МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОБЛІКУ ЛАБОРАТОРНИХ АНАЛІЗІВ»**

Виконала: студентка 4 курсу, групи 6.1213-пі-3
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.Д. Федорченко
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного запезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Федорченко Вікторії Дмитрівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи для обліку лабораторних аналізів

керівник роботи Лісняк Андрій Олександрович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 01.06.2026 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Технічне завдання.
2. Проєктування.
3. Програмна реалізація.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 09.12.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.12.2025	
2.	Збір вихідних даних.	26.01.2026	
3.	Обробка методичних та теоретичних	27.02.2026	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2026	
5.	Розробка третього розділу.	20.05.2026	
6.	Оформлення та нормоконтроль	25.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2026	

Студент _____ В.Д. Федорченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної системи для обліку лабораторних аналізів»: 69 с., 37 рис., 1 табл., 20 джерел, 4 додатки.

DOCKER, LABORATORY WORKFLOW, MEDLAB, MVC, MYSQL, PDO, PHP, REST-LIKE API, UML, WEB PLATFORM.

Об’єкт дослідження – вебплатформи для підтримки процесів медичної лабораторії.

Мета роботи – спроєктувати та реалізувати вебплатформу для створення направлень, лабораторної обробки та перегляду результатів.

Методи дослідження – аналітичний, моделювання, проєктування, програмний.

У роботі виконано проєктування інформаційної моделі, архітектури MVC, UML-діаграм та розробку MedLab. Клієнт – PHP-шаблони; сервер – PHP 8.3 (Router, Controllers, Services, Repositories); БД – MySQL 8.4; розгортання – Docker. Реалізовано автентифікацію, RBAC за ролями, повний workflow лабораторії, AJAX-пошук, двохрівневі статуси, часткове збереження результатів, адмін-модуль і статистику. Проведено функціональне тестування. Платформа забезпечує цифровий процес «лікар → лабораторія → пацієнт» і придатна для захисту та розвитку.

SUMMARY

Bachelor’s qualifying paper “Development of an Information System for Laboratory Test Management”: 69 pages, 37 figures, 1 table, 20 references, 4 supplements.

DOCKER, LABORATORY WORKFLOW, MEDLAB, MVC, MYSQL, PDO, PHP, REST-LIKE API, UML, WEB PLATFORM.

The object of research is web platforms for supporting medical laboratory processes.

The purpose of the work is to design and implement a web platform for creating referrals, laboratory processing, and viewing results.

Research methods include analytical method, modeling, design, and software development.

The work involves the design of an information model, MVC architecture, UML diagrams, and the development of MedLab. The client is implemented using PHP templates; the server is based on PHP 8.3 (Router, Controllers, Services, Repositories); the database utilizes MySQL 8.4; deployment is managed via Docker. Authentication, Role-Based Access Control (RBAC), a complete laboratory workflow, AJAX search, two-level statuses, partial saving of results, an administrator module, and statistics are implemented. Functional testing was conducted. The platform provides a digital workflow from doctor to laboratory to patient and is suitable for defense and further development.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Технічне завдання 9
1.1 Перелік умовних позначень, скорочень, термінів 9
1.2 Функціональні вимоги 10
1.2.1 Призначення і мета створення системи 11
1.2.2 Функціональні можливості системи 11
1.3 Нефункціональні вимоги 13
1.4 Опис предметної області 14
1.5 Опис системи 16
1.6 Огляд та порівняння аналогів 18
2 Проєктування 21
2.1 Діаграма варіантів використання 21
2.2 ER-модель бази даних та опис сутностей системи 22
2.2.1 Опис варіантів використання 25
2.3 Діаграма діяльності 30
2.4 Діаграма класів 31
2.5 Діаграма послідовності 34
2.6 Діаграма розгортання 34
2.7 Висновки до розділу 36
3 Програмна реалізація 37
3.1 Опис інструментів розробки 37
3.2 Основні класи, інтерфейси та функції системи 38
3.3 Реалізація клієнтської частини 40
3.4 Реалізація серверної частини 42

3.5 Керівництво користувача 46
3.5.1 Вхід та реєстрація 46
3.5.2 Робота пацієнта 49
3.5.3 Робота лікаря 50
3.5.4 Робота лаборанта 53
3.5.5 Робота адміністратора 55
3.5.6 Перегляд статистики системи 56
Висновки 58
Перелік посилань 59
Додаток А ApiController 61
Додаток Б OrderService 62
Додаток В UserRepository 64
Додаток Г Router 68

ВСТУП

Автоматизація обліку лабораторних досліджень є важливим напрямком цифровізації охорони здоров’я. Використання паперової документації та електронних таблиць ускладнює роботу й підвищує ризик помилок, тому розробка інформаційної системи для обліку лабораторних аналізів є актуальною задачею.

Автоматизація лабораторних процесів дозволяє централізовано зберігати дані, контролювати виконання досліджень і забезпечувати швидкий доступ до результатів аналізів.

Мета роботи – розробити інформаційну систему для обліку лабораторних аналізів, що автоматизує створення лабораторних замовлень, обробку результатів досліджень та взаємодію між лікарями, лаборантами і пацієнтами.

Для досягнення поставленої мети необхідно вирішити такі **задачі**:

– сформулювати функціональні та нефункціональні вимоги до системи;
– проаналізувати предметну область діяльності медичної лабораторії;
– розглянути та порівняти існуючі програмні рішення;
– спроєктувати структуру бази даних та побудувати ER-модель;
– розробити UML-діаграми основних процесів системи;
– реалізувати веборієнтовану інформаційну систему;
– забезпечити розмежування доступу між користувачами різних ролей;
– підготувати керівництво користувача.

Об’єкт дослідження – процес обліку лабораторних аналізів у мед-закладі.

Предмет дослідження – засоби та методи проєктування і розробки веборієнтованої інформаційної системи для автоматизації процесів лабораторних досліджень.

Методи дослідження: аналітичний, моделювання, проєктування, програмний та тестування програмного забезпечення.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Перелік умовних позначень, скорочень, термінів

Система – вебплатформа MedLab для обліку лабораторних аналізів, яка об’єднує клієнтський інтерфейс, серверну частину, базу даних та середовище розгортання Docker.

Користувач – авторизована особа, яка взаємодіє із системою. У межах розроблюваної платформи користувач може мати роль пацієнта, лікаря, лаборанта або адміністратора.

Гість – неавторизована особа, якій доступні сторінки входу та реєстрації пацієнта.

Пацієнт – користувач, який переглядає призначені йому аналізи, статуси замовлень і результати обстежень у особистому кабінеті.

Лікар – користувач, який створює замовлення на аналізи, обирає пацієнта, типи досліджень і конкретні параметри для лабораторного виконання.

Лаборант – користувач, який виконує лабораторний workflow: перегляд черги, вибір аналізів, взяття в роботу, внесення результатів.

Адміністратор – користувач, який керує обліковими записами (крім створення іншого адміністратора через UI) та переглядає статистику системи.

Замовлення – основна сутність платформи; документ обліку з унікальним кодом, прив’язкою до лікаря та пацієнта, статусом і набором позицій.

Позиція замовлення – один призначений тип аналізу з власним життєвим циклом статусів.

Тип аналізу – запис довідника, що описує вид лабораторного дослідження.

Параметр аналізу – вимірюваний показник із одиницею виміру та референсним діапазоном.

Лабораторний workflow – послідовність дій лаборанта: огляд замовлення → вибір аналізів (`/select`) → взяття в роботу (`/take`) → внесення результатів (`/enter`, `/results`).

Результат аналізу – зафіксоване лаборантом значення параметра після виконання дослідження.

Вебплатформа – сукупність компонентів MedLab: Nginx, PHP-FPM, MySQL, PHP-шаблони, JavaScript-модулі та Docker Compose.

Дашборд – сторінка узагальненої інформації для відповідної ролі (статистика замовлень або користувачів).

RBAC – Role-Based Access Control, розмежування доступу за ролями через `requireRole()` у контролерах.

Front Controller – `public/index.php`, єдина точка входу всіх HTTP-запитів.

Health endpoint – маршрут `/api/health` для перевірки працездатності застосунку та підключення до БД.

PHP – мова реалізації серверної логіки [1, 2, 14].

MySQL – СУБД для збереження даних [3].

PDO – інтерфейс доступу до MySQL із підготовленими запитами [15].

Docker Compose – інструмент оркестрації контейнерів Nginx, PHP і MySQL [8, 9].

MVC – шаблон розділення Model (репозиторії), View (шаблони), Controller.

REST API – HTTP-ендпоінти для пошуку пацієнтів, аналізів і моніторингу.

Сесія – механізм авторизації без JWT; дані користувача зберігаються в PHP Session.

UML – мова UML-діаграм проєктування.

ER-модель – модель бази даних із сімома таблицями.

Seed-скрипт – `scripts/seed_*.php` для наповнення БД тестовими даними.

UPSERT – операція вставки або оновлення результату при повторному збереженні.

1.2 Функціональні вимоги

Функціональні вимоги визначають основні можливості системи обліку лабораторних аналізів. Вони включають реєстрацію користувачів, створення та

обробку лабораторних замовлень, внесення результатів досліджень, їх перегляд і адміністрування системи.

Інформаційна система MedLab повинна забезпечувати повний цикл роботи з лабораторними аналізами та підтримувати розмежування прав доступу відповідно до ролей користувачів.

1.2.1 Призначення і мета створення системи

Функціональне призначення системи – автоматизація обліку лабораторних аналізів і зберігання інформації про користувачів, замовлення та результати досліджень.

Експлуатаційне призначення системи – підтримка процесів створення, виконання та перегляду лабораторних аналізів.

Мета створення системи – спростити облік лабораторних досліджень і забезпечити зручну взаємодію між лікарем, лаборантом і пацієнтом.

1.2.2 Функціональні можливості системи

У системі передбачено чотири ролі користувачів: пацієнт, лікар, лаборант і адміністратор.

Кожна роль має власний набір функцій відповідно до завдань, які виконуються в межах лабораторного процесу.

Пацієнт отримує доступ лише до власних лабораторних досліджень і результатів аналізів. Лікар має можливість створювати лабораторні замовлення та переглядати історію досліджень своїх пацієнтів. Лаборант відповідає за виконання аналізів і внесення результатів досліджень до системи. Адміністратор здійснює керування користувачами та перегляд статистичних даних.

Система повинна надавати пацієнтам такі можливості:

– зареєструвати обліковий запис;
– авторизуватися в системі;
– переглядати список власних лабораторних замовлень;
– переглядати результати виконаних аналізів;
– переглядати інформацію власного профілю;
– отримувати інформацію про поточний статус лабораторних досліджень.

Лікарям система повинна надавати такі можливості:

– авторизуватися в системі;
– переглядати список створених замовлень;
– створювати нові лабораторні замовлення;
– здійснювати пошук пацієнтів;
– переглядати історію лабораторних досліджень пацієнта;
– переглядати результати виконаних аналізів;
– додавати коментарі до лабораторних замовлень;
– контролювати статус виконання замовлень.

Лаборантам система повинна надавати такі можливості:

– переглядати чергу лабораторних замовлень;
– переглядати нові замовлення;
– брати замовлення в роботу;
– обирати аналізи для виконання;
– вносити результати лабораторних досліджень;
– частково зберігати результати аналізів;
– завершувати виконання лабораторних досліджень;
– переглядати завершені замовлення.

Адміністраторам система повинна надавати такі можливості:

– переглядати список користувачів;
– створювати нових користувачів;
– переглядати статистичні дані системи;
– контролювати роботу користувачів;
– виконувати адміністрування системи.

1.3 Нефункціональні вимоги

Нефункціональні вимоги визначають загальні характеристики системи, які не описують окремі дії користувача, але впливають на якість, зручність, безпеку, продуктивність і можливість подальшої підтримки вебплатформи [7, 12, 13].

Вимоги до інтерфейсу користувача. Інтерфейс системи має бути простим, зрозумілим і зручним для користувачів різних категорій. Навігація між сторінками повинна бути логічною, а всі основні функції мають бути доступними без необхідності спеціальної технічної підготовки.

Вимоги до адаптивності. Вебзастосунок повинен коректно працювати на різних типах пристроїв і підтримувати відображення в сучасних браузерах.

