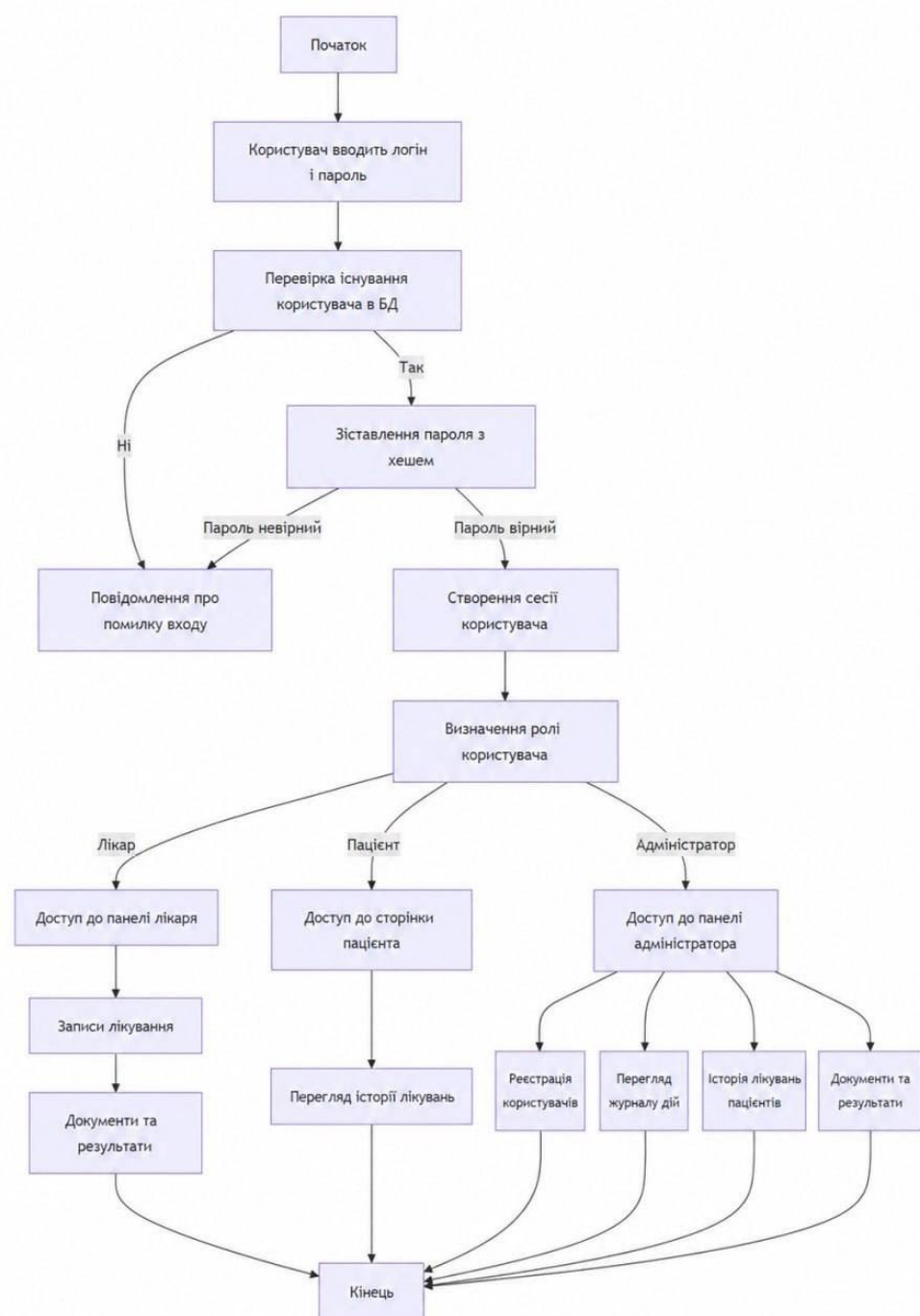


Рисунок 2.6 — Алгоритм авторизації та контролю доступу

Для збереження даних застосовується резервне копіювання бази даних (рисунок 2.7). Копії створюються автоматично за розкладом або вручну адміністратором системи. У разі збоїв або втрати інформації можна виконати швидке відновлення з останнього резервного файлу.



Рисунок 2.7 — Механізм збереження та захисту даних

Усі завантажені документи зберігаються на сервері у спеціальній директорії з обмеженим доступом, а їх шляхи зберігаються у базі даних. Це гарантує відповідність між записом лікування та його файлами.

Таким чином, система реалізує комплексний підхід до безпеки даних. Контроль доступу гарантує, що кожен користувач має лише дозволені права, а серверна логіка запобігає несанкціонованим діям. Використання HTTPS і підготовлених запитів мінімізує ризик перехоплення або підміни інформації, тоді як резервне копіювання гарантує можливість відновлення даних у разі технічних несправностей.

У результаті система забезпечує надійне зберігання, обробку та передачу персональних медичних даних, відповідаючи сучасним вимогам до безпечних вебрішень.

3 РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ ЕЛЕКТРОННОГО ОБЛІКУ ПАЦІЄНТІВ ТА МЕДИЧНИХ ПРИЗНАЧЕНЬ

Структура вебсистеми побудована на основі власної реалізації архітектурного підходу MVC, який забезпечує розділення коду на логічні шари та полегшує подальше розширення і супровід.

3.1 Структура проєкту та логіка взаємодії модулів

У проєкті кожен компонент відповідає за окрему частину функціональності системи (рисунок 3.1). Контролери виступають у ролі посередників між користувачем та внутрішньою логікою системи. Вони аналізують вхідні HTTP-запити, виконують первинну валідацію даних та ініціюють відповідні операції, викликаючи необхідні методи моделей. Після завершення обробки контролер визначає, який саме вигляд (View) повинен бути повернутий користувачеві як відповідь.

Моделі уособлюють шар бізнес-логіки та роботу з даними. Вони інкапсулюють структуру об'єктів системи та правила їхньої обробки. Моделі безпосередньо взаємодіють із шаром доступу до даних, забезпечуючи цілісність інформації та її коректне збереження чи отримання. Важливо, що модель залишається незалежною від представлення, що дозволяє використовувати одну й ту саму логіку для різних інтерфейсів.

Види відповідають за формування графічного інтерфейсу користувача. На основі отриманих від контролерів даних, види генерують HTML-розмітку, яка транслюється в браузер. Цей рівень фокусується виключно на відображенні інформації та збиранні подій користувача, не містячи складної обчислювальної логіки.

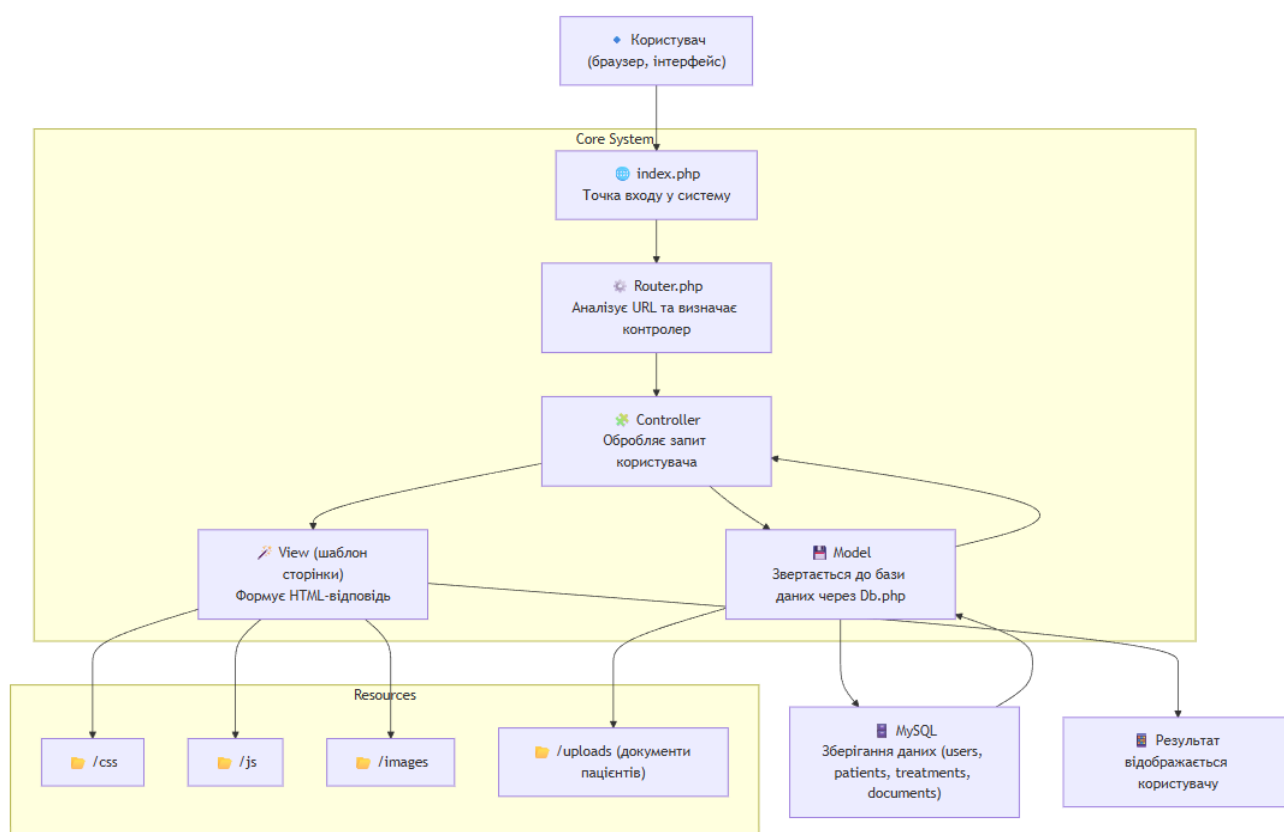


Рисунок 3.1 — Логічна структура файлів

Окрім основних компонентів, у системі реалізовано допоміжні модулі. Клас Db.php відповідає за підключення до бази даних, конфігураційний файл config.php містить основні параметри налаштування системи, а маршрутизатор Router.php визначає, який контролер необхідно викликати для обробки запиту користувача.

Також у структурі проєкту передбачено окремі папки з ресурсами, що містять файли CSS, JavaScript та зображення, необхідні для оформлення інтерфейсу та забезпечення коректної роботи вебдодатку.