Вимоги до доступності. Форми введення даних повинні містити зрозумілі підписи та повідомлення про помилки. Усі кнопки й елементи керування мають мати назви, що відповідають виконуваним діям.

Вимоги до продуктивності. Система повинна забезпечувати швидке завантаження сторінок та оперативне виконання запитів до бази даних. Час відгуку основних сторінок не повинен перевищувати кількох секунд за нормальних умов експлуатації.

Вимоги до безпеки. Доступ до функціональних модулів системи повинен надаватися лише авторизованим користувачам. Паролі користувачів мають зберігатися у вигляді хешів із використанням механізмів PHP. Взаємодія з базою даних повинна виконуватися через підготовлені SQL-запити для захисту від SQL-ін’єкцій. Система повинна перевіряти права користувача на виконання кожної операції.

Вимоги до цілісності даних. Дані про лабораторні замовлення, аналізи та результати повинні зберігатися узгоджено. Видалення або зміна записів не повинні призводити до втрати пов’язаних даних або порушення структури бази даних.

Вимоги до надійності. Система повинна коректно обробляти типові помилки користувача, помилки авторизації, відсутність доступу до ресурсів та

некоректні вхідні дані. У разі виникнення помилки користувач повинен отримувати зрозуміле повідомлення.

Вимоги до сумісності. Вебзастосунок повинен коректно працювати в актуальних версіях браузерів Google Chrome, Mozilla Firefox, Microsoft Edge та Opera.

Вимоги до масштабованості. Архітектура системи повинна забезпечувати можливість подальшого розширення функціоналу, додавання нових ролей користувачів, типів аналізів і модулів без необхідності повної переробки програмного забезпечення.

Вимоги до конфіденційності. Конфігураційні параметри системи, зокрема налаштування підключення до бази даних, не повинні бути доступними стороннім користувачам і мають зберігатися окремо від програмного коду.

1.4 Опис предметної області

Предметною областю розроблюваної системи є облік лабораторних аналізів у медичних закладах. До цієї області належать процеси створення лабораторних замовлень, призначення аналізів лікарем, виконання досліджень лаборантом, внесення результатів до системи та надання доступу до них пацієнтам і лікарям [13].

У сучасних медичних установах лабораторні дослідження є одним із найважливіших джерел інформації для встановлення діагнозу, оцінки стану здоров’я пацієнта та контролю ефективності лікування. Щодня лабораторії виконують значну кількість аналізів, результати яких повинні бути своєчасно зафіксовані, збережені та доступні для подальшого використання медичним персоналом.

У багатьох випадках облік лабораторних досліджень ведеться за допомогою паперової документації або електронних таблиць. Такий підхід ускладнює пошук інформації, збільшує ризик втрати даних та створює додаткові

труднощі під час контролю виконання лабораторних замовлень. Використання спеціалізованої інформаційної системи дозволяє централізувати зберігання даних та забезпечити автоматизацію основних бізнес-процесів лабораторії.

Основною сутністю предметної області є лабораторне замовлення. Лабораторне замовлення створюється лікарем для конкретного пацієнта та містить перелік необхідних досліджень. До кожного дослідження можуть входити окремі параметри, значення яких визначаються під час проведення лабораторного аналізу.

Важливою складовою предметної області є користувачі системи. У межах інформаційної системи передбачено чотири ролі: пацієнт, лікар, лаборант та адміністратор. Кожна роль виконує власні функції та має визначені права доступу до інформації.

Лікар відповідає за створення лабораторних замовлень, вибір необхідних аналізів та перегляд результатів досліджень. Лаборант здійснює обробку замовлень, виконує лабораторні дослідження та вносить результати до системи. Пацієнт має можливість переглядати власні результати аналізів та відстежувати статус виконання замовлень. Адміністратор забезпечує керування користувачами та контроль роботи системи [12].

Окремим елементом предметної області є довідник лабораторних аналізів. Він містить типи аналізів, параметри досліджень, одиниці вимірювання та референтні значення. Використання довідника забезпечує стандартизацію введення даних та спрощує роботу медичного персоналу.

Важливою особливістю предметної області є необхідність контролю статусів виконання лабораторних досліджень. Замовлення проходить кілька етапів життєвого циклу: створення, виконання та завершення. Для кожного аналізу в межах замовлення також може визначатися власний статус виконання, що дозволяє більш точно контролювати процес лабораторної обробки.

Основними проблемами предметної області, які має вирішити система, є складність централізованого обліку лабораторних досліджень, ризик втрати або дублювання інформації, необхідність швидкого доступу до результатів аналізів та забезпечення взаємодії між усіма учасниками лабораторного процесу.

У межах розроблюваної системи робота з лабораторним замовленням розглядається як послідовність взаємопов’язаних етапів: створення замовлення лікарем, передача його до лабораторії, виконання досліджень, внесення результатів та перегляд отриманих даних лікарем і пацієнтом.

1.5 Опис системи

Розроблювана система MedLab є клієнт-серверною веборієнтованою інформаційною системою для обліку лабораторних аналізів. Вона призначена для автоматизації процесів створення лабораторних замовлень, виконання лабораторних досліджень, внесення результатів аналізів та надання доступу до них лікарям і пацієнтам через вебінтерфейс.

Клієнтська частина системи реалізована з використанням HTML, CSS, Tailwind CSS та JavaScript [4, 7, 20]. Вона забезпечує відображення інтерфейсу користувача, роботу з формами введення даних, взаємодію з серверною частиною та відображення результатів виконання лабораторних досліджень. Для кожної ролі користувача передбачено окремий набір сторінок та функціональних можливостей.

Серверна частина системи реалізована мовою програмування PHP 8.3 із використанням архітектурного підходу MVC (Model–View–Controller). Серверна частина відповідає за обробку HTTP-запитів, реалізацію бізнес-логіки, перевірку прав доступу користувачів, роботу із замовленнями, результатами аналізів та іншими даними системи.

Для збереження інформації використовується система керування базами даних MySQL 8.4. Взаємодія із базою даних здійснюється через PDO та підготовлені SQL-запити. У базі даних зберігаються користувачі системи, лабораторні замовлення, типи аналізів, параметри аналізів та результати лабораторних досліджень [3, 16].

Розгортання системи виконується за допомогою Docker Compose. Контейнерне середовище містить вебсервер Nginx, сервер PHP-FPM, базу даних

MySQL та допоміжний сервіс phpMyAdmin для адміністрування бази даних. Такий підхід спрощує встановлення, налаштування та перенесення системи між різними середовищами.

У системі передбачено чотири основні ролі користувачів: пацієнт, лікар, лаборант та адміністратор. Пацієнт має можливість переглядати власні результати аналізів та історію досліджень. Лікар створює лабораторні замовлення, здійснює пошук пацієнтів та переглядає результати аналізів. Лаборант відповідає за виконання лабораторних досліджень і внесення результатів до системи. Адміністратор здійснює керування користувачами та перегляд статистичних даних.

Основними модулями системи є:

– модуль авторизації та реєстрації користувачів, який забезпечує вхід до системи та розмежування доступу за ролями;
– модуль управління лабораторними замовленнями, який дозволяє лікарю створювати та переглядати замовлення;
– модуль лабораторної обробки, який забезпечує виконання аналізів і внесення результатів досліджень;
– модуль перегляду результатів, який надає доступ пацієнтам і лікарям до результатів лабораторних досліджень;
– модуль адміністрування користувачів, який дозволяє створювати нові облікові записи та керувати ними;
– модуль статистики, який відображає узагальнену інформацію про виконані лабораторні дослідження та роботу системи.

Основний бізнес-процес системи полягає в тому, що лікар створює лабораторне замовлення для пацієнта та обирає необхідні аналізи. Після цього замовлення надходить до лабораторії, де лаборант бере його в роботу та вносить результати досліджень. Після завершення обробки результати стають доступними лікарю та пацієнту через вебінтерфейс системи.

Таким чином, інформаційна система MedLab забезпечує повний цикл роботи з лабораторними дослідженнями: створення замовлення, виконання

аналізів, збереження результатів, контроль статусів та перегляд отриманих даних усіма учасниками лабораторного процесу.

1.6 Огляд та порівняння аналогів

Перед розробкою інформаційної системи MedLab було проведено аналіз існуючих рішень для автоматизації лабораторної діяльності, зокрема медичних інформаційних систем (MIC), лабораторних інформаційних систем (LIS), електронних таблиць та паперового обліку. Аналіз дозволив визначити їхні переваги й недоліки та сформувати перелік функціональних можливостей, необхідних для реалізації у розроблюваній системі.

Медичні інформаційні системи (MIC) – комплексне рішення для автоматизації роботи медичних закладів, що забезпечують ведення електронних карток пацієнтів, облік лікування та інтеграцію з лабораторіями. Їхніми перевагами є широкий функціонал і підтримка великої кількості користувачів, а недоліками є висока вартість впровадження та складність налаштування.

Лабораторні інформаційні системи (LIS) – призначені для автоматизації лабораторних процесів, включаючи реєстрацію зразків, облік результатів і формування звітності. Вони забезпечують широкий набір спеціалізованих функцій, проте характеризуються високою вартістю та складністю впровадження.

Google Sheets та Microsoft Excel – Електронні таблиці часто використовуються для обліку результатів аналізів завдяки простоті використання та низькій вартості. Водночас вони не забезпечують належного рівня безпеки, розмежування доступу та контролю лабораторних процесів..

Паперовий облік – найпростіший способ зберігання інформації та не потребує спеціального програмного забезпечення. Однак він ускладнює пошук даних, підвищує ризик втрати документів і не забезпечує оперативного доступу до результатів досліджень.

У таблиці 1.1 наведено порівняння аналогів із розроблюваною системою.

Таблиця 1.1 – Порівняння аналогів

Критерій	**Паперовий облік**	**Excel / Google Sheets**	**LIS**	**MIC**	**MedLab**
Централізоване зберігання даних	Ні	Частково	Так	Так	Так
Розмежування доступу	Ні	Ні	Так	Так	Так
Вебдоступ	Ні	Так	Так	Так	Так
Облік лабораторних замовлень	Ні	Частково	Так	Так	Так
Контроль статусів виконання	Ні	Ні	Так	Так	Так
Історія досліджень пацієнта	Ні	Частково	Так	Так	Так
Простота впровадження	Висока	Висока	Низька	Низька	Висока
Вартість використання	Низька	Низька	Висока	Висока	Низька

Аналіз аналогів показує, що існуючі рішення для автоматизації лабораторної діяльності мають широкий функціонал, проте часто орієнтовані або на комплексне управління медичним закладом, або на використання виключно в межах великих лабораторних мереж. Медичні інформаційні системи забезпечують повний цикл роботи лікувального закладу, включаючи ведення електронних медичних карток, запис пацієнтів та управління персоналом, тоді як спеціалізовані лабораторні інформаційні системи зосереджені на автоматизації лабораторних процесів і потребують значних витрат на впровадження та супровід. Використання електронних таблиць або паперового обліку є більш доступним варіантом, однак такі підходи не забезпечують належного рівня контролю, безпеки та централізованого зберігання даних.

Головною перевагою інформаційної системи MedLab є автоматизація основних лабораторних процесів: реєстрації пацієнтів, створення та контролю лабораторних замовлень, обліку досліджень і зберігання результатів аналізів, із підтримкою авторизації користувачів, розмежування прав доступу та централізованого зберігання даних; система реалізована на базі PHP, MySQL що забезпечує низьку вартість впровадження, просте розгортання та можливість подальшого масштабування.

Отже, розроблена інформаційна система не є повним аналогом великих медичних інформаційних комплексів, проте ефективно вирішує задачу

автоматизації лабораторних процесів, забезпечуючи необхідний набір функцій для роботи лабораторії без надлишкової складності та значних фінансових витрат. Результати проведеного аналізу підтверджують доцільність розробки системи та обґрунтовують вибір функціональних можливостей, реалізованих у проєкті.

2 ПРОЄКТУВАННЯ

2.1 Діаграма варіантів використання

Діаграма варіантів використання відображає основні функціональні можливості інформаційної системи та показує дії, які можуть виконувати різні користувачі. Для системи MedLab така діаграма дозволяє наочно продемонструвати розмежування доступу між пацієнтом, лікарем, лаборантом та адміністратором.

У системі передбачено чотири актори: пацієнт, лікар, лаборант та адміністратор. Пацієнт переглядає замовлення й результати аналізів, лікар створює замовлення та працює з даними пацієнтів, лаборант вносить результати досліджень, а адміністратор керує користувачами та контролює роботу системи. Діаграму варіантів використання системи наведено на рисунку 2.1.

Основним прецедентом системи є «Створити лабораторне замовлення», оскільки саме він запускає основний бізнес-процес інформаційної системи MedLab. У межах цього сценарію лікар обирає пацієнта, додає необхідні типи аналізів, визначає параметри дослідження та за потреби вказує коментар. Після створення замовлення система формує унікальний код і надає йому початковий статус «Нове».

Не менш важливим прецедентом є «Внести результати аналізів». Він виконується лаборантом після того, як замовлення потрапляє до лабораторної черги. Лаборант бере аналізи в роботу, заповнює результати параметрів і після завершення всіх необхідних досліджень система оновлює статуси аналізів та замовлення.

Прецедент «Переглянути результати аналізів» доступний пацієнту та лікарю. Пацієнт може переглядати лише власні результати, а лікар – результати тих замовлень, які були створені ним для відповідного пацієнта. Таке розмежування доступу забезпечує захист персональних даних і відповідає логіці роботи медичної лабораторії.

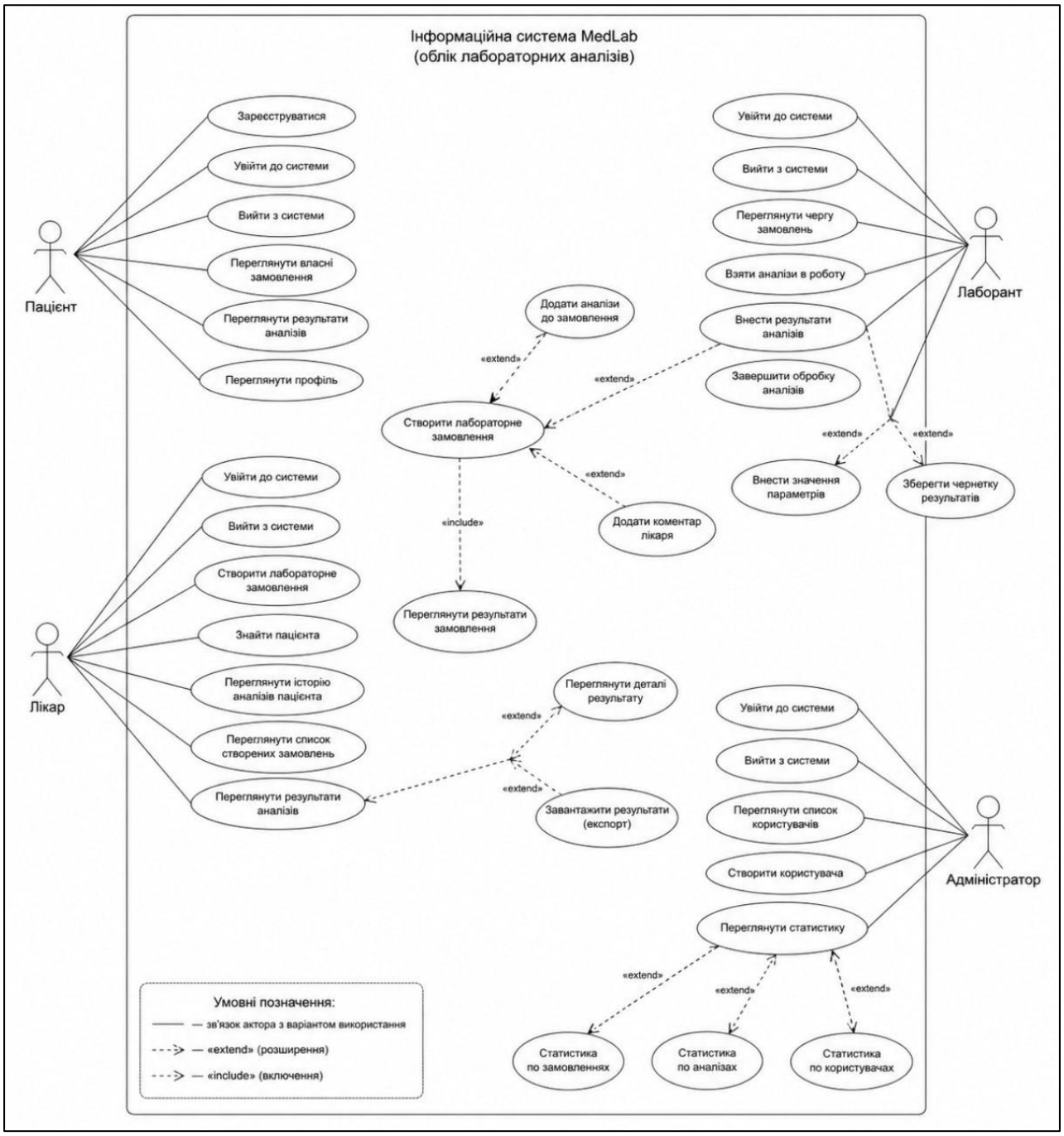


Рисунок 2.1 – Діаграма варіантів використання

2.2 ER-модель бази даних та опис сутностей системи

Проєктування бази даних є одним із ключових етапів створення інформаційної системи, оскільки саме база даних забезпечує зберігання, цілісність та узгодженість інформації, необхідної для роботи програмного забезпечення. Для інформаційної системи MedLab база даних повинна зберігати

дані про користувачів, лабораторні замовлення, типи аналізів, параметри досліджень та результати виконаних лабораторних аналізів [3].

Під час проєктування структури бази даних було враховано особливості предметної області медичної лабораторії. Основний бізнес-процес системи пов’язаний із тим, що лікар створює замовлення на лабораторні дослідження для пацієнта, лаборант виконує ці дослідження та вносить результати, а пацієнт і лікар отримують доступ до результатів через вебінтерфейс. ER-модель системи наведено на рисунку 2.2.

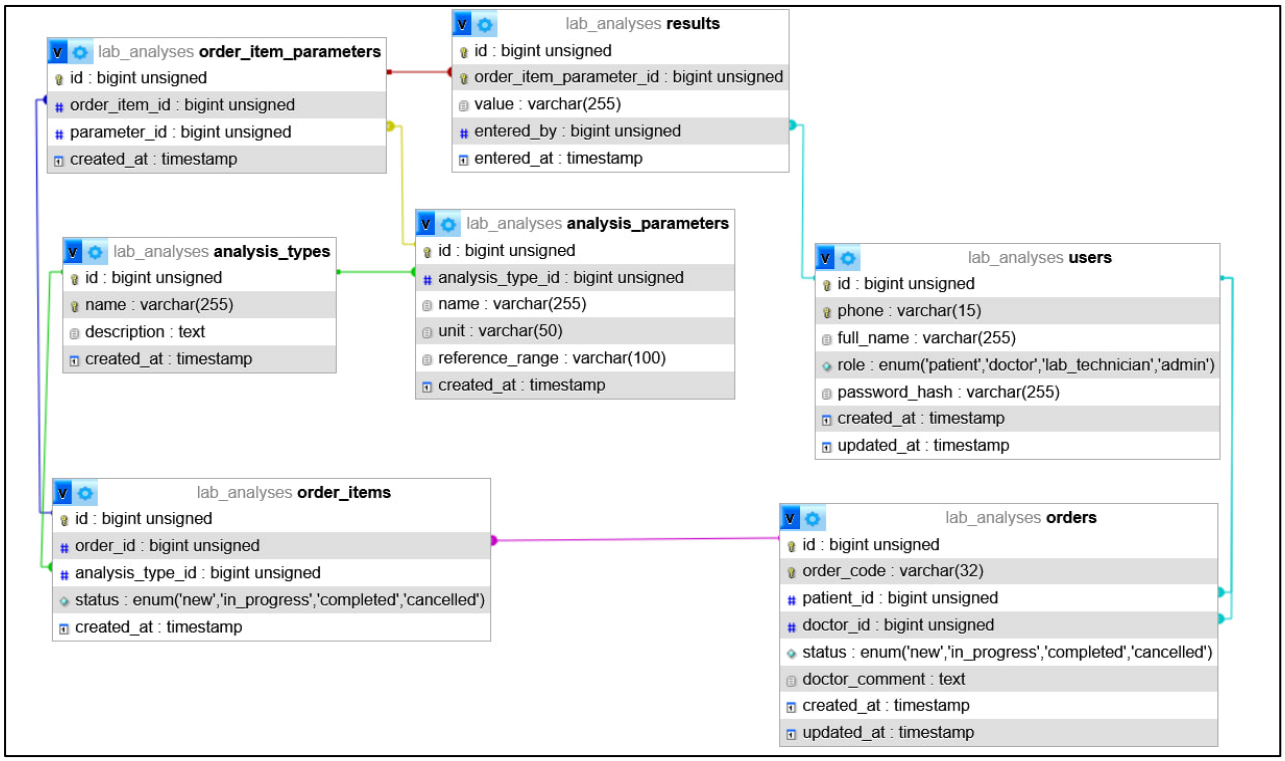


Рисунок 2.2 – ER-діаграма системи

База даних системи побудована за реляційною моделлю, що забезпечує структуроване зберігання даних, підтримку зв’язків між сутностями та цілісність інформації. Такий підхід є оптимальним для системи обліку лабораторних аналізів.

Сутність users використовується для зберігання даних про всіх користувачів системи. У межах однієї таблиці зберігаються пацієнти, лікарі, лаборанти та адміністратори. Роль користувача визначається окремим полем

role. Такий підхід дозволяє використовувати єдину модель авторизації для всіх категорій користувачів.

Сутність analysis_types містить довідник типів лабораторних аналізів. Наприклад, у системі можуть бути передбачені такі типи аналізів, як загальний аналіз крові, біохімічний аналіз крові та загальний аналіз сечі. Кожен тип аналізу має назву та опис.

Сутність analysis_parameters зберігає параметри, які входять до складу певного типу аналізу. Для кожного параметра визначається назва, одиниця вимірювання та референтний діапазон. Наприклад, для загального аналізу крові можуть зберігатися параметри «Гемоглобін», «Еритроцити», «Лейкоцити» та інші.

Сутність orders описує лабораторне замовлення. Замовлення містить унікальний код, посилання на пацієнта, посилання на лікаря, статус виконання, коментар лікаря та дату створення. Саме ця сутність є центральною у бізнес-процесі системи.

Сутність order_items використовується для зберігання окремих аналізів, що входять до складу замовлення. Одне замовлення може містити кілька аналізів, тому між таблицями orders і order_items існує зв’язок «один до багатьох».

Сутність order_item_parameters визначає, які саме параметри аналізу були призначені лікарем у межах конкретного замовлення. Це дозволяє не обмежуватися повним набором параметрів певного аналізу, а обирати лише ті показники, які необхідні для конкретного пацієнта.

Сутність results зберігає результати лабораторних досліджень. Кожен результат пов’язаний із конкретним параметром позиції замовлення та містить значення, інформацію про лаборанта, який вніс результат, і дату внесення.

Зв’язки між сутностями відображають логіку предметної області. Один користувач із роллю пацієнта може мати багато лабораторних замовлень. Один лікар також може створити багато замовлень. Одне замовлення може містити кілька аналізів, а кожен аналіз може містити кілька параметрів. Для кожного параметра може бути збережено один результат.

2.2.1 Опис варіантів використання

Прецедент «Зареєструватися».

Призначення: даний варіант використання надає можливість новому пацієнту створити обліковий запис у системі.

Передумови: користувач знаходиться на сторінці реєстрації та не має облікового запису в системі.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач відкриває сторінку реєстрації та вводить свої персональні дані: прізвище, ім’я, по батькові, номер телефону та пароль. Після натискання кнопки реєстрації система перевіряє коректність введених даних, перевіряє унікальність номера телефону, виконує хешування пароля та створює новий обліковий запис із роллю пацієнта. Після успішної реєстрації користувач отримує можливість авторизуватися в системі та користуватися її функціями.

Виняткова ситуація 1: введені дані не відповідають вимогам валідації – система виводить повідомлення про помилку.

Виняткова ситуація 2: користувач із таким номером телефону вже існує – система повідомляє про неможливість створення дубліката облікового запису.

Прецедент «Увійти до системи».