У структурі проєкту головним є файл index.php, який виступає точкою входу (рисунок 3.2). Усі HTTP-запити надходять саме до нього. Усередині цього файлу ініціалізується маршрутизатор, який аналізує URL-адресу та визначає, який контролер і метод потрібно викликати.

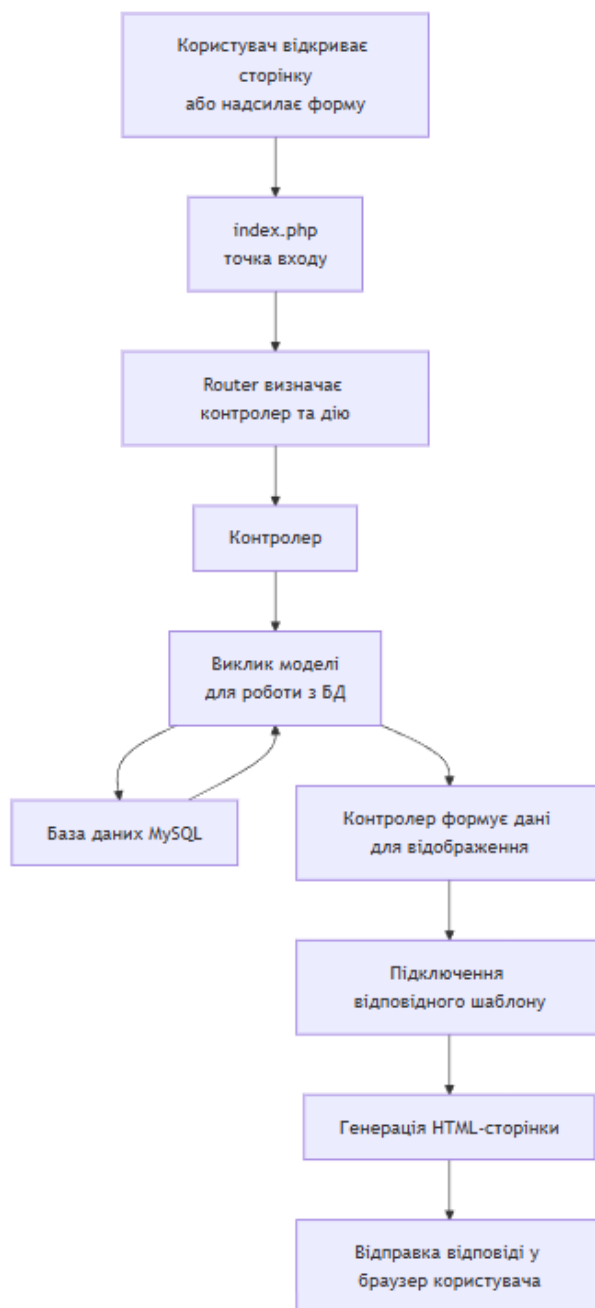


Рисунок 3.2 — Логіка виконання запиту користувача

Контролер отримує дані з моделі, передає їх у відповідний шаблон, і той формує HTML-сторінку для користувача. Уся логіка взаємодії модулів реалізована через внутрішній маршрутизатор. Це дозволяє уникати дублювання коду та підтримувати чисту структуру проєкту.

Коли лікар створює новий запис лікування, запит із форми надходить у

TreatmentsController, який викликає метод `create()` моделі `Treatment`. Модель зберігає дані у таблиці `treatments`, після чого контролер передає результат у представлення `treatments/success.php`, де користувач бачить підтвердження про успішне створення запису.

Кожен модуль функціонує незалежно, але взаємодіє через чітко визначені інтерфейси. Це робить систему гнучкою, стабільною та придатною до подальшого масштабування.

3.2 Реалізація серверної частини

Серверна частина вебсистеми виконує ключові функції, пов'язані з обробкою запитів користувачів, роботою з базою даних, контролем доступу та формуванням динамічного контенту.

Розробка серверної логіки здійснювалася мовою PHP, що дозволяє ефективно взаємодіяти з реляційною базою даних MySQL та формувати сторінки у реальному часі на основі користувацьких дій.

Основа серверної частини становить файл `index.php`, який виконує роль точки входу до системи. Усі запити від користувачів спрямовуються через вебсервер Apache до цього файлу, після чого ініціалізується головний маршрутизатор `Router.php`.

Маршрутизатор аналізує URL-адресу, визначає, який контролер і яку дію необхідно викликати, та передає відповідні параметри. Завдяки цьому забезпечується розділення логіки обробки різних запитів — наприклад, перегляд списку пацієнтів, створення нового лікування чи завантаження документів.

Контролери виступають центральною частиною серверної логіки. Вони приймають запити, перевіряють права доступу користувача, звертаються до моделей для виконання запитів до бази даних і передають результати у шаблони для відображення.

Кожен контролер спеціалізується на певній сутності: PatientsController відповідає за дані пацієнтів, TreatmentsController — за лікування, DocsController — за документи, UsersController — за управління користувачами.

Моделі містять функції, що виконують конкретні SQL-запити вибірка, оновлення, вставка, видалення записів. Для підключення до бази даних використовується допоміжний клас Db.php, у якому реалізовано підключення через розширення PDO. Такий підхід забезпечує безпеку та унеможливорює SQL-ін'єкції.

Усі запити до бази даних виконуються у межах підготовлених операторів, а отримані результати передаються контролеру у вигляді масиву. Контролер потім формує масив даних і передає його у відповідний шаблон, де ці дані динамічно вставляються в HTML. Нижче наведено фрагмент коду (рисунок 3.3), який демонструє принцип обробки запиту на сервері — від маршрутизації до відображення сторінки.

```
<?php
require_once 'core/Router.php';
require_once 'core/Db.php';
require_once 'config/config.php';

session_start();

$router = new Router();
$router->run();

class Router {
    public function run() {
        $uri = trim($_SERVER['REQUEST_URI'], '/');
        $segments = explode('/', $uri);

        $controllerName = ucfirst($segments[0] ?? 'users') . 'Controller';
        $actionName = $segments[1] ?? 'login';

        $controllerFile = 'controllers/' . $controllerName . '.php';

        if (file_exists($controllerFile)) {
            require_once $controllerFile;
            $controller = new $controllerName();
            if (method_exists($controller, $actionName)) {
                $controller->$actionName();
                return;
            }
        }

        include 'views/templates/404.php';
    }
}
```

Рисунок 3.3 — Фрагмент коду серверної частини

Також серверна частина реалізує механізм авторизації користувачів. Після входу в систему створюється сесія, у якій зберігаються дані про користувача та його роль. Це дозволяє обмежити доступ до сторінок і забезпечити розмежування функцій між лікарем, адміністратором і пацієнтом. Модуль працює синхронно з журналом подій, будь-яка зміна або оновлення даних дає новий запис.

3.3 Реалізація бази даних MySQL

Для забезпечення надійного зберігання інформації про пацієнтів, адміністраторів і лікарів, записи лікувань та супровідні документи у системі розроблено реляційну базу даних на основі MySQL.

База даних реалізує зв'язки між сутностями «користувач», «пацієнт», «лікування» та «документ» за моделлю «один-до-багатьох», що дозволяє гнучко управляти інформацією та швидко виконувати запити [24, 25].

Основна ідея проєктування бази полягала у створенні простої, але розширюваної структури з урахуванням майбутніх можливостей — додавання нових типів користувачів, модулів аналітики чи планування візитів [26].

База даних системи відповідає принципам нормалізації третьої нормальної форми, що дозволяє уникнути дублювання даних та підвищує узгодженість і цілісність записів. У реалізації використано декілька основних таблиць. Таблиця `users` містить інформацію про користувачів системи, зокрема логін, пароль та роль користувача. Таблиця `patients` призначена для зберігання особових даних пацієнтів. Таблиця `treatments` містить записи лікувань, включаючи скарги, опис, рекомендації, дату створення запису, а також інформацію про лікаря, адміністратора та пацієнта. Таблиця `documents` використовується для зберігання завантажених файлів, пов'язаних із конкретним лікуванням. Таблиця `logs` реалізує журнал дій користувачів, у якому фіксуються операції створення, редагування та перегляду записів у системі.

Кожна таблиця має первинний ключ `id` та зовнішні ключі для зв'язку з іншими сутностями (рисуюнок 3.4). Поля, що містять текстові дані, зберігаються у форматі `TEXT`, `VARCHAR`, `DATETIME`.

```
CREATE DATABASE medical_system CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE medical_system;

-- Таблиця користувачів
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    login VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('doctor', 'patient', 'admin') DEFAULT 'doctor',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Таблиця пацієнтів
CREATE TABLE patients (
    id INT AUTO_INCREMENT PRIMARY KEY,
    surname VARCHAR(100) NOT NULL,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    birth_date DATE,
    notes TEXT
);

-- Таблиця лікувань
CREATE TABLE treatments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT NOT NULL,
    doctor_id INT NOT NULL,
    complaints TEXT,
    diagnosis TEXT,
    description TEXT,
    status VARCHAR(50) DEFAULT 'active',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (patient_id) REFERENCES patients(id) ON DELETE CASCADE,
    FOREIGN KEY (doctor_id) REFERENCES users(id) ON DELETE SET NULL
);
```

Рисуюнок 3.4 — Фрагмент створення бази даних

Для забезпечення узгодженості даних між таблицями використано зовнішні ключі `FOREIGN KEY` з каскадним оновленням і видаленням.

3.4 Реалізація користувацького інтерфейсу

Користувацький інтерфейс вебсистеми електронного обліку пацієнтів розроблено із дотриманням принципів простоти, інтуїтивності та швидкого доступу до основних функцій.

Оскільки система має три типи користувачів — лікаря, адміністратора та пацієнта, інтерфейс для кожного адаптований до його потреб і ролі в роботі з даними.

Розробка інтерфейсу виконана засобами HTML5, CSS3 та JavaScript, без використання зовнішніх фреймворків типу Bootstrap. Це дозволило досягти мінімального навантаження на сторінки й забезпечити швидке завантаження навіть на слабких пристроях [27].

Кожна сторінка представлена окремим файлом у директорії `views`, що відповідає певному модулю системи `views/patients/list.php`, `views/treatments/view.php`.

Основні елементи інтерфейсу складаються з кількох компонентів. Шапка сторінки містить логотип, назву системи та кнопку виходу з облікового запису. Основна частина інтерфейсу використовується для відображення таблиць, форм і вмісту сторінок системи. Панель навігації забезпечує швидкий перехід між основними розділами вебдодатку. Форми введення даних застосовуються для створення, редагування та збереження інформації в системі.