Призначення: даний варіант використання надає можливість користувачу отримати доступ до функцій системи відповідно до його ролі.

Передумови: користувач має зареєстрований обліковий запис.

Основний потік подій: користувач відкриває сторінку авторизації та вводить номер телефону і пароль. Система знаходить користувача в базі даних, виконує перевірку пароля за допомогою механізму password_verify() та створює сесію користувача. Після успішної авторизації система визначає роль користувача та перенаправляє його до відповідного розділу: кабінету пацієнта, лікаря, лаборанта або адміністратора.

Виняткова ситуація 1: пароль введено неправильно – система повідомляє про помилку авторизації.

Виняткова ситуація 2: обліковий запис не знайдено – система не надає доступ до захищених сторінок.

Прецедент «Вийти із системи».

Призначення: даний варіант використання надає можливість завершити роботу із системою.

Передумови: користувач авторизований у системі.

Основний потік подій: користувач натискає кнопку виходу. Система завершує поточну сесію, видаляє службові дані авторизації та перенаправляє користувача на сторінку входу. Після завершення сесії доступ до захищених сторінок стає неможливим без повторної авторизації.

Виняткова ситуація 1: сесія вже завершена – система все одно перенаправляє користувача на сторінку входу.

Прецедент «Переглянути власні замовлення».

Призначення: даний варіант використання надає можливість пацієнту переглядати список власних лабораторних замовлень.

Передумови: користувач авторизований як пацієнт.

Основний потік подій: після відкриття сторінки особистого кабінету система виконує запит до бази даних та отримує список лабораторних замовлень поточного пацієнта. На сторінці відображаються код замовлення, дата створення, лікар, який призначив дослідження, та поточний статус виконання. Пацієнт може перейти до перегляду деталей будь-якого замовлення.

Виняткова ситуація 1: замовлення відсутні – система відображає повідомлення про відсутність результатів.

Виняткова ситуація 2: користувач не авторизований – система перенаправляє його на сторінку входу.

Прецедент «Переглянути результати аналізів».

Призначення: даний варіант використання надає можливість пацієнту або лікарю переглянути результати лабораторних досліджень.

Передумови: замовлення існує в системі та користувач має право доступу до нього.

Основний потік подій: користувач відкриває сторінку замовлення. Система перевіряє права доступу, отримує результати аналізів із бази даних та відображає перелік показників, їх значення, одиниці вимірювання та референтні діапазони. Також відображається статус замовлення та інформація про лікаря.

Виняткова ситуація 1: замовлення не знайдено – система відображає сторінку помилки.

Виняткова ситуація 2: користувач намагається переглянути чуже замовлення – доступ блокується.

Прецедент «Створити лабораторне замовлення».

Призначення: даний варіант використання надає можливість лікарю створити нове лабораторне замовлення.

Передумови: лікар авторизований у системі, а пацієнт зареєстрований у базі даних.

Основний потік подій: лікар відкриває форму створення замовлення, знаходить пацієнта, обирає необхідні типи аналізів та параметри досліджень. За потреби лікар додає коментар до замовлення. Після підтвердження система перевіряє дані, створює запис замовлення, генерує унікальний код та встановлює статус «Нове». Створене замовлення автоматично потрапляє до черги лаборанта.

Виняткова ситуація 1: пацієнта не знайдено – система не дозволяє створити замовлення.

Виняткова ситуація 2: не вибрано жодного аналізу – система виводить повідомлення про помилку.

Виняткова ситуація 3: виникла помилка збереження даних – система повідомляє про технічну проблему

Прецедент «Знайти пацієнта».

Призначення: даний варіант використання надає можливість лікарю швидко знайти пацієнта в системі.

Передумови: лікар авторизований у системі.

Основний потік подій: лікар вводить частину ПІБ або номер телефону в поле пошуку. Система виконує пошук у базі даних та повертає список

користувачів, які відповідають введеним критеріям. Лікар обирає потрібного пацієнта та переходить до його профілю або історії аналізів.

Виняткова ситуація 1: пацієнтів не знайдено – система відображає повідомлення про відсутність результатів.

Виняткова ситуація 2: пошуковий запит порожній – пошук не виконується.

Прецедент «Переглянути історію аналізів пацієнта».

Призначення: даний варіант використання надає можливість лікарю переглянути попередні лабораторні дослідження пацієнта.

Передумови: лікар авторизований та вибрав пацієнта.

Основний потік подій: лікар відкриває сторінку історії пацієнта. Система отримує всі лабораторні замовлення цього пацієнта та відображає їх у хронологічному порядку. Лікар може переглянути результати кожного дослідження та оцінити динаміку показників.

Виняткова ситуація 1: у пацієнта відсутні лабораторні дослідження – система повідомляє про це користувача.

Прецедент «Переглянути чергу замовлень».

Призначення: даний варіант використання надає можливість лаборанту переглядати список замовлень, що потребують обробки.

Передумови: лаборант авторизований у системі.

Основний потік подій: лаборант відкриває сторінку лабораторної черги. Система відображає перелік нових, активних та завершених замовлень із зазначенням статусів, дати створення та інформації про пацієнта. Лаборант може обрати будь-яке доступне замовлення для подальшої роботи.

Виняткова ситуація 1: у черзі відсутні замовлення – система відображає відповідне повідомлення.

Прецедент «Взяти аналізи в роботу».

Призначення: даний варіант використання надає можливість лаборанту розпочати виконання лабораторних досліджень.

Передумови: існує замовлення зі статусом «Нове».

Основний потік подій: лаборант відкриває замовлення, обирає аналізи, які необхідно виконати, та натискає кнопку «Взяти в роботу». Система змінює статус вибраних аналізів на «В роботі», а також виконує оновлення статусу всього замовлення.

Виняткова ситуація 1: не обрано жодного аналізу – система повідомляє про помилку.

Виняткова ситуація 2: замовлення вже перебуває в роботі або завершене – система блокує виконання операції.

Прецедент «Внести результати аналізів».

Призначення: даний варіант використання надає можливість лаборанту внести результати лабораторних досліджень.

Передумови: замовлення перебуває в роботі.

Основний потік подій: лаборант відкриває сторінку внесення результатів та вводить значення показників. Система перевіряє введені дані та зберігає їх у базі даних. Якщо всі параметри аналізу заповнені, статус аналізу автоматично змінюється на «Виконано». Після завершення всіх аналізів система оновлює статус замовлення на «Завершено».

Виняткова ситуація 1: введені дані мають некоректний формат – система повідомляє про помилку.

Виняткова ситуація 2: виникла помилка під час збереження результатів – система повідомляє про технічну проблему.

Прецедент «Створити користувача».

Призначення: даний варіант використання надає можливість адміністратору створювати нові облікові записи користувачів.

Передумови: користувач авторизований та має роль адміністратора.

Основний потік подій: адміністратор відкриває форму створення користувача, вводить персональні дані, обирає роль та задає пароль. Система перевіряє введені дані, створює новий запис у базі даних та надає можливість новому користувачу увійти до системи.

Виняткова ситуація 1: номер телефону вже використовується – система не створює користувача.

Виняткова ситуація 2: дані не проходять перевірку – система повідомляє про помилку.

Прецедент «Переглянути статистику».

Призначення: даний варіант використання надає можливість адміністратору отримати статистичну інформацію про роботу системи.

Передумови: користувач має роль адміністратора.

Основний потік подій: адміністратор відкриває сторінку статистики. Система формує звіт на основі даних бази даних та відображає загальну кількість користувачів, кількість лабораторних замовлень, кількість виконаних аналізів, а також статистику роботи лікарів і лаборантів.

Виняткова ситуація 1: статистичні дані відсутні – система відображає нульові значення.

Виняткова ситуація 2: виникла помилка під час формування звіту – система повідомляє про технічну проблему.

2.3 Діаграма діяльності

Діаграма діяльності відображає процес обробки лабораторного замовлення в системі MedLab – від його створення лікарем до отримання результатів пацієнтом. У процесі беруть участь лікар, лаборант, пацієнт та система.

Дана діаграма демонструє основні етапи життєвого циклу лабораторного замовлення: створення, обробку, внесення результатів, зміну статусів та перегляд результатів користувачами системи. Також враховано альтернативний сценарій, коли результати аналізів внесені не повністю, що дозволяє зберігати проміжні дані та продовжувати роботу із замовленням пізніше.

Моделювання цього процесу дозволяє детально візуалізувати послідовність дій для кожного з учасників системи та оптимізувати алгоритми перевірки прав доступу на кожному кроці. Графічне представлення потоків керування спрощує подальшу програмну реалізацію відповідних контролерів і сервісів. Діаграму діяльності процесу обробки наведено на рисунку 2.3.

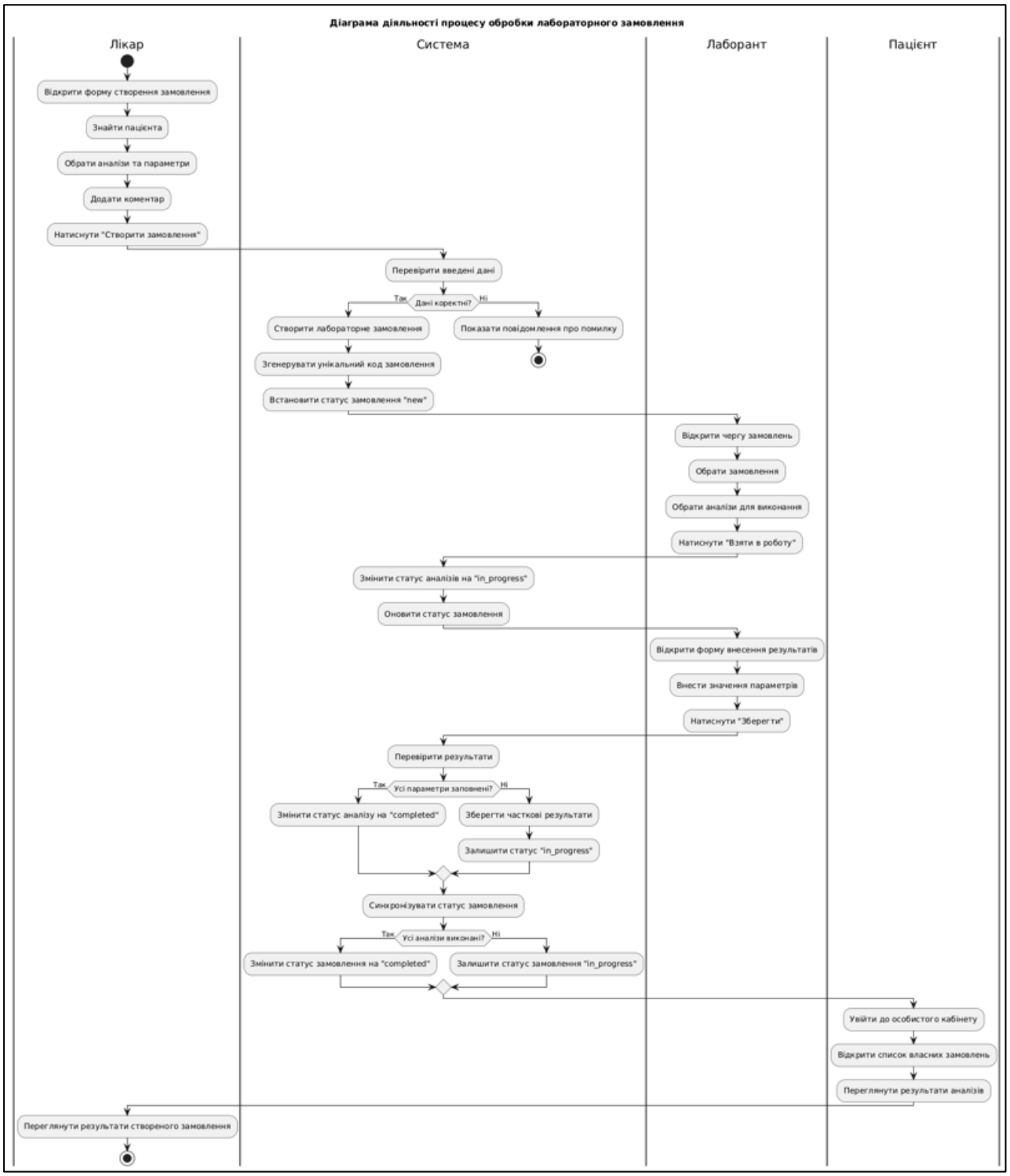


Рисунок 2.3 – Діаграма діяльності процесу створення навчального матеріалу

2.4 Діаграма класів

Діаграма класів використовується для відображення основних структурних елементів системи, їхніх атрибутів, методів і зв’язків між ними. На

відміну від діаграми варіантів використання, яка описує взаємодію користувачів із системою, діаграма класів дозволяє показати внутрішню структуру програмного забезпечення та взаємозв’язки між його компонентами.

Розроблювана інформаційна система MedLab побудована за архітектурним шаблоном MVC (Model–View–Controller) із використанням окремого шару сервісів та репозиторіїв. Такий підхід забезпечує розділення відповідальності між компонентами системи, спрощує супровід програмного коду та дозволяє розширювати функціональні можливості застосунку без значних змін існуючої архітектур.

У структурі системи можна виділити чотири основні групи класів: контролери, сервіси, репозиторії та інфраструктурні компоненти. Контролери відповідають за обробку HTTP-запитів користувачів та формування відповідей. Сервіси реалізують бізнес-логіку системи. Репозиторії забезпечують доступ до бази даних і виконання операцій збереження та отримання інформації. Інфраструктурні компоненти відповідають за маршрутизацію, роботу з базою даних, сесіями та відображенням сторінок.

До основних класів системи належать Controller, DoctorController, LabController, PatientController та AdminController, які реалізують функціональність для відповідних ролей користувачів. Бізнес-логіка зосереджена у класах OrderService, AuthService та AdminUserService. Для роботи з базою даних використовуються класи OrderRepository, UserRepository та AnalysisRepository. Допоміжні функції забезпечуються класами Database, Router, Session та View.

Діаграму класів інформаційної системи MedLab наведено на рисунку 2.4. Побудована діаграма відображає основні залежності між компонентами системи. Контролери взаємодіють із сервісами для виконання бізнес-операцій та з репозиторіями для отримання необхідних даних. Сервіси використовують репозиторії для роботи з інформацією про користувачів, замовлення та лабораторні аналізи. Репозиторії здійснюють безпосередню взаємодію з базою даних MySQL через компонент Database.

Така структура забезпечує низьку зв’язаність між модулями та відповідає сучасним підходам до розробки вебзастосунків.

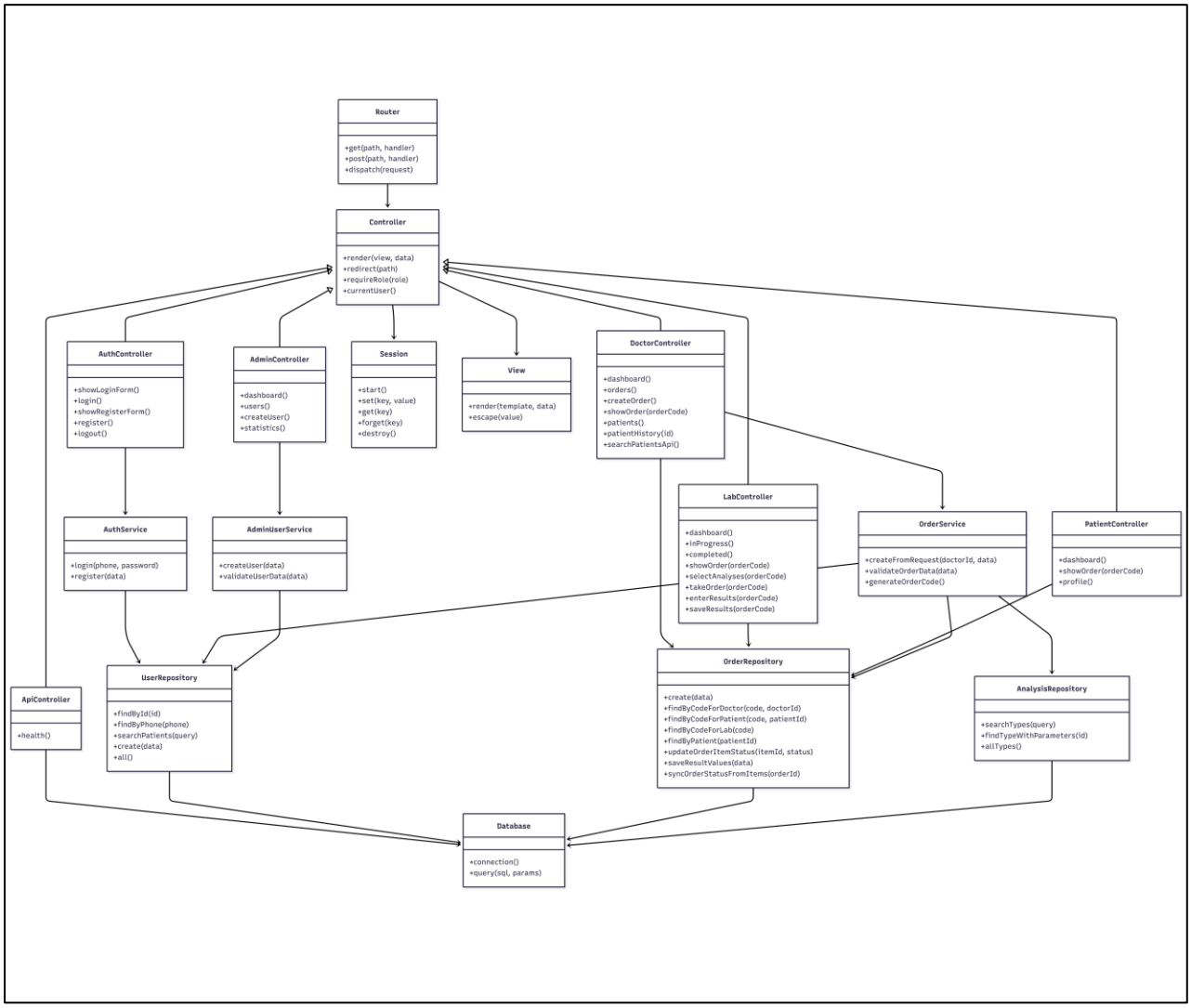


Рисунок 2.4 – Діаграма класів інформаційної системи MedLab

Побудована діаграма відображає основні залежності між компонентами системи. Контролери взаємодіють із сервісами для виконання бізнес-операцій та з репозиторіями для отримання необхідних даних. Сервіси використовують репозиторії для роботи з інформацією про користувачів, замовлення та лабораторні аналізи. Репозиторії здійснюють безпосередню взаємодію з базою даних MySQL через компонент Database. Така структура забезпечує низьку зв’язаність між модулями та відповідає сучасним підходам до розробки вебзастосунків.

2.5 Діаграма послідовності

Діаграма послідовності відображає порядок взаємодії між компонентами системи під час виконання певного сценарію.

Для системи MedLab побудовано діаграму послідовності для прецеденту «Створити лабораторне замовлення». У цьому процесі лікар створює замовлення, система перевіряє дані та зберігає їх у базі даних. Діаграму наведено на рисунку 2.5.

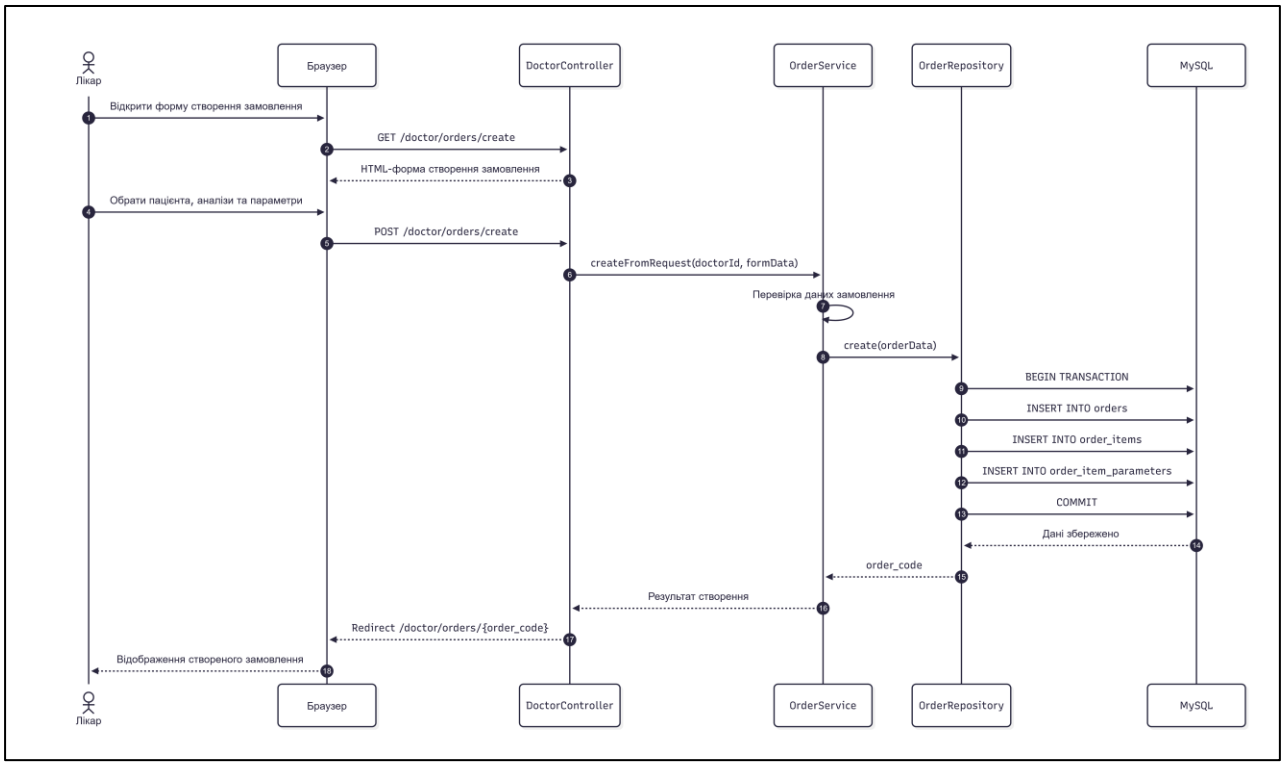


Рисунок 2.5 – Діаграма послідовності процесу створення лабораторного замовлення

2.6 Діаграма розгортання

Діаграма розгортання відображає фізичну структуру системи MedLab та розміщення її основних компонентів. Користувачі працюють із вебзастосунком через браузер, запити обробляються Nginx і PHP-FPM, а дані зберігаються в

MySQL. Для адміністрування бази даних використовується phpMyAdmin. Усі компоненти розгортаються за допомогою Docker Compose [1, 19]. Діаграму розгортання системи наведено на рисунку 2.6.