Після авторизації лікар потрапляє на головну сторінку системи, де відображається список пацієнтів із короткою інформацією: прізвище, ім'я, номер телефону, дата народження.

Після переходу на сторінку конкретного пацієнта лікар може створити новий запис лікування, додати документи, внести результати.

Дані відображаються у вигляді таблиць із чергуванням кольорів рядків для покращення читабельності. Форми створення та редагування побудовані на основі стандартних елементів `<input>`, `<textarea>`, `<select>` (рисуюнок 3.5).

Пацієнт після входу до системи бачить лише власну історію лікувань — список прийомів, дати, короткі описи, а також можливість перегляду завантажених документів.

```
<div class="pop-up">
  <div class="center">
    <div class="top">
      <p class="name">Документ</p>
      <div class="close">
        
      </div>
    </div>
  </div>

  <form class="data" method="POST" action="/doc/file" enctype="multipart/form-data">
    <input class="inp-id" type="text" name="id" hidden>
    <input class="inp-proj-id" type="text" name="proj_id" hidden value="<?=(int)$ _GET['id']?>">

    <p class="label">Назва</p>
    <input class="full inp-name" type="text" name="name" required>
```

Рисунок 3.5 — Фрагмент коду інтерфейсу

Для візуального оформлення застосовується власний CSS-файл. Стили побудовані на нейтральних кольорах, адаптованих до медичної тематики — відтінки синього та сірого.

Таким чином, користувацький інтерфейс системи побудований з урахуванням ролей користувачів, забезпечує інтуїтивну навігацію та простоту використання. Завдяки використанню стандартних вебтехнологій система залишається легкою, швидкою й сумісною з усіма сучасними браузерами.

3.5 Модуль керування користувачами

Модуль керування користувачами (рисунок 3.6) відповідає за створення, авторизацію, редагування та видалення користувачів системи. Він є базовим компонентом серверної логіки, оскільки визначає, хто саме має доступ до даних і які операції користувач може виконувати.

Модуль реалізовано на рівні контролера UsersController.php, та представлень. Після входу користувачів до системи створюється сесія PHP, у якій зберігається їхній ідентифікатор `user_id` та роль `role`.

```
class UsersController {

    // Стопінка входу
    public function login() {
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $login = trim($_POST['login']);
            $password = trim($_POST['password']);

            $user = User::getByLogin($login);

            if ($user && password_verify($password, $user['password'])) {
                $_SESSION['user_id'] = $user['id'];
                $_SESSION['role'] = $user['role'];

                if ($user['role'] === 'doctor') {
                    header("Location: /patients/list");
                } else {
                    header("Location: /treatments/my");
                }
                exit;
            } else {
                $error = "Невірний логін або пароль!";
            }
        }
        include 'views/users/login.php';
    }

    // Реєстрація нового користувача (для адміністратора)
    public function register() {
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $login = trim($_POST['login']);
            $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
            $role = $_POST['role'];

            User::create($login, $password, $role);
            header("Location: /users/list");
            exit;
        }
        include 'views/users/register.php';
    }

    // Вихід із системи
    public function logout() {
        session_destroy();
        header("Location: /users/login");
        exit;
    }
}
```

Рисунок 3.6 — Фрагмент коду контролер користувачів

Ця інформація використовується для контролю доступу до сторінок системи: лікар має повні права доступу до функціоналу, тоді як пацієнт отримує обмежений доступ лише до власних записів і даних. Основними функціями модуля є

авторизація користувачів, що включає перевірку логіна та пароля і створення сесії, реєстрація нових користувачів, вихід із системи шляхом очищення сесії та перенаправлення на сторінку входу, а також перевірка доступу для визначення ролі користувача та надання відповідних прав доступу.

3.6 Модуль створення та перегляду записів лікування

Модуль створення та перегляду записів лікування є центральним елементом системи, адже саме він реалізує основну функцію — ведення історії лікувань пацієнтів.

Його завданням є забезпечення можливості адміністратору створювати, переглядати та редагувати записи, що стосуються кожного пацієнта, а також лікарю прикріплювати додаткові документи до цих записів.

Після авторизації лікар переходить до списку своїх пацієнтів, де для кожного доступна дія «Історія лікувань». При відкритті цієї сторінки викликається метод контролера `TreatmentsController::list($patient_id)`, який звертається до моделі `Treatment` і отримує всі записи, пов'язані з конкретним пацієнтом. Результат вибірки передається у представлення `views/treatments/list.php`, де відображається у вигляді таблиці з датами, коротким описом і кнопками перегляду.

Лікар може додати новий запис, натиснувши кнопку «Новий запис», після чого відкривається форма створення лікування `/treatments/add/{patient_id}` (рисунок 3.7). Після натискання кнопки «Зберегти» форма надсилає POST-запит до методу `TreatmentsController::create()`, який перевіряє коректність даних, додає запис у базу таблиці `treatments` і автоматично фіксує цю дію у журналі `logs`.

Усі лікування прив'язані до конкретного пацієнта `patient_id` і лікаря `doctor_id`, що дозволяє легко фільтрувати історію за кожним користувачем. Для пацієнта при вході система відображає лише ті записи, які належать до його власного ідентифікатора, забезпечуючи конфіденційність. Для зручності використовується

спільний шаблон сторінки, що автоматично підтягує заголовок, меню та контентну частину. Завдяки такій структурі код легко розширювати — наприклад, можна додати функції пошуку за датою чи фільтрацію за статусом лікування.

Метод `create()` перевіряє, чи отримано дані методом POST. Дані з форми `complaints`, `diagnosis`, `description` проходять базову валідацію. Викликається метод моделі `Treatment::create()` для збереження запису у таблиці `treatments`.

```
public function create() {
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $patientId = $_POST['patient_id'];
        $doctorId = $_SESSION['user_id'];
        $complaints = trim($_POST['complaints']);
        $diagnosis = trim($_POST['diagnosis']);
        $description = trim($_POST['description']);

        // Валідація
        if (empty($complaints) || empty($description)) {
            $error = "Будь ласка, заповніть усі обов'язкові поля.";
            include 'views/treatments/add.php';
            return;
        }

        // Збереження у базу
        $result = Treatment::create($patientId, $doctorId, $complaints, $diagnosis, $description);

        if ($result) {
            // Додаємо запис у журнал дій
            $db = Db::getConnection();
            $stmt = $db->prepare("INSERT INTO logs (user_id, action, description) VALUES (?, 'create_treatment', ?)");
            $stmt->execute([$doctorId, "Створено новий запис лікування для пацієнта ID: $patientId"]);

            header("Location: /treatments/list/$patientId");
            exit;
        } else {
            $error = "Помилка при створенні запису.";
        }
    }

    include 'views/treatments/add.php';
}
```

Рисунок 3.7 — Фрагмент коду створення нового запису лікування

Якщо операція успішна — створюється відповідний запис у таблиці `logs`. Після цього користувач перенаправляється на сторінку історії лікувань відповідного пацієнта.

Такий підхід гарантує простоту розширення: можна легко додати функціонал редагування, видалення і прикріплення документів до лікування без зміни основної логіки.

3.7 Модуль додавання результатів прийому та документів

Модуль додавання результатів прийому та документів є важливим доповненням до основного функціоналу системи, оскільки дозволяє зберігати файли медичних досліджень, знімки, довідки, аналізи і протоколи лікування, що безпосередньо пов'язані з конкретним записом лікування.

Цей модуль забезпечує можливість лікарю прикріплювати результати прийому до електронної картки пацієнта, а також гарантує зручний і швидкий доступ до них у будь-який момент. Після створення і відкриття конкретного запису лікування лікар має змогу додати до нього документи за допомогою окремої форми.

На сторінці перегляду лікування відображається розділ «Документи прийому», де подано список уже завантажених файлів із коротким описом і датою додавання, а також кнопка «Додати документ». Натискаючи цю кнопку, користувач переходить до простої форми, у якій можна вибрати файл на комп'ютері, вказати короткий коментар і тип документа.

Механізм роботи модуля складається з кількох послідовних етапів:

1) завантаження файлу на сервер — після підтвердження форми файл передається через метод POST і зберігається у директорії `uploads/medical_docs/`;

2) система автоматично генерує унікальну назву файлу для запобігання конфліктам і зберігає його з розширенням, що відповідає початковому формату;

4) запис метаданих у базу даних — у таблицю `documents` додається новий рядок, що містить шлях до файлу, його початкову назву, коментар користувача, дату та ідентифікатор лікування `treatment_id`;

5) прив'язка до запису лікування — документ логічно пов'язаний з конкретним прийомом, тому відображається у списку лише для відповідного пацієнта та лікаря.

Перегляд документів — у режимі перегляду лікування всі прикріплені файли відображаються списком із можливістю завантаження і відкриття у новій вкладці браузера. Система контролює типи дозволених файлів та їх розмір, запобігаючи

завантаженню небезпечних і надто великих даних. Для цього передбачена перевірка MIME-типу файлу та обмеження максимальної ваги. У разі невідповідності система виводить користувачу повідомлення про помилку.

Усі файли зберігаються у спеціально відведеній директорії `uploads/medical_docs/`, до якої доступ іззовні обмежено за допомогою налаштувань сервера `.htaccess`. Це унеможливлює несанкціонований перегляд медичних документів без авторизації.

Відображення документів на сторінці лікування реалізовано у вигляді таблиці. Пацієнт, увійшовши до системи, може переглядати лише ті документи, які належать до його власних прийомів, тоді як лікар має повний доступ до всіх файлів своїх пацієнтів. Реалізація цього модуля дозволила створити єдине цифрове сховище результатів прийомів, у якому всі документи зберігаються впорядковано та доступні без необхідності використання паперових карток. Це суттєво підвищує ефективність ведення медичної документації, спрощує обмін інформацією між лікарями та покращує якість медичного обслуговування. Завдяки цьому вебсистема виконує повноцінну роль електронної медичної картки пацієнта.

3.8 Модуль журналу дій

Модуль журналу дій є важливою складовою серверної частини системи, оскільки забезпечує контроль усіх змін і дій користувачів у процесі роботи з електронною базою пацієнтів. Його головна мета — забезпечити прозорість, відстежуваність і безпеку даних, а також створити інструмент для аналізу активності користувачів у системі.

Кожна дія, що виконується лікарем у системі, автоматично фіксується у спеціальній таблиці бази даних `logs`. До таких дій належать авторизація користувача, створення, редагування та видалення пацієнтів, додавання нових записів лікування, прикріплення документів, а також оновлення даних

користувачів. Кожен запис у журналі містить ідентифікатор користувача, який виконав дію, назву виконаної операції, короткий опис із зазначенням об'єкта, до якого застосовувалася дія, а також дату та час її виконання.