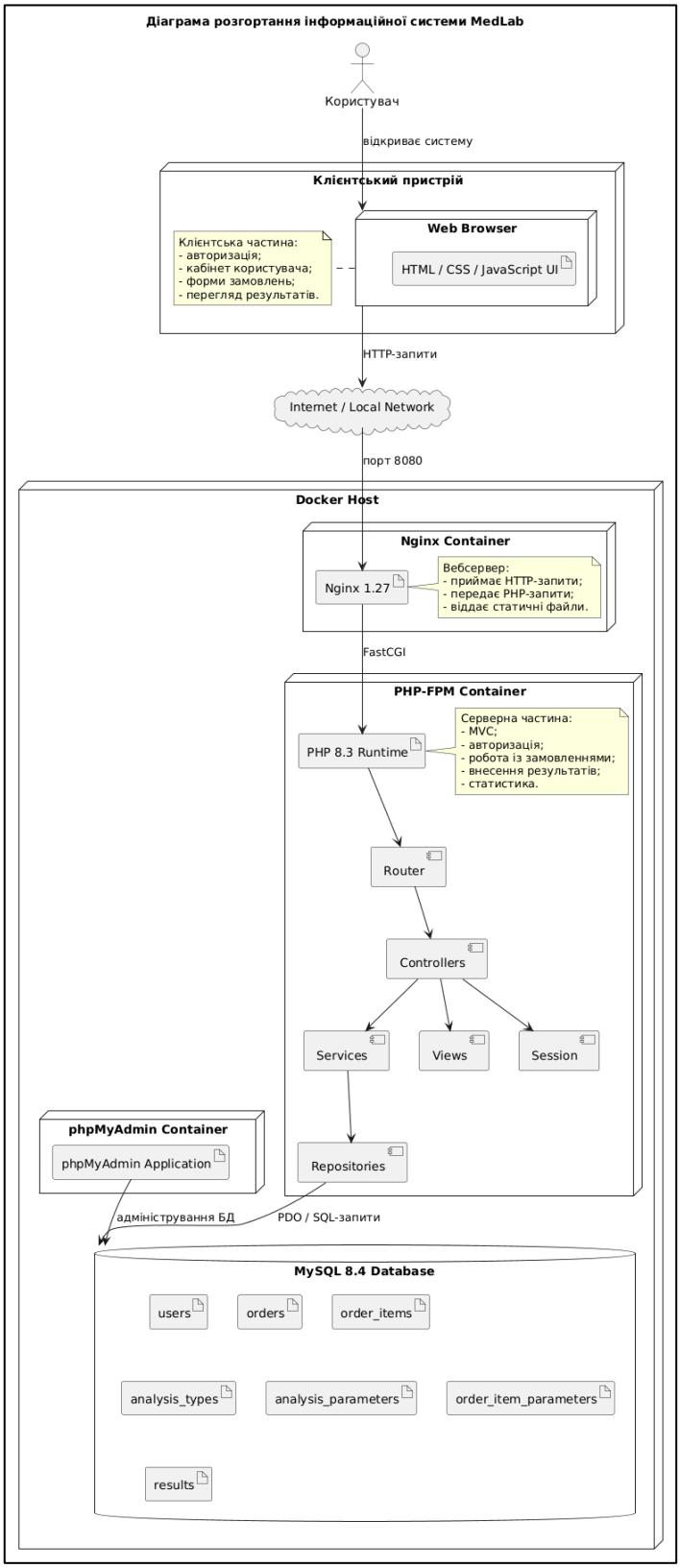


Рисунок 2.6 – Діаграма розгортання

2.7 Висновки до розділу

У другому розділі було виконано проєктування інформаційної системи для обліку лабораторних аналізів MedLab. На цьому етапі визначено структуру бази даних, основні сутності системи та зв’язки між ними. Побудована ER-модель відображає взаємодію між користувачами, лабораторними замовленнями, аналізами, параметрами досліджень і результатами, що забезпечує цілісність та узгодженість даних.

Також було розроблено UML-діаграми, які описують систему з різних поглядів. Діаграма варіантів використання відображає функціональні можливості пацієнта, лікаря, лаборанта та адміністратора. Діаграма діяльності демонструє послідовність виконання основного бізнес-процесу системи – від створення лабораторного замовлення до перегляду результатів аналізів. Діаграма класів відображає структуру програмного забезпечення та взаємозв’язки між контролерами, сервісами, репозиторіями й інфраструктурними компонентами системи. Діаграма послідовності деталізує процес створення лабораторного замовлення та взаємодію між користувачем, серверною частиною й базою даних. Діаграма розгортання відображає фізичну структуру системи та взаємодію між веббраузером, вебсервером Nginx, PHP-застосунком, базою даних MySQL і допоміжними сервісами Docker-середовища.

Отже, у результаті проєктування було сформовано повне уявлення про архітектуру, структуру даних, функціональні можливості та принципи взаємодії компонентів інформаційної системи MedLab. Побудовані моделі стали основою для реалізації програмного продукту, опис якої наведено в наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис інструментів розробки

Для реалізації інформаційної системи обліку лабораторних аналізів MedLab було використано сучасні програмні засоби та технології, які забезпечують розробку вебзастосунку, зберігання даних, підтримку клієнт-серверної взаємодії та розгортання програмного забезпечення в ізольованому середовищі.

Серверну частину системи реалізовано мовою програмування PHP версії 8.3. Використання PHP дозволило реалізувати бізнес-логіку системи, механізми авторизації користувачів, обробку лабораторних замовлень, внесення результатів аналізів та взаємодію з базою даних. Під час розробки було використано архітектурний шаблон MVC (Model-View-Controller), який забезпечує розділення логіки застосунку, представлення даних та їх обробки.

Для зберігання інформації використано реляційну систему керування базами даних MySQL версії 8.4. У базі даних зберігаються облікові записи користувачів, лабораторні замовлення, типи аналізів, параметри досліджень та результати лабораторних досліджень. Взаємодія з базою даних здійснюється за допомогою PDO, що забезпечує безпечне виконання SQL-запитів і захист від SQL-ін’єкцій.

Для організації веботочення використано вебсервер Nginx. Він відповідає за прийом HTTP-запитів від користувачів та передачу їх до PHP-FPM для подальшої обробки. Використання Nginx дозволяє забезпечити високу продуктивність та стабільність роботи вебзастосунку.

Для розгортання системи застосовано Docker та Docker Compose. Контейнеризація дозволила ізолювати всі компоненти системи, спростити процес встановлення та забезпечити однакове середовище виконання незалежно від операційної системи. Окремі контейнери використовуються для вебсервера Nginx, PHP-FPM, MySQL та phpMyAdmin.

Для створення користувацького інтерфейсу використано HTML5, CSS3, JavaScript та фреймворк Tailwind CSS. Tailwind CSS забезпечує швидке створення адаптивного інтерфейсу та дозволяє підтримувати єдиний стиль усіх сторінок системи. За допомогою JavaScript реалізовано інтерактивні елементи інтерфейсу та перевірку даних на стороні клієнта.

Під час розробки використовувалося середовище Cursor, яке забезпечує зручне написання програмного коду, підтримку мови PHP, інтеграцію з системою контролю версій Git, автоматичне доповнення коду та інструменти для підвищення продуктивності розробки. Для керування версіями програмного коду використовувалася система Git.

3.2 Основні класи, інтерфейси та функції системи

Основним бізнес-процесом інформаційної системи MedLab є створення лабораторного замовлення, виконання лабораторних досліджень та перегляд результатів аналізів користувачами системи. Тому основними програмними компонентами є ті, що забезпечують роботу із замовленнями, лабораторними аналізами, результатами досліджень та керуванням користувачами.

На рисунку 3.1 наведено фрагмент реалізації створення лабораторного замовлення та збереження його даних у базі даних.

```
99      public function createOrder(): void
100     {
101         $user = $this->requireRole('doctor');
102         $result = $this->orderService->createFromRequest((int) $user['id'], $_POST);
103
104         if (isset($result['errors'])) {
105             Session::flash('errors', $result['errors']);
106             Session::flash('old', $_POST);
107             $patientId = (int) ($_POST['patient_id'] ?? 0);
108             $redirect = '/doctor/orders/create' . ($patientId > 0 ? '?patient_id=' . $patientId : '');
109             $this->redirect($redirect);
110         }
111
112         $order = $result['order'];
113         Session::flash('success', 'Замовлення ' . $order['order_code'] . ' створено о ' . OrderHelper::formatDateTime($order['created_at']) . '.');
114         $this->redirect('/doctor/orders/' . rawurlencode($order['order_code']));
115     }
```

Рисунок 3.1 – Фрагмент реалізації створення лабораторного замовлення

Система реалізована відповідно до архітектурного шаблону MVC (Model-View-Controller), що забезпечує розділення бізнес-логіки, інтерфейсу користувача та роботи з даними. Основними складовими системи є маршрутизатор, контролери, сервіси, репозиторії та шаблони відображення.

Router є центральним компонентом системи, який відповідає за обробку HTTP-запитів та їх передачу відповідним контролерам. Завдяки маршрутизатору забезпечується централізоване керування всіма URL-адресами вебзастосунку.

AuthController – відповідає за реєстрацію користувачів, авторизацію, завершення сеансу роботи та перевірку прав доступу до захищених сторінок системи.

PatientController реалізує функції пацієнта, перегляд власних лабораторних замовлень, результатів досліджень та особистих даних.

DoctorController – реалізує функціональність лікаря. За допомогою цього контролера здійснюється пошук пацієнтів, створення лабораторних замовлень, перегляд історії аналізів та перегляд результатів досліджень.

LabController забезпечує роботу лаборанта із системою. Він відповідає за перегляд черги замовлень, вибір досліджень для виконання, внесення результатів аналізів та зміну статусів замовлень.

AdminController призначений для виконання адміністративних функцій. За його допомогою здійснюється перегляд списку користувачів, створення нових облікових записів та отримання статистичної інформації про роботу системи.

OrderService містить основну бізнес-логіку створення лабораторних замовлень. Він відповідає за перевірку введених даних, генерацію унікального коду замовлення та підготовку інформації для збереження в базі даних.

AuthService реалізує логіку авторизації користувачів, перевірку облікових даних та управління сесіями.

Для взаємодії з базою даних використовуються репозиторії. UserRepository забезпечує роботу з даними користувачів, OrderRepository відповідає за лабораторні замовлення та результати досліджень, а AnalysisRepository використовується для роботи з типами аналізів та їх параметрами.

Database є інфраструктурним компонентом, який забезпечує підключення до бази даних MySQL та виконання SQL-запитів через PDO [15].

Для відображення інформації користувачам використовуються шаблони View, які формують HTML-сторінки на основі отриманих даних. Інтерфейс системи побудований із використанням HTML, CSS, JavaScript та Tailwind CSS.

3.3 Реалізація клієнтської частини

Клієнтська частина інформаційної системи MedLab реалізована з використанням HTML5, CSS3, JavaScript та Tailwind CSS [4, 5, 6]. Вона відповідає за відображення інтерфейсу користувача, взаємодію з формами, первинну перевірку введених даних та надсилання HTTP-запитів до серверної частини.

Інтерфейс системи побудований відповідно до рольової моделі доступу та забезпечує окремі робочі області для пацієнта, лікаря, лаборанта та адміністратора. Після авторизації користувач отримує доступ лише до тих функцій, які передбачені його роллю.

Одним із ключових елементів клієнтської частини є форма створення лабораторного замовлення, яка дозволяє лікарю обрати пацієнта, призначити аналізи та сформувати замовлення. Перед збереженням система перевіряє коректність введених даних. Фрагмент реалізації форми створення лабораторного замовлення послідовно наведено на рисунках 3.2 та 3.3.

```
63      <form method="post" action="/doctor/orders/create" class="order-form" id="orderForm" novalidate>
64
65          <div class="form-section form-section-card order-step-card order-step-card--patient">
66
67              <div class="section-heading">
68
69                  <span class="step-label">Крок 1</span>
70
71                  <div class="section-heading-body">
72
73                      <h3>Оберіть пацієнта</h3>
74
75                      <p>Знайдіть пацієнта за ПІБ або номером телефону</p>
76
77                  </div>
78
79              </div>
```

Рисунок 3.2

```
88
89              <div class="entity-search patient-search" id="patientSearch">
90
91                  <div
92
93                      class="entity-search-selected"
94
95                      id="patientSelected"
96
97                      <?= $hasInitialPatient ? '' : 'hidden' ?>
98
99                  >
```

Рисунок 3.3

Після успішного створення замовлення користувач автоматично перенаправляється на сторінку перегляду створеного замовлення, де відображаються його основні реквізити, статус виконання та перелік призначених аналізів.

Для роботи лаборанта реалізовано сторінки перегляду черги лабораторних замовлень та форми внесення результатів досліджень. Під час введення результатів система виконує перевірку коректності значень та контролює заповнення обов’язкових параметрів аналізу.

Фрагмент реалізації форми внесення результатів лабораторних досліджень наведено на рисунку 3.4.

```
16  <form method="post" action="<?= \App\View::escape($resultsAction) ?>" class="lab-results-form" id="labOrderForm">
17      <section class="panel">
18          <div class="panel-header">
19              <h2>Крок 2 · Внесення результатів</h2>
20              <span class="badge badge-progress">Лаборант</span>
21          </div>
22
23          <p class="panel-text lab-results-hint">
24              Оберіть аналізи для заповнення та введіть значення — число (наприклад, <strong>142</strong>) або текст (наприклад, <strong>відсутній</strong>).
25              Можна зберегти частково заповнені результати та повернутися пізніше.
26          </p>
27
28          <?php
29          $selectedIds = $defaultSelectedIds;
30          $allowSelection = true;
31          $mode = 'entry';
32          $showOrderCodes = $hasMultipleOrders;
33          $itemsForPicker = $entryItems;
34          require __DIR__ . '/../partials/analysis-picker.php';
35          ?>
36      </section>
37
38      <section class="panel lab-results-panel">
39          <div class="panel-header">
40              <h2>Таблиці результатів</h2>
41          </div>
42
43          <?php require __DIR__ . '/../partials/results-entry-cards.php'; ?>
44      </section>
45
46      <div class="panel-actions panel-actions-spaced panel-actions--split lab-results-actions">
47          <a href="<?= \App\View::escape($orderBase) ?>" class="btn btn-secondary">← До замовлення</a>
48          <button type="submit" class="btn btn-primary btn-lg">Зберегти результати</button>
49      </div>
50  </form>
```

Рисунок 3.4 – Форма внесення результатів лабораторних досліджень

Для пацієнтів реалізовано сторінки перегляду власних замовлень та результатів аналізів. Інтерфейс дозволяє швидко отримати інформацію про статус досліджень, дату виконання аналізів та результати лабораторних показників.