Записи створюються автоматично під час виконання відповідних операцій у контролерах. Коли лікар додає новий запис лікування, після успішного збереження даних у таблиці treatments система одразу створює рядок у таблиці logs. Це дає змогу не лише вести історію змін, а й контролювати коректність дій користувачів.

На стороні інтерфейсу для адміністратора передбачено сторінку «Журнал дій», де дані відображаються у вигляді таблиці. Записи виводяться у хронологічному порядку — найновіші зверху. Для зручності реалізовано пошук за логіном користувача і типом дії, що дозволяє швидко знаходити потрібні події.

Функціональність журналу дій особливо важлива з точки зору аудиту та безпеки, адже вона дозволяє відстежити, хто і коли змінював інформацію, швидко знайти джерело помилки і неправильної дії, перевірити активність лікарів. Крім того, ведення журналу дій допомагає у виявленні несанкціонованих втручань у систему. Завдяки цьому адміністратор має повну картину функціонування системи в реальному часі.

Отже, модуль журналу дій є системою моніторингу внутрішньої активності, яка не тільки підвищує безпеку, але й слугує важливим адміністративним інструментом. Він дає змогу оперативно контролювати роботу медичних працівників, аналізувати ефективність їхньої взаємодії із системою та забезпечує повну простежуваність усіх процесів.

3.9 Тестування та перевірка працездатності

Після завершення етапів проєктування та реалізації вебсистеми електронного обліку пацієнтів було проведено повне тестування для перевірки її стабільності, коректності роботи основних модулів і зручності використання.

Метою тестування було переконатися, що всі компоненти системи взаємодіють належним чином, а користувачі можуть виконувати свої функції без помилок і збоїв. Перевірка працездатності системи здійснювалася методом функціонального тестування — шляхом покрокового проходження усіх сценаріїв роботи користувачів. Також виконано інтеграційне тестування (таблиця 3.1), під час якого перевірялися зв'язки між окремими модулями: авторизацією, базою даних, журналом дій, документами та модулями лікування.

Таблиця 3.1 — Перевірка модулів

№	Етап перевірки	Результат
1	Реєстрація нового користувача	Виконано успішно, дані зберігаються в базі
2	Авторизація користувача (лікар / пацієнт / адміністратор)	Коректне перенаправлення відповідно до ролі
3	Створення нового пацієнта	Запис додається, відображається у списку
4	Створення нового лікування	Дані зберігаються в таблиці treatments
5	Додавання документів до лікування	Файли успішно завантажуються в /uploads/medical_docs/
6	Перегляд історії лікувань пацієнта	Інформація відображається коректно
7	Перевірка прав доступу пацієнта	Пацієнт бачить лише власні записи
8	Ведення журналу дій	Усі операції фіксуються в таблиці logs
9	Вихід із системи	Сесія знищується, перенаправлення на сторінку входу

У процесі тестування використовувалися як тестові облікові записи, так і реальні дані пацієнтів, що дало змогу оцінити повний життєвий цикл даних — від створення користувача до додавання лікування й документів.

Усі перевірки проводилися у тестовому середовищі з використанням вебсервера XAMPP Apache + MySQL + PHP. Додатково перевірено сумісність системи у браузерях Google Chrome, Mozilla Firefox, Microsoft Edge — усі функції працювали коректно.

Під час тестування було перевірено, що система не допускає завантаження файлів заборонених типів, не приймає порожні або некоректно заповнені поля у формах, коректно обробляє помилки введення користувачем та автоматично виводить повідомлення про результати виконаних операцій.

У ході тестування система продемонструвала стабільну роботу без критичних помилок. Усі функції, передбачені технічним завданням, виконуються у повному обсязі. Створення користувачів, пацієнтів, записів лікування, документів та журналів дій працює коректно. Перевірка доступу за ролями реалізована успішно, зокрема пацієнт не має можливості переглядати записи інших користувачів. Усі дані коректно вносяться до бази даних без втрат, а зв'язки між таблицями підтримуються за допомогою зовнішніх ключів.

Проведене тестування довело, що вебсистема електронного обліку пацієнтів повністю відповідає поставленим вимогам, є функціонально завершеною, стабільною та готовою до реальної експлуатації. Усі компоненти системи взаємодіють узгоджено, а користувачі можуть ефективно виконувати свої завдання без помилок. Система забезпечує високу швидкість, захищеність даних та зручність у користуванні.

4 СЦЕНАРІЙ ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ ВЕБСИСТЕМИ

Для стабільної роботи вебсистеми електронного обліку пацієнтів і медичних призначень визначено мінімальні та рекомендовані вимоги до серверного й клієнтського середовищ.

4.1 Системні вимоги

Ці параметри забезпечують належну швидкодію, безперебійність обробки даних та коректне відображення інтерфейсу користувача у веббраузері.

Для розміщення системи необхідно використовувати вебсервер, що підтримує виконання скриптів мовою PHP та роботу з базою даних MySQL. Мінімальні вимоги до серверної частини включають операційну систему Linux Ubuntu 20.04 або Windows 10/11, вебсервер Apache 2.4 або Nginx версії 1.18 і вище, PHP версії 8 або новішої, MySQL 5.7 і вище, не менше 1 ГБ оперативної пам'яті, одне ядро процесора та щонайменше 500 МБ вільного місця на диску для зберігання бази даних і завантажених документів.

Рекомендованими параметрами є використання операційної системи Linux Ubuntu Server, PHP 8.2 із підтримкою розширень PDO, OpenSSL та GD, MySQL 8.0, від 2 до 4 ГБ оперативної пам'яті, двох ядер процесора, а також SSL-сертифіката для забезпечення захищеного HTTPS-з'єднання [28]. Серверна частина системи може бути розгорнута локально за допомогою XAMPP або Laragon під час розробки, а також на віддаленому хостингу для реальної експлуатації [29].

Користувачі отримують доступ до системи через звичайний веббраузер, тому встановлення додаткового програмного забезпечення не потрібне. Мінімальні вимоги до клієнтської частини включають будь-який сучасний браузер із

підтримкою HTML5, CSS3 та JavaScript, роздільну здатність екрана не менше 1366×768 та активне інтернет-з'єднання зі швидкістю не менше 1 Мбіт/с. Рекомендовано використовувати Google Chrome або Mozilla Firefox останніх версій, екран із роздільною здатністю Full HD 1920×1080 та стабільне інтернет-з'єднання зі швидкістю від 5 Мбіт/с.

До загальних вимог належать наявність користувача MySQL із правами SELECT, INSERT, UPDATE та DELETE, права на запис для директорії uploads/medical_docs/ на рівні chmod 755 або вище, а також використання базового антивірусного захисту та регулярного резервного копіювання бази даних [30].

Система не потребує потужних обчислювальних ресурсів і може бути розгорнута навіть на бюджетному віртуальному. Вона сумісна з усіма основними веббраузерами, що забезпечує універсальний доступ як для лікарів, адміністраторів, так і для пацієнтів.

4.2 Сценарії роботи користувачів

У системі реалізовано три основних типи користувачів — лікар і пацієнт, адміністратор. Кожен має власний набір можливостей і різний інтерфейс, оптимізований для своїх задач.

Авторизація здійснюється через єдину форму входу (рисунок 4.1), після чого користувач автоматично перенаправляється у свій персональний кабінет. Після запуску вебзастосунку користувач бачить форму авторизації, у якій необхідно ввести логін і пароль.

У разі успішного входу система перевіряє рівень доступу user_level і перенаправляє користувача на відповідну сторінку. Лікар потрапляє до панелі керування пацієнтами[30]. Пацієнт потрапляє до особистого кабінету з історією лікування.

Логін

Пароль

Увійти

Рисунок 4.1 — Форма авторизації

Після авторизації лікар бачить головну сторінку з переліком усіх пацієнтів, записаних на прийом (рисунок 4.2). Інтерфейс системи містить кілька основних розділів. Розділ «Перегляд історії пацієнта» забезпечує швидкий доступ до попередніх прийомів і медичних записів пацієнта.

Редагувати Вигалити Пошук Створити Дмитро Шевченко

ПІБ ПАЦІЄНТА	ЛІКАР	ПОЧАТОК	ВИПУСКА	СТАТУС
☐ Іван Олексійченко	Дмитро Шевченко	12-05-2026	24-05-2026	В процесі
☐ Батринчук Надія	Дмитро Шевченко	03-05-2026	02-05-2026	Завершено

Запис Вихід

Рисунок 4.2 — Перегляд поточних записів

Розділ «Додати результати прийому і документ» дозволяє прикріплювати до лікування різні файли, зокрема результати МРТ, аналізи, фотографії та інші медичні документи.

Розділ «Додати запис лікування» призначений для створення нового прийому

із зазначенням скарг пацієнта, дати візиту, діагнозу та опису лікування (рисунок 4.3)

Новий запис

Початок лікування
дд.мм.рррр

Виписка
дд.мм.рррр

Скарги пацієнта

Хвороба

Опис

Лікар
Дмитро Шевченко

Пацієнт

Статус

Зберегти

Рисунок 4.3 — Форма для створення нового запису

Кожен прийом містить короткий опис скарг, дати лікування, статус і посилання для відкриття повної інформації (рисунок 4.4).

Редагувати	Виділити	Пошук	Створити	Дмитро Шевченко
НАЗВА	ЗАУВАЖЕННЯ	ОПИС		
☐ Біохімічний аналіз (цукор та холестерин)	Рівень глюкози в нормі. Показники холестерину на верхній межі; рекомендовано корекцію раціону харчування	Перевірка рівня глюкози натще та ліпідного профілю пацієнта.	➔	🔍

Рисунок 4.4 — Інформація про прийом

Для кожного запису лікування лікар може додати опис результатів прийому і прикріпити медичний документ (рисунок 4.5). Підтримуються різні формати файлів — PDF, JPG, PNG. Після додавання документ автоматично з’являється в особистому кабінеті пацієнта.

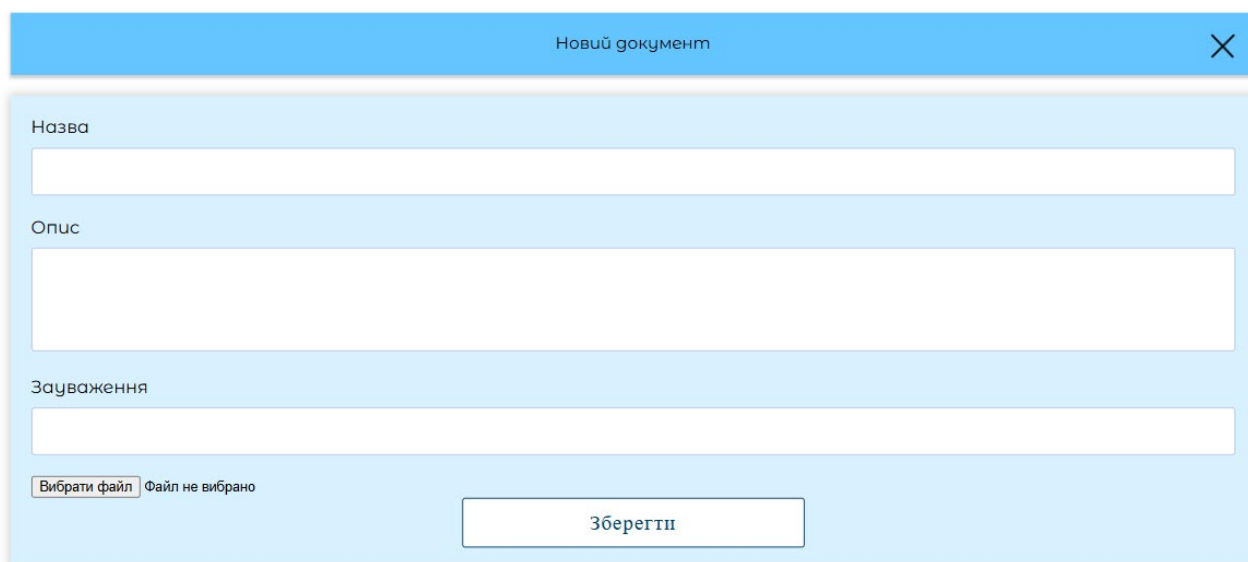


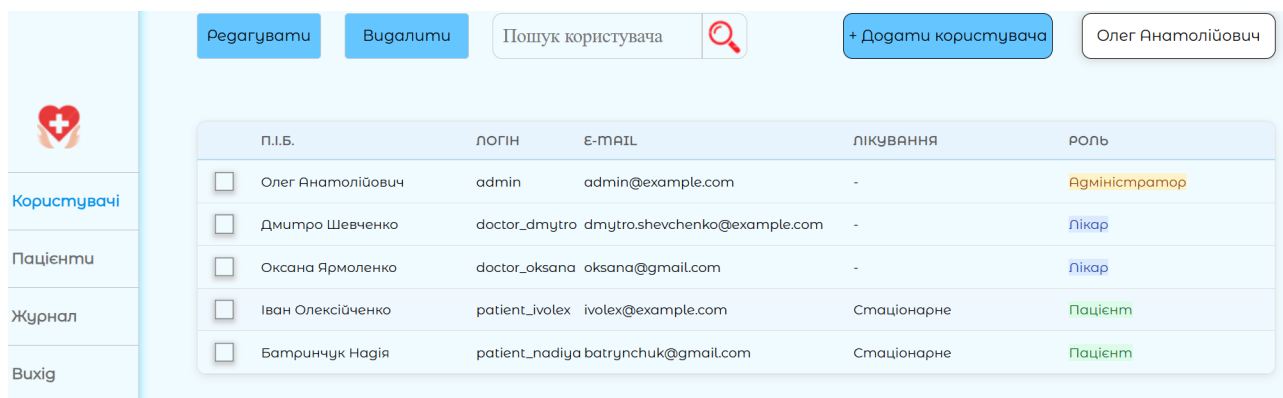
Рисунок 4.5 — Форма внесення даних про прийом

Розроблена вебсистема є простою у використанні та інтуїтивно зрозумілою. Кожен сценарій передбачає мінімальну кількість дій для виконання основних завдань від створення пацієнта до ведення повної історії лікування й додавання результатів прийомів.

4.3 Робота з ролями

У розробленій вебсистемі реалізовано механізм рольового доступу, який визначає, які дії користувач може виконувати після авторизації. Ролі поділяються на три основні рівні — лікар та пацієнт, адміністратор (рисунок 4.6). Відповідно до

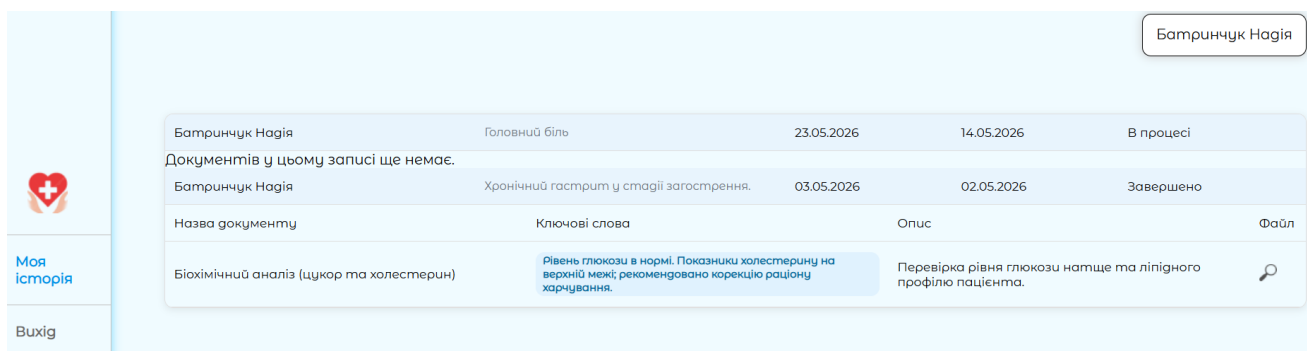
рівня доступу `user_level`, система автоматично обмежує і розширює можливості користувача.



	П.І.Б.	ЛОГІН	Е-МАІЛ	ЛІКУВАННЯ	РОЛЬ
☐	Олег Анатолійович	admin	admin@example.com	-	Адміністратор
☐	Дмитро Шевченко	doctor_dmytro	dmytro.shevchenko@example.com	-	Лікар
☐	Оксана Ярмоленко	doctor_oksana	oksana@gmail.com	-	Лікар
☐	Іван Олексійченко	patient_ivolex	ivolex@example.com	Стационарне	Пацієнт
☐	Батринчук Надія	patient_nadiya	batrynychuk@gmail.com	Стационарне	Пацієнт

Рисунок 4.6 — Список всіх користувачів

Адмін має можливість створити нового пацієнта через окрему форму. Після збереження пацієнт отримує можливість входу до системи. У разі успішного входу пацієнт може переглядати інформацію про свої записи і прикріплені документи (рисунок 4.7).



Батринчук Надія				
Батринчук Надія	Головний біль	23.05.2026	14.05.2026	В процесі
Документів у цьому записі ще немає.				
Батринчук Надія	Хронічний гастрит у стадії загострення.	03.05.2026	02.05.2026	Завершено
Назва документу	Ключові слова	Опис	Файл	
Біохімічний аналіз (цукор та холестерин)	Рівень глюкози в нормі. Показники холестерину на верхній межі; рекомендовано корекцію раціону харчування.	Перевірка рівня глюкози натще та ліпідного профілю пацієнта.		

Рисунок 4.7 — Можливості пацієнта при вході

У разі успішної авторизації під роллю лікаря, користувач може переглядати всіх пацієнтів назначених цьому лікарю у вигляді блокового списку (рисунок 4.8).

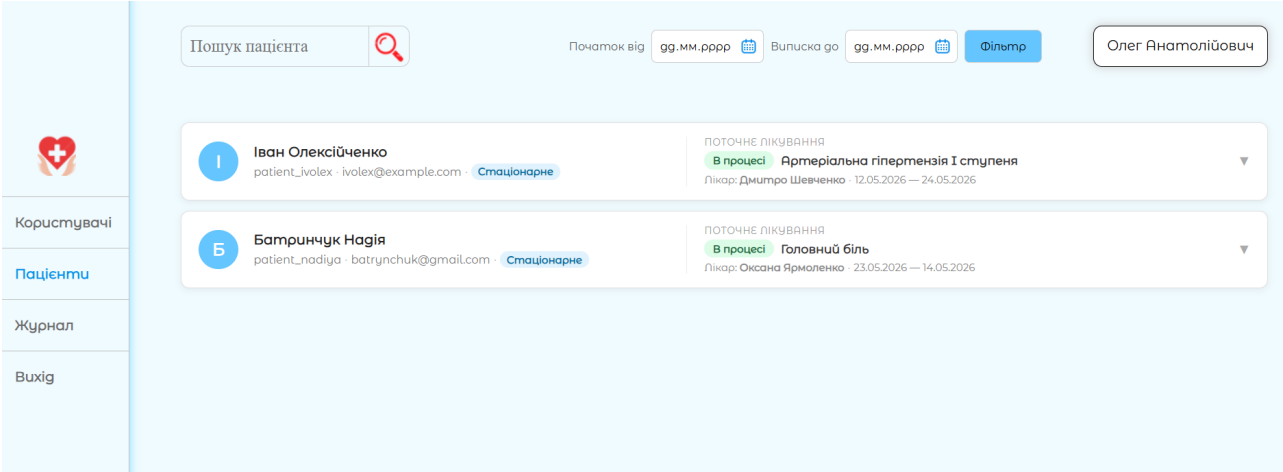


Рисунок 4.8 — Список всіх пацієнтів

Після натискання на певного пацієнта відображається детальна картка з інформацією для забезпечення зручного доступу до історії лікування (рисунок 4.9).