Клієнтська частина також містить систему повідомлень про помилки та результати виконання операцій. У випадку некоректного введення даних або відсутності прав доступу користувач отримує відповідне повідомлення з поясненням причини помилки.

Таким чином, клієнтська частина MedLab забезпечує зручний доступ до функціональних можливостей системи, виконує первинну перевірку даних та формує інтерфейс взаємодії користувачів із серверною частиною застосунку.

3.4 Реалізація серверної частини

Серверна частина інформаційної системи MedLab реалізована мовою програмування PHP 8.3 та побудована відповідно до архітектурного шаблону MVC (Model-View-Controller). Вона відповідає за обробку HTTP-запитів, авторизацію користувачів, перевірку вхідних даних, взаємодію з базою даних та виконання основної бізнес-логіки системи [2].

Обробка запитів починається з маршрутизатора (Router), який визначає необхідний контролер та метод для виконання відповідної дії. Такий підхід дозволяє централізовано керувати всіма маршрутами вебзастосунку та спрощує подальше розширення системи.

Фрагмент реалізації маршрутизації наведено на рисунку 3.5.

```
36  $router->get('/doctor', fn () => $doctor->dashboard());
37  $router->get('/doctor/orders', fn () => $doctor->orders());
38  $router->get('/doctor/orders/create', fn () => $doctor->createOrderForm());
39  $router->post('/doctor/orders/create', fn () => $doctor->createOrder());
40  $router->get('/doctor/orders/{orderCode}', fn (string $orderCode) => $doctor->showOrder($orderCode));
41  $router->get('/doctor/patients', fn () => $doctor->searchPatients());
42  $router->get('/doctor/patients/{patientId}', fn (string $patientId) => $doctor->patientHistory((int) $patientId));
43
44  $router->get('/api/doctor/patients/search', fn () => $doctor->searchPatientsApi());
45  $router->get('/api/doctor/analyses/search', fn () => $doctor->searchAnalysesApi());
46  $router->get('/api/doctor/analyses/{id}', fn (string $id) => $doctor->analysisTypeApi($id));
```

Рисунок 3.5 – Реалізація маршрутизації в системі

Для роботи із лабораторними замовленнями використовується контролер DoctorController. Він приймає дані форми створення замовлення, виконує первинну перевірку введених даних та передає їх до сервісного шару для подальшої обробки.

Фрагмент контролера створення лабораторного замовлення наведено на рисунку 3.6.

```php
public function createOrder(): void
{
    $user = $this->requireRole('doctor');
    $result = $this->orderService->createFromRequest((int) $user['id'], $_POST);

    if (isset($result['errors'])) {
        Session::flash('errors', $result['errors']);
        Session::flash('old', $_POST);
        $patientId = (int) ($_POST['patient_id'] ?? 0);
        $redirect = '/doctor/orders/create' . ($patientId > 0 ? '?patient_id=' . $patientId : '');
        $this->redirect($redirect);
    }

    $order = $result['order'];
    Session::flash('success', 'Замовлення ' . $order['order_code'] . ' створено о ' . OrderHelper::formatDateTime($order['created_at']) . '.');
    $this->redirect('/doctor/orders/' . rawurlencode($order['order_code']));
}
```

Рисунок 3.6 – Контролер створення лабораторного замовлення

Основна бізнес-логіка системи реалізована в класі OrderService, який відповідає за створення лабораторних замовлень і перевірку коректності даних. У разі виявлення помилки система повідомляє користувача та скасовує виконання операції.

На рисунках 3.7 та 3.8 послідовно наведено реалізацію створення лабораторного замовлення.

```php
public function createFromRequest(int $doctorId, array $input): array
{
    $errors = [];

    $patientId = (int) ($input['patient_id'] ?? 0);
    $comment = trim((string) ($input['doctor_comment'] ?? ''));
    $selectedTypes = array_map('intval', (array) ($input['analysis_types'] ?? []));
    $selectedParameters = (array) ($input['parameters'] ?? []);

    $patient = $this->users->findPatientById($patientId);

    if ($patient === null) {
        $errors['patient_id'] = 'Оберіть пацієнта через пошук.';
    }

    if ($selectedTypes === []) {
        $errors['analysis_types'] = 'Оберіть хоча б один тип аналізу.';
    }

    $parametersByType = [];
```

Рисунок 3.7

```
40
41          foreach ($selectedTypes as $typeId) {
42              $typeParameterIds = array_map('intval', (array) ($selectedParameters[$typeId] ?? []));
43
44              if ($typeParameterIds === []) {
45                  $errors["parameters_{$typeId}"] = 'Оберіть хоча б один параметр для аналізу.';
46                  continue;
47              }
48
49              $validated = [];
50
51              foreach ($typeParameterIds as $parameterId) {
52                  if (!$this->analysis->parameterBelongsToType($parameterId, $typeId)) {
53                      $errors['parameters'] = 'Обрано некоректні параметри аналізу.';
54                      break 2;
55                  }
56
57                  $validated[] = $parameterId;
58              }
59
60              $parametersByType[$typeId] = array_values(array_unique($validated));
61          }
62
63          if ($errors !== []) {
64              return ['errors' => $errors];
65          }
66
67          $order = $this->orders->create($doctorId, $patientId, $parametersByType, $comment);
68
69          return ['order' => $order];
70      }
71  }
```

Рисунок 3.8

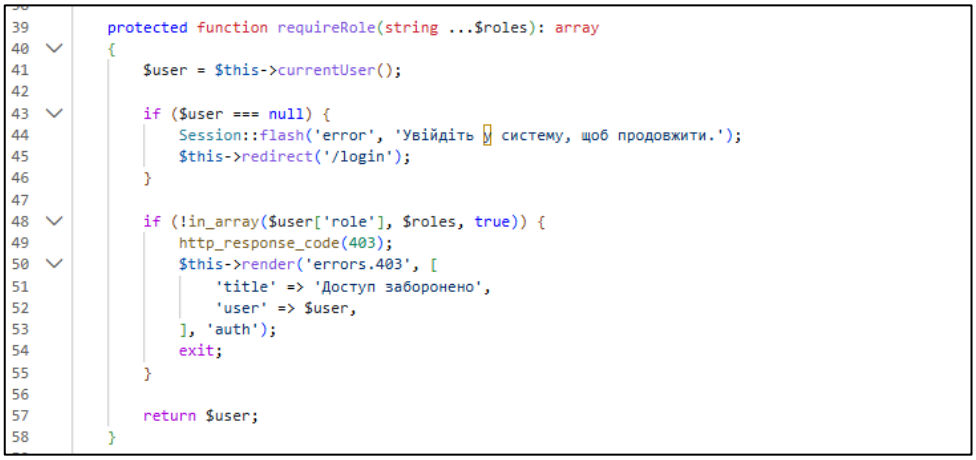
Для взаємодії з базою даних використовується шар репозиторіїв. Клас OrderRepository відповідає за виконання SQL-запитів, збереження інформації про замовлення та отримання даних із бази. Такий підхід дозволяє відокремити бізнес-логіку від механізмів доступу до даних.

Фрагмент репозиторію для роботи із замовленнями наведено на рисунку 3.9.

```
442         $this->db->beginTransaction();
443
444         try {
445             $orderCode = $this->generateUniqueOrderCode();
446
447             $stmt = $this->db->prepare(
448                 'INSERT INTO orders (order_code, patient_id, doctor_id, status, doctor_comment)
449                  VALUES (:order_code, :patient_id, :doctor_id, :status, :doctor_comment)'
450             );
451             $stmt->execute([
452                 'order_code' => $orderCode,
453                 'patient_id' => $patientId,
454                 'doctor_id' => $doctorId,
455                 'status' => 'new',
456                 'doctor_comment' => $comment !== '' ? $comment : null,
457             ]);
458
459             $orderId = (int) $this->db->lastInsertId();
460
461             $itemStmt = $this->db->prepare(
462                 'INSERT INTO order_items (order_id, analysis_type_id, status) VALUES (:order_id, :analysis_type_id, :status)'
463             );
464             $paramStmt = $this->db->prepare(
465                 'INSERT INTO order_item_parameters (order_item_id, parameter_id)
466                  VALUES (:order_item_id, :parameter_id)'
467             );
```

Рисунок 3.9 – Репозиторій роботи із лабораторними замовленнями

Для забезпечення безпеки системи реалізовано механізм авторизації користувачів. Перед виконанням захищених операцій система перевіряє наявність активної сесії та роль поточного користувача. Завдяки цьому лікар може створювати замовлення, лаборант – вносити результати аналізів, пацієнт – переглядати власні результати, а адміністратор – керувати користувачами системи. Фрагмент перевірки прав доступу наведено на рисунку 3.10.



```php
protected function requireRole(string ...$roles): array
{
    $user = $this->currentUser();

    if ($user === null) {
        Session::flash('error', 'Увійдіть у систему, щоб продовжити.');
        $this->redirect('/login');
    }

    if (!in_array($user['role'], $roles, true)) {
        http_response_code(403);
        $this->render('errors.403', [
            'title' => 'Доступ заборонено',
            'user' => $user,
        ], 'auth');
        exit;
    }

    return $user;
}
```

Рисунок 3.10 – Перевірка ролей та прав доступу

Після внесення результатів дослідження лаборантом система автоматично змінює статус замовлення та зберігає результати в таблиці результатів. Надалі ці дані стають доступними для перегляду лікарем і пацієнтом.

Фрагмент збереження результатів аналізів наведено на рисунку 3.11.



```php
$this->orders->saveResultValues($normalized, (int) $user['id']);

$completedItemNames = [];
$affectedOrderIds = [];

foreach ($inProgressItemIds as $itemId) {
    if (!$this->orders->hasAllResultsForItem($itemId)) {
        continue;
    }

    $this->orders->updateOrderItemStatus($itemId, 'completed');
    $item = $itemsById[$itemId] ?? null;

    if ($item !== null) {
        $completedItemNames[] = (string) $item['analysis_name'];
        $affectedOrderIds[(int) $item['order_id']] = true;
    }
}

$completedOrderCodes = [];

foreach (array_keys($affectedOrderIds) as $orderId) {
    $this->orders->syncOrderStatusFromItems($orderId);
```

Рисунок 3.11 – Збереження результатів лабораторного дослідження

Таким чином, серверна частина MedLab забезпечує централізовану обробку запитів, перевірку прав доступу, виконання бізнес-логіки та безпечну взаємодію з базою даних. Основні операції системи виконуються на сервері, що дозволяє забезпечити цілісність даних та коректність роботи інформаційної системи.

3.5 Керівництво користувача

3.5.1 Вхід та реєстрація

Для початку роботи з інформаційною системою MedLab користувачу необхідно перейти за адресою системи у веббраузері. Після завантаження сторінки відкривається форма авторизації (рис. 3.12).