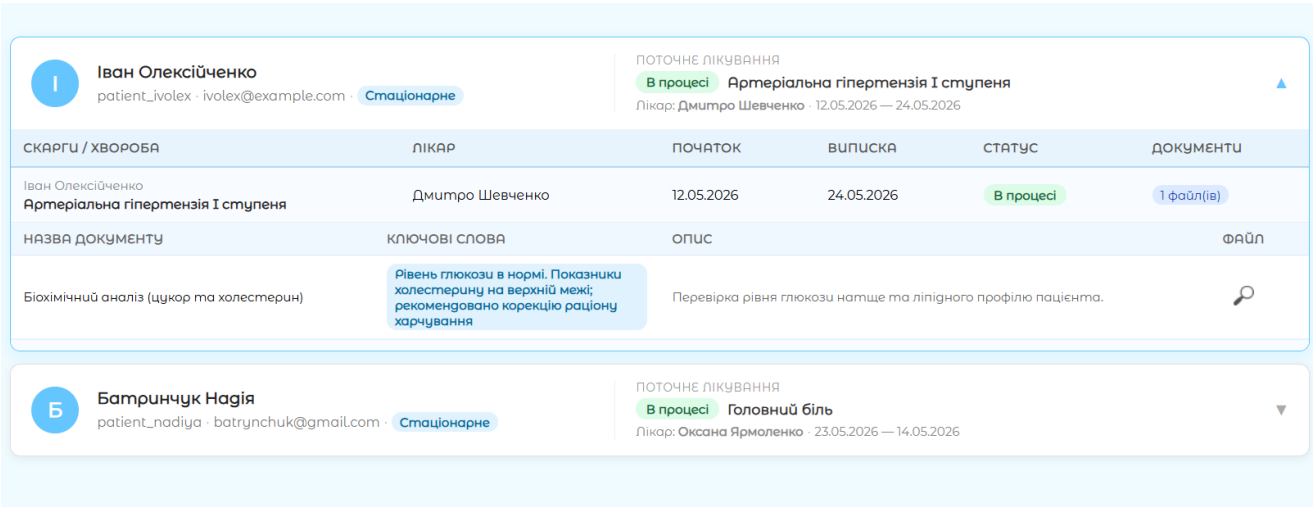


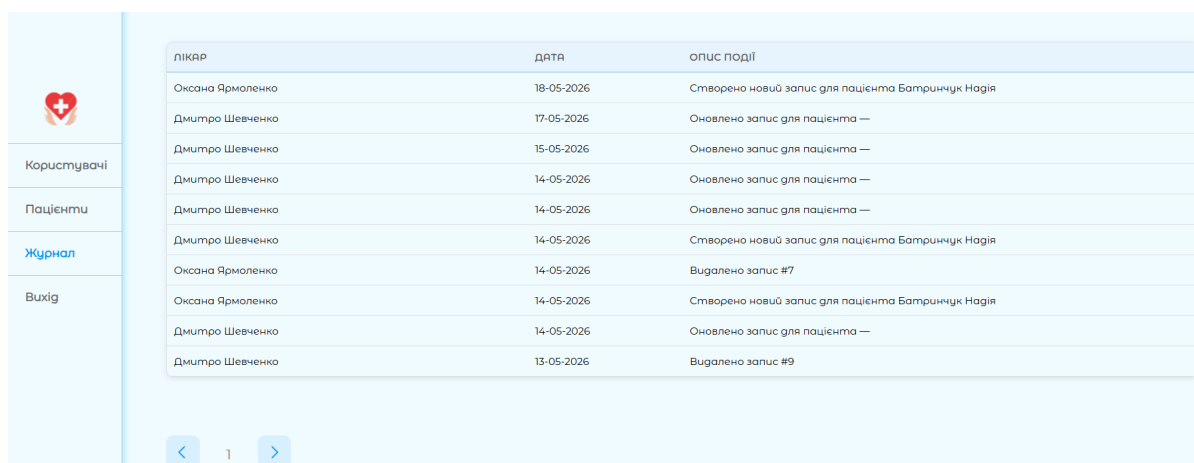
Рисунок 4.9 — Детальніше про пацієнта

Така організація дозволяє лікарю швидко перемикатися між персональними даними, статусом стаціонарного перебування та журналом діагнозів, а інтегрований модуль керування документацією з функцією виділення ключових клінічних висновків мінімізує час на аналіз результатів лабораторних досліджень.

4.4 Журнал подій

Журнал подій є важливою частиною системи, яка забезпечує прозорість, контроль дій користувачів і спрощує аудит змін у базі даних. Він дозволяє відстежувати, хто саме виконував певні дії, коли вони відбулися і що саме було змінено (рисунок 4.10). Цей модуль допомагає адміністраторам контролювати активність у системі та запобігати випадковим і несанкціонованим змінам даних.

Інформація журналу зберігається в окремій таблиці logs бази даних, що забезпечує просту структуру та швидкий пошук за датою або користувачем. Записи формуються автоматично під час виконання дій користувачами. Це означає, що після кожного створення лікування, завантаження документа чи реєстрації нового користувача система самостійно додає новий рядок у таблицю logs.



ЛІКАР	ДАТА	ОПИС ПОДІЇ
Оксана Ярмоленко	18-05-2026	Створено новий запис для пацієнта Батринчук Надія
Дмитро Шевченко	17-05-2026	Оновлено запис для пацієнта —
Дмитро Шевченко	15-05-2026	Оновлено запис для пацієнта —
Дмитро Шевченко	14-05-2026	Оновлено запис для пацієнта —
Дмитро Шевченко	14-05-2026	Оновлено запис для пацієнта —
Дмитро Шевченко	14-05-2026	Створено новий запис для пацієнта Батринчук Надія
Оксана Ярмоленко	14-05-2026	Вигалено запис #7
Оксана Ярмоленко	14-05-2026	Створено новий запис для пацієнта Батринчук Надія
Дмитро Шевченко	14-05-2026	Оновлено запис для пацієнта —
Дмитро Шевченко	13-05-2026	Вигалено запис #9

Рисунок 4.10 — Журнал подій

На сторінці журналу дій адміністратор може переглядати останні події у вигляді таблиці, відсортованої за датою. Інтерфейс дозволяє швидко зрозуміти, хто, коли і яку саме дію виконував. Модуль журналу подій забезпечує контроль, відповідальність і безпеку у роботі вебсистеми. Додатково реалізована можливість пошуку по журналу (рисунок 4.11).

Створено



Олег Анатолійович

ЛІКАР	ДАТА	ОПИС ПОДІЇ
Дмитро Шевченко	11-05-2026	Створено новий запис для пацієнта Іван Олексійченко
Оксана Ярмоленко	13-05-2026	Створено новий запис для пацієнта Батринчук Надія
Оксана Ярмоленко	13-05-2026	Створено новий запис для пацієнта Іван Олексійченко
Дмитро Шевченко	13-05-2026	Створено новий запис для пацієнта Іван Олексійченко
Дмитро Шевченко	13-05-2026	Створено новий запис для пацієнта Батринчук Надія
Оксана Ярмоленко	14-05-2026	Створено новий запис для пацієнта Батринчук Надія
Дмитро Шевченко	14-05-2026	Створено новий запис для пацієнта Батринчук Надія

Рисунок 4.11 — Результат пошуку

Він дозволяє адміністраторам мати повну історію змін, підвищуючи довіру до системи та знижуючи ризик помилок. Таким чином, журнал є не лише технічним інструментом, а й важливим елементом управління якістю роботи медичної установи.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено повнофункціональну вебсистему електронного обліку пацієнтів та медичних призначень, призначену для автоматизації основних процесів взаємодії між лікарем, адміністратором і пацієнтом у медичному закладі.

1. Проведено аналіз сучасних медичних інформаційних платформ, у результаті якого виявлено їхні переваги та недоліки. Визначено, що більшість існуючих рішень є надлишково складними, що обґрунтувало актуальність створення спеціалізованої, інтуїтивно зрозумілої системи для приватних клінік та невеликих медичних закладів.

2. Обрано та обґрунтовано технології для реалізації системи, а саме: серверна частина розроблена на мові PHP з використанням архітектурного шаблону MVC, а база даних реалізована за допомогою MySQL. Використання мінімалістичного підходу до інтерфейсу без сторонніх фреймворків забезпечило високу продуктивність та легкість супроводження коду.

3. Спроектовано та реалізовано реляційну базу даних, яка забезпечує структуроване зберігання персональної інформації про пацієнтів, історії їхнього лікування, медичних призначень та супутньої документації, що гарантує цілісність даних та їхню швидку вибірку.

4. Розроблено вебсистему з розмежуванням прав доступу за ролями «адміністратор», «лікар» та «пацієнт». Впроваджено модуль журналу подій для контролю за діями користувачів та забезпечено можливість розгортання рішення як на локальному, так і на віддаленому сервері. Тестування підтвердило повну відповідність системи заявленому функціоналу та її готовність до впровадження у медичну практику.

Система підтримує можливість розгортання як на локальному сервері, так і на віддаленому хостингу, що робить її універсальною та придатною до практичного впровадження.

Система може бути використана в невеликих приватних клініках, медичних кабінетах або як навчальна платформа для демонстрації принципів створення безпечних рішень у сфері охорони здоров'я. У перспективі функціонал платформи може бути розширений шляхом інтеграції модулів аналітики та систем автоматичних сповіщень пацієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Upper Saddle River, USA : *Prentice Hall*, 2017. 432 p.
2. Joseph I. Software Architect's Handbook: Become a Successful Software Architect by Implementing Effective Architecture Concepts. Birmingham, UK : *Packt Publishing*, 2018. 594 p.
3. Батринчук Н. Вебсистема електронного обліку пацієнтів та медичних призначень. Scientific Method: Reality and Future Trends of Researching : *Proceedings of the VIII International Scientific and Theoretical Conference*, м. Монреаль, 29 трав. 2026 р. Монреаль, Канада, 2026. С. 158–160. URL: <https://doi.org/10.36074/scientia-29.05.2026>.
4. Ullman L. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide. Berkeley, USA : *Peachpit Press*, 2017. 696 p.
5. Welling L., Thomson L. PHP and MySQL Web Development, 5th Edition. Boston, USA : *Addison-Wesley*, 2016. 928 p.
6. Duckett J. HTML & CSS: Design and Build Websites. Indianapolis, USA : *Wiley Publishing*, 2011. 490 p.
7. Keith J., Andrew J. DOM Scripting: Web Design with JavaScript and the Document Object Model. New York, USA : *Apress*, 2010. 480 p.
8. Lerdorf R., Tatro K., MacIntyre P. Programming PHP, 4th Edition. Sebastopol, USA : *O'Reilly Media*, 2020. 550 p.
9. McArthur J. Laravel: Up & Running. A Framework for Building Modern PHP Apps. Sebastopol, USA : *O'Reilly Media*, 2019. 480 p.
10. Stephens R. Beginning MySQL Database Design and Optimization. Berkeley, USA : *Apress*, 2019. 712 p.
11. Beighley L. Head First PHP & MySQL. Sebastopol, USA : *O'Reilly Media*, 2017. 812 p.

12. Meloni J.C. PHP, MySQL & JavaScript All in One. Indianapolis, USA : *Sams Publishing*, 2022. 880 p.
13. Tiwari A. Building a Web Application with PHP and MySQL: A Step-by-Step Guide. Birmingham, UK : *Packt Publishing*, 2020. 352 p.
14. Kizza J. Guide to Computer Network Security, 5th ed. New York, USA : *Springer*, 2020. 568 p.
15. Stallings W. Network Security Essentials: Applications and Standards. London, UK : *Pearson Education*, 2023. 448 p.
16. Baeza-Yates R., Ribeiro-Neto B. Modern Information Retrieval. Boston, USA : *Addison-Wesley*, 2011. 944 p.
17. ISO/IEC 27001:2022. Information Security, Cybersecurity and Privacy Protection — Information Security Management Systems — Requirements. International Organization for Standardization, Geneva, 2022.
18. Guide to Privacy and Security of Electronic Health Information. *HealthIT*. URL: <https://www.healthit.gov/resource-library> (date of access: 18.05.2025).
19. Web Security Guidelines. *Mozilla Developer Network*. URL: <https://developer.mozilla.org/en-US/docs/Web/Security> (date of access: 11.11.2025).
20. Official PHP Documentation. *PHP*. URL: <https://www.php.net/docs.php> (date of access: 11.11.2025).
21. MySQL 8.0 Developer Guide. *MySQL*. URL: <https://dev.mysql.com/doc> (date of access: 11.11.2025).
22. OWASP Top 10 Web Application Security Risks. *OWASP Foundation*. URL: <https://owasp.org/www-project-top-ten> (date of access: 11.11.2025).
23. Frontend Toolkit. *Bootstrap Documentation*. URL: <https://getbootstrap.com/docs/> (date of access: 11.11.2025).
24. Simple ORM for PHP. *RedBeanPHP*. URL: <https://redbeanphp.com> (date of access: 11.11.2025).
25. Icon Library for Web Applications. *FontAwesome*. URL: <https://fontawesome.com/> (date of access: 11.11.2025).

26. Web Content Accessibility Guidelines (WCAG) 2.2. *W3C*. URL: <https://www.w3.org/WAI/standards-guidelines/wcag/> (date of access: 11.11.2025).
27. Anemos Web System Repository. *GitHub*. URL: <https://github.com> (date of access: 11.11.2025).
28. Discussions on PHP MVC Application Design Patterns. *Stack Overflow*. URL: <https://stackoverflow.com/questions/tagged/php-mvc> (date of access: 11.11.2025).
29. How to Deploy a PHP and MySQL Application on Ubuntu. *DigitalOcean*. URL: <https://www.digitalocean.com/community/tutorials> (date of access: 11.11.2025).
30. eHealth Network — Interoperability and Data Protection in Medical Systems. *European Commission*. URL: <https://health.ec.europa.eu/ehealth> (date of access: 11.11.2025).

ДОДАТОК А

Вебсистема електронного обліку пацієнтів та медичних призначень

Апробація результатів

НТУУ «КПІ ім. Ігоря Сікорського»

Аркушів 3

Київ — 2026

Батринчук Надія 

здобувач вищої освіти кафедри Цифрових технологій в енергетиці
*Національний технічний університет України «Київський політехнічний
 інститут імені Ігоря Сікорського», Україна*

Науковий керівник: Рудик Володимир Іванович 

асистент кафедри Цифрових технологій в енергетиці
*Національний технічний університет України «Київський політехнічний
 інститут імені Ігоря Сікорського», Україна*

ВЕБСИСТЕМА ЕЛЕКТРОННОГО ОБЛІКУ ПАЦІЄНТІВ ТА МЕДИЧНИХ ПРИЗНАЧЕНЬ

Сучасний розвиток цифрових технологій у сфері охорони здоров'я сприяє активному впровадженню медичних інформаційних систем для автоматизації процесів обліку пацієнтів, ведення медичної документації та зберігання історії лікування. Використання електронних систем дозволяє підвищити ефективність роботи медичних працівників, мінімізувати кількість помилок під час роботи з документацією та забезпечити швидкий доступ до медичної інформації [1]. Незважаючи на це, значна кількість невеликих клінік і приватних медичних кабінетів продовжують використовувати паперові журнали або прості локальні електронні файли, що створює труднощі під час пошуку інформації, призводить до дублювання записів і ускладнює контроль історії лікування пацієнтів.

У межах дослідження було проведено аналіз сучасних медичних інформаційних систем, що використовуються для електронного обліку пацієнтів та медичних призначень. Однією з найбільш поширених систем є Doctor Eleks, яка орієнтована на комплексну автоматизацію діяльності медичних закладів. Система забезпечує ведення електронних карток пацієнтів, формування графіків прийому, інтеграцію з лабораторними сервісами та створення статистичної звітності. Основною перевагою Doctor Eleks є широкий функціонал і підтримка великої кількості медичних процесів. Водночас система характеризується високою складністю впровадження, потребує значних ресурсів для налаштування та навчання персоналу, а також має високу вартість використання, що робить її менш доступною для невеликих медичних установ [2].

Іншим поширеним рішенням є Helsi Pro — веборієнтована система, що забезпечує онлайн-запис пацієнтів, електронну взаємодію між лікарем і пацієнтом та інтеграцію з державною системою eHealth. Перевагами Helsi Pro

є доступність через браузер, простий інтерфейс і можливість дистанційного доступу до медичних послуг. Однак система значною мірою залежить від стабільності інтернет-з'єднання та центрального сервера. Крім того, функціональність системи орієнтована переважно на інтеграцію з державною медичною інфраструктурою, що обмежує можливості локального використання або адаптації під потреби окремих медичних закладів [3].

Під час аналізу також було розглянуто систему Medstar, яка використовується для ведення електронної медичної документації та структурованого зберігання інформації про пацієнтів. Система підтримує роботу з медичними картками, призначеннями, лабораторними результатами та звітністю. Основною перевагою Medstar є висока структурованість даних та можливість роботи з великими обсягами інформації. Водночас система орієнтована переважно на великі клініки та медичні центри, через що має складний інтерфейс і підвищені технічні вимоги до серверного обладнання [4].

Окрему увагу приділено державній системі eHealth, яка забезпечує централізований обмін медичними даними між медичними закладами, лікарями та пацієнтами. Система підтримує електронні декларації, електронні рецепти та взаємодію з аптечними сервісами. Основними перевагами eHealth є високий рівень безпеки та відповідність державним стандартам у сфері охорони здоров'я. Проте система має складну процедуру підключення, потребує інтеграції із зовнішніми сервісами та не призначена для автономного використання у невеликих медичних закладах [5].

Проведений аналіз показав, що більшість сучасних медичних інформаційних систем мають широкий функціонал і орієнтовані на використання у великих медичних установах. Такі системи забезпечують інтеграцію з державними сервісами, підтримують аналітичні модулі та автоматизують значну кількість процесів. Водночас для невеликих клінік або приватних медичних кабінетів вони часто є надмірно складними, потребують значних фінансових витрат і мають високі вимоги до технічної інфраструктури.

У межах роботи розроблено вебсистему електронного обліку пацієнтів та медичних призначень, орієнтовану на спрощення роботи лікаря та централізоване зберігання медичної інформації. Система побудована на основі архітектурного шаблону MVC, що забезпечує розділення логіки обробки даних, інтерфейсу користувача та роботи з базою даних. Серверна частина реалізована мовою PHP, а для зберігання даних використовується система керування базами даних MySQL [6]. Клієнтська частина створена

засобами HTML, CSS та JavaScript, що забезпечує роботу системи через браузер без необхідності встановлення додаткового програмного забезпечення.

Функціональні можливості розробленої системи включають авторизацію користувачів, розмежування доступу між лікарем і пацієнтом, створення та редагування записів лікування, ведення історії прийомів, прикріплення медичних документів і автоматичне журналювання дій користувачів. Лікар має можливість працювати з базою пацієнтів, створювати записи лікування та додавати результати обстежень, тоді як пацієнт отримує доступ лише до власної історії лікування.

Основними перевагами розробленої вебсистеми є простота розгортання, автономність роботи, зрозумілий інтерфейс і невисокі вимоги до апаратного забезпечення. На відміну від великих комерційних платформ, система не потребує складної інтеграції із зовнішніми сервісами та може працювати локально на вебсервері. Завдяки використанню MVC-архітектури забезпечується можливість подальшого розширення функціональності системи. До недоліків розробленого рішення можна віднести відсутність інтеграції з державними медичними сервісами, а також обмежений набір аналітичних функцій у порівнянні з корпоративними медичними платформами.

Таким чином, проведений аналіз сучасних медичних інформаційних систем дозволив визначити їх основні переваги та недоліки, а також сформулювати вимоги до власного програмного рішення. Розроблена вебсистема електронного обліку пацієнтів та медичних призначень поєднує простоту використання, автономність роботи та захищене зберігання медичної інформації. Запропоноване рішення може бути використане у невеликих клініках, приватних медичних кабінетах або навчальних цілях як приклад сучасної медичної інформаційної системи.

Список використаних джерел:

1. World Health Organization. Global strategy on digital health 2020–2025. Geneva : WHO, 2021. 84 p.
2. Doctor Eleks : вебсайт. URL: Doctor Eleks (дата звернення: 20.05.2026).
3. Helsi : вебсайт. URL: Helsi (дата звернення: 20.05.2026).
4. Medstar Solutions : вебсайт. URL: Medstar Solutions (дата звернення: 20.05.2026).
5. eHealth : вебсайт. URL: eHealth Ukraine (дата звернення: 20.05.2026).
6. Welling L., Thomson L. PHP and MySQL Web Development. 5th ed. Boston : Addison-Wesley Professional, 2017. 688 p.

CERTIFICATE OF PARTICIPATION

ISC № 260529- 50

Nadiya Batrynychuk

participated in the VIII International Scientific and Theoretical Conference
**«Scientific Method: Reality and
 Future Trends of Researching»**

May 29, 2026

Montreal; Canada

Academic Workload: 24 hours of participation / 0.8 ECTS credits

Confirmed Outcome: scientific paper published

DOI 10.36074/scientia-29.05.2026

ISBN 979-8-89660-285-9

Miriam G.
 MIRIAM GOLDENBLAT

President and responsible organizer
 International Center of Scientific Research
 Unified Register of Public Associations Number: 1499141.



Hosted by
 an authorized
 Crossref member

The conference is
 registered in the
 ResearchBib

Conference
 proceedings
 are displayed in
 Google Books &
 Google Scholar

Bowker



ДОДАТОК Б

Вебсистема електронного обліку пацієнтів та медичних призначень

Лістинг створення БД

НТУУ «КПІ ім. Ігоря Сікорського»

Аркушів 1

Київ — 2026

```
CREATE DATABASE IF NOT EXISTS ai_health CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE ai_health;
```

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  login VARCHAR(100) NOT NULL UNIQUE,
  pass VARCHAR(255) NOT NULL,
  user_level INT DEFAULT 4,
  doc_lvl TINYINT DEFAULT 0,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

```
CREATE TABLE projects (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  label VARCHAR(255),
  description TEXT,
  date DATE,
  deadline DATE,
  status TINYINT DEFAULT 0,
  user_id INT,
  audit_id INT,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE SET NULL ON UPDATE CASCADE,
  FOREIGN KEY (audit_id) REFERENCES users(id) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE docs (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  url VARCHAR(255) NOT NULL,
  status TINYINT DEFAULT 0,
  project_id INT,
  user_id INT,
  keywords VARCHAR(255),
  uploaded_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (project_id) REFERENCES projects(id) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE logs (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  date DATETIME DEFAULT CURRENT_TIMESTAMP,
  about TEXT
) ENGINE=InnoDB;
```

ДОДАТОК В

Вебсистема електронного обліку пацієнтів та медичних призначень

Лістинг файлу doccontroller.php

НТУУ «КПІ ім. Ігоря Сікорського»

Аркушів 3

Київ — 2026

```

<?php

namespace controllers;

class DocController extends Controller {

    public function indexAction(){
        /\R::fancyDebug( TRUE );

        $condition = 'WHERE user_id = ?';
        $item_for_page = 10;

        !empty( $_GET['page'] ) ? $page = clear($_GET['page'], 'int') : $page = 0;
        $numOfOrder = \R::count( 'docs', 'WHERE user_id = ?', [$_SESSION['user']['id']] );

        if( $numOfOrder > $item_for_page ){

            $page -= 1;
            if( $page < 0 ){
                $page = 0;
            }

            $start_get_from = $page * $item_for_page;

            $condition .= ' limit ' . $start_get_from . ', ' . $item_for_page;

            $page_quantity = $numOfOrder / $item_for_page;
            $page_quantity = ceil($page_quantity);

            if( $page_quantity >= ($page + 1) ){

                $docs = \R::find( 'docs', $condition, [$_SESSION['user']['id']] );
            } else {
                $page_quantity = 0;
            }

        } else {
            $docs = \R::find( 'docs', $condition, [$_SESSION['user']['id']] );
        }

        $users = \R::findAll( 'users' );
        $projects = \R::findAll( 'projects' );

        $this->set(compact('docs', 'users', 'projects', 'page_quantity', 'page'));
    }

    public function viewAction(){

        /\R::fancyDebug( TRUE );

        $condition = 'WHERE project_id = ?';
        $item_for_page = 10;

        !empty( $_GET['page'] ) ? $page = clear($_GET['page'], 'int') : $page = 0;
        !empty( $_GET['id'] ) ? $id = clear($_GET['id'], 'int') : $id = 0;

        $numOfOrder = \R::count( 'docs', 'WHERE user_id = ?', [$_SESSION['user']['id']] );

        if( $numOfOrder > $item_for_page ){

            $page -= 1;
            if( $page < 0 ){
                $page = 0;
            }

            $start_get_from = $page * $item_for_page;

            $condition .= ' limit ' . $start_get_from . ', ' . $item_for_page;

            $page_quantity = $numOfOrder / $item_for_page;
            $page_quantity = ceil($page_quantity);

            if( $page_quantity >= ($page + 1) ){

```

```

        $docs = \R::find( 'docs', $condition, [ $id ] );
    }else{
        $page_quantity = 0;
    }

} else{
    $docs = \R::find( 'docs', $condition, [ $id ] );
}

$users = \R::findAll( 'users' );
$projects = \R::findAll( 'projects' );

$this->set(compact('docs', 'users', 'projects', 'page_quantity', 'page'));
}

public function searchAction(){

    $search = clear($_POST['search'], 'text');
    $searchText = '%' . $search . '%';

    $docs = \R::find( 'docs', ' name LIKE ? or description LIKE ? or keywords LIKE ?', [ $searchText, $searchText, $searchText ] );

    $users = \R::findAll( 'users' );
    $projects = \R::findAll( 'projects' );

    $this->set(compact('docs', 'users', 'projects'));
}

public function fileAction(){

    $name = clear($_POST['name'], 'text');

    if( $_FILES['load']['error'] != 0 ){
        $_SESSION['error'] = "Помилка завантаження файлу";
    }

    $uploadaddir = 'files/';
    $file = basename($_FILES['load']['name']);
    $existFile = $uploadaddir . $name . time() . $file;

    if( is_uploaded_file($_FILES['load']['tmp_name']) ){
        if( !file_exists($existFile) ){
            if( move_uploaded_file($_FILES['load']['tmp_name'], $existFile) ) {
                //
            }
            else {
                $_SESSION['error'] = "Помилка завантаження";
            }
        } else {
            $_SESSION['error'] = "Помилка завантаження";
        }
    }

    $id = clear($_POST['id'], 'int');
    $projectId = clear($_POST['proj_id'], 'int');
    $about = clear($_POST['about'], 'text');

    $words = clear($_POST['words'], 'text');

    if( isset($_SESSION['error']) ){
        if( $id > 0 ){
            $_SESSION['error'] = "";
        } else {
            redirect();
            die;
        }
    }

    $project = \R::findOne( 'projects', 'id = ?', [ $projectId ] );

    if( $id > 0 ){
        $docs = \R::findOne( 'docs', 'id = ?', [ $id ] );
    } else {

```

```

        $docs = \R::dispense( 'docs' );
    }
    $docs->name = $name;
    $docs->description = $about;
    $docs->user_id = $project->audit_id;
    $docs->project_id = $projId;
    $docs->url = $existFile;
    $docs->keywords = $words;
    $docId = \R::store( $docs );

    $_SESSION['error'] = 'Запит виконано';

    redirect();
}
public function removeAction(){

    $id = clear($_GET['id'], 'int');

    $docs = \R::findOne( 'docs', ' WHERE id = ? ', [ $id ] );

    \R::trash( $docs );

    $_SESSION['error'] = "Дані видалено";

    redirect();
}
}

```

ДОДАТОК Г

Вебсистема електронного обліку пацієнтів та медичних призначень

Лістинг файлу `/views/doc/index.php`

НТУУ «КПІ ім. Ігоря Сікорського»

Аркушів 3

Київ — 2026

```

<div class="doc-page">
  <div class="main">
    <div class="menu">
      <a class="link" href="#">
        <div class="img">
          
        </div>
      </a>

      <a class="link active project" href="/project">
        <span>Запис</span>
      </a>
      <a class="link user" href="/">
        <span>Користувачі</span>
      </a>
      <a class="link log" href="/log">
        <span>Журнал</span>
      </a>
      <a class="link exit" href="/welcome/exit">
        <span>Вихід</span>
      </a>
    </div>

    <div class="content">
      <div class="manage">
        <div class="left">
          <div class="item edit">
            Редагувати
          </div>
          <div class="item del">
            Видалити
          </div>
          <form class="form" method="POST" action="/doc/search">
            <input class="search" type="text" name="search" required placeholder="Пошук"
minlength="3">

            <input class="btn" type="submit" value="">
          </form>
        </div>

        <div class="right">

          <div class="create-show">
            <div class="create">Створити</div>
          </div>

          <div>
            <div class="name"><?=$_SESSION['user']['name']?></div>
          </div>
        </div>
      </div>
      <div class="items">
        <div class="item <?php if ( $_SESSION['user']['user_level'] > 2 ){ echo 'fix';} ?>">
          <p class="select"></p>
          <p class="name">Назва</p>
          <p class="name">Зауваження</p>
          <p class="name">Опис</p>
          <p class="show"></p>
          <p class="load"></p>
        </div>

        <?php if (!empty($docs)): ?>
          <div class="data-list" >
            <?php foreach ($docs as $doc): ?>
              <div class="item <?php if($_SESSION['user']['user_level'] > 2 ){echo 'fix';}
data-status="<?=$doc->status;?>"
data-user="<?=$doc->user_id;?>"
data-name="<?=$doc->name;?>"
data-des="<?=$doc->description;?>"
data-words="<?=$doc->keywords;?>"
data-doclvl="<?=$doc->doc_lvl;?>"
data-id="<?=$doc->id;?>"
              >

```

```

src="/img/i20.png"></a>

<p class="select"></p>

<p class="name"><?=$doc->name;?></p>
<p class="name"><?=$doc->keywords;?></p>
<p class="name"><?=$doc->description;?></p>

<p class="show show-img" data-url="/<?=$doc->url;?>">
    
</p>
<p class="load">
    <a href="/<?=$doc->url;?>" target="_blank"><img

</p>
</div>
<?php endforeach; ?>
</div>
<?php endif; ?>
</div>
<?php if ( !empty($page_quantity) ): ?>
    <?php $page++;?>
    <div class="pages">
        <?php if ($page !== 1): ?>
            <a class="prev" href="/doc/view?page=<?=$page - 1;?>">
                
            </a>
        <?php else: ?>
            <a class="prev" href="#">
                
            </a>
        <?php endif; ?>
        <p class="curr"><?=$page;?></p>
        <?php if ($page < $page_quantity): ?>
            <a class="next" href="/doc/view?page=<?=$page + 1;?>">
                
            </a>
        <?php else: ?>
            <a class="next" href="#">
                
            </a>
        <?php endif; ?>
    </div>
<?php endif; ?>
</div>
</div>

<div class="pop-up">
    <div class="center">
        <div class="top">
            <p class="name">Документ</p>
            <div class="close">
                
            </div>
        </div>
    </div>

    <form class="data" method="POST" action="/doc/file" enctype="multipart/form-data">
        <input class="inp-id" type="text" name="id" hidden>
        <input class="inp-proj-id" type="text" name="proj_id" hidden value="<?=(int)$_GET['id'];?>">

        <p class="label">Назва</p>
        <input class="full inp-name" type="text" name="name" required>

        <p class="label">Опис</p>
        <textarea class="full moretext inp-about" name="about"></textarea>

        <p class="label">Зауваження</p>
        <input class="full inp-words" type="text" name="words" required>

```

```

        <input type="hidden" name="MAX_FILE_SIZE" value="51200000" />
        <input class="load inp-load" name="load" type="file" required>

        <input class="btn" type="submit" value="Зберегти">
    </form>
</div>
</div>

<div class="pop-up-img">
    <div class="inner">
        
    </div>
</div>
```