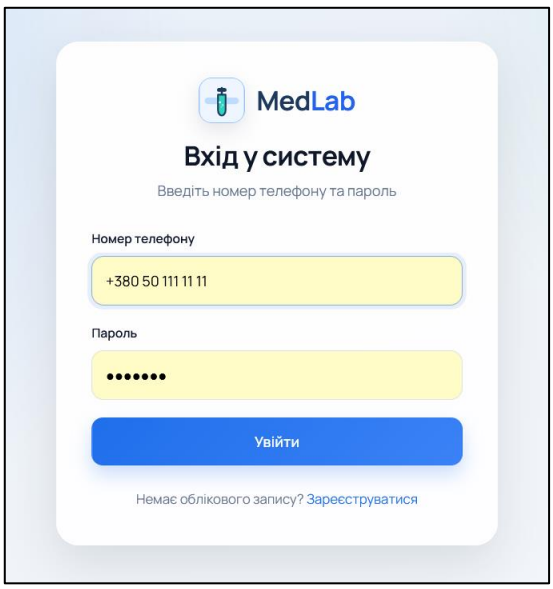


Рисунок 3.12 – Форма авторизації користувача

Для входу до системи користувач вводить номер телефону та пароль, після чого натискає кнопку «Увійти». Система перевіряє введені дані та, у разі успішної авторизації, перенаправляє користувача до особистого кабінету відповідно до його ролі (див. рис. 3.13).

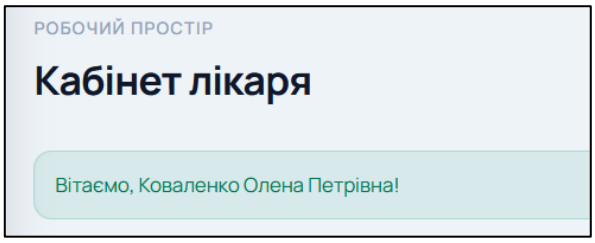


Рисунок 3.13 – Успішна авторизація в системі

Якщо користувач ще не має облікового запису, він може перейти до форми реєстрації. Для цього необхідно натиснути кнопку «Зареєструватися» на сторінці входу (рис. 3.14).

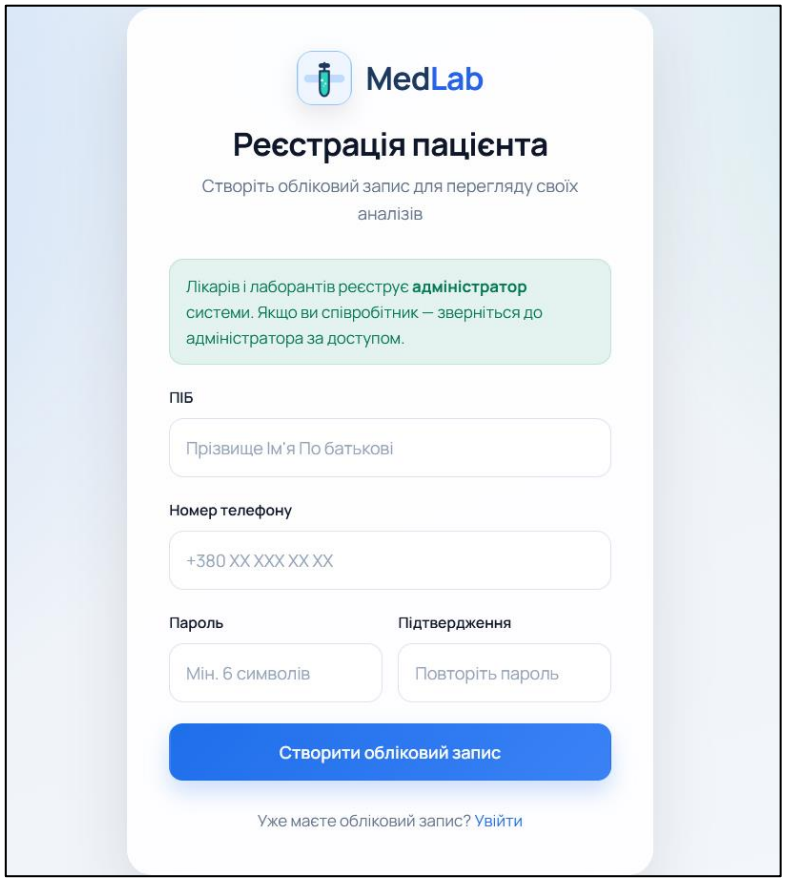


Рисунок 3.14 – Форма реєстрації нового користувача

Під час реєстрації користувач вводить прізвище, ім’я та по батькові, номер телефону, пароль і підтвердження пароля. Після заповнення всіх полів необхідно натиснути кнопку «Зареєструватися». Якщо введені дані відповідають встановленим вимогам, система створює новий обліковий запис та повідомляє про успішне завершення реєстрації (див. рис. 3.15).

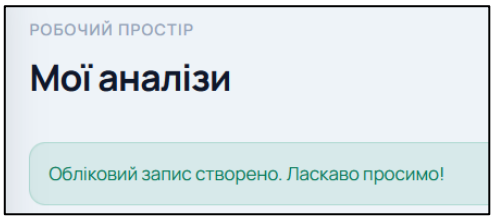


Рисунок 3.15 – Повідомлення про успішну реєстрацію пацієнта

У випадку введення некоректних даних, використання вже зареєстрованого номера телефону або невідповідності паролів система відображає відповідне повідомлення про помилку та пропонує виправити введену інформацію (рис. 3.16).

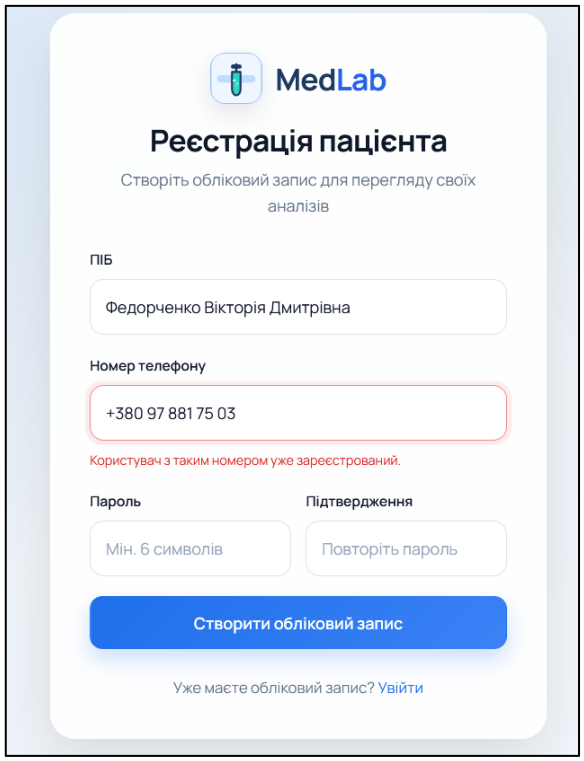


Рисунок 3.16 – Повідомлення про помилку під час реєстрації

Після успішної авторизації користувач отримує доступ до функцій системи відповідно до своєї ролі. Пацієнт може переглядати власні лабораторні замовлення та результати аналізів, лікар – створювати замовлення та переглядати історію досліджень пацієнтів, лаборант – вносити результати аналізів, а адміністратор – керувати користувачами системи та переглядати статистичну інформацію.

3.5.2 Робота пацієнта

Після авторизації пацієнт отримує доступ до особистого кабінету, де може переглядати свої лабораторні замовлення та результати аналізів (рис. 3.17). При виборі замовлення відкривається сторінка з інформацією про призначені аналізи, дату створення та статус виконання (рис. 3.18).

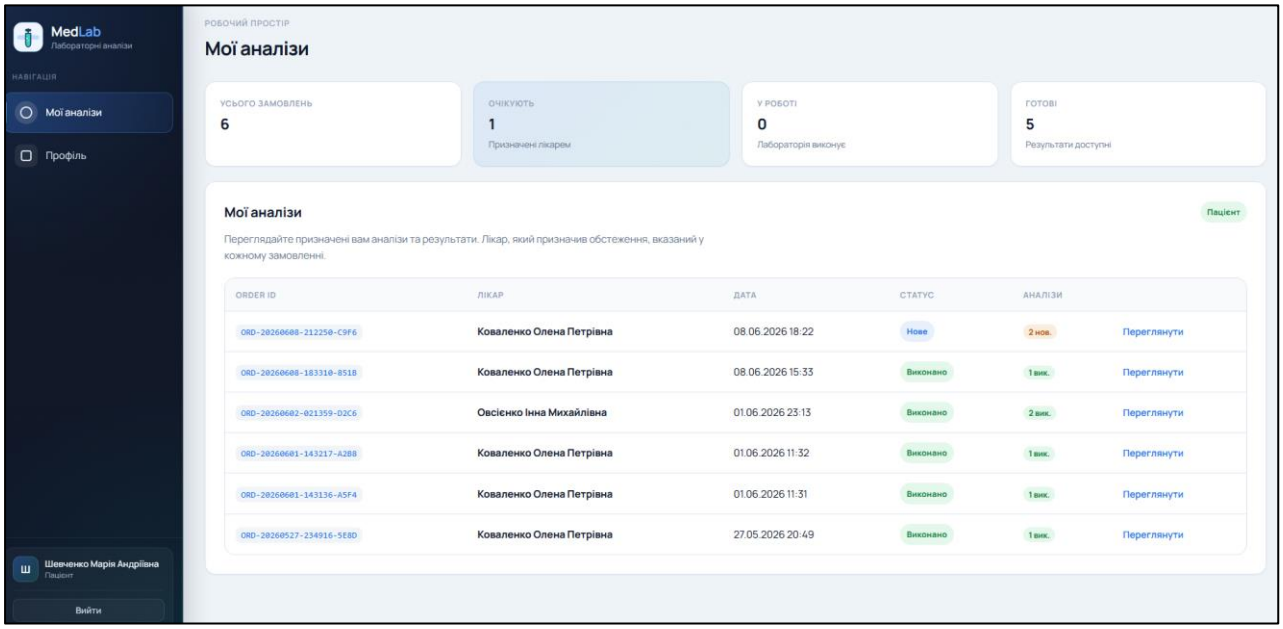


Рисунок 3.17 – Особистий кабінет пацієнта

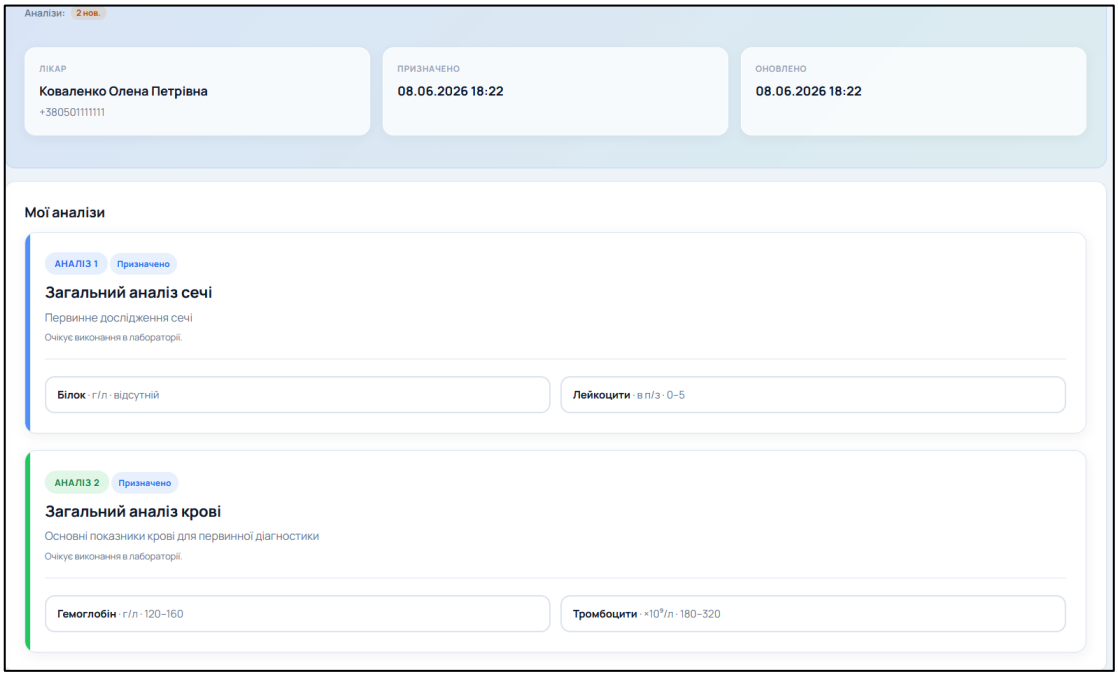


Рисунок 3.18 – Перегляд лабораторного замовлення

Після завершення досліджень пацієнт може переглянути результати аналізів, включаючи значення показників та референтні межі (рис. 3.19).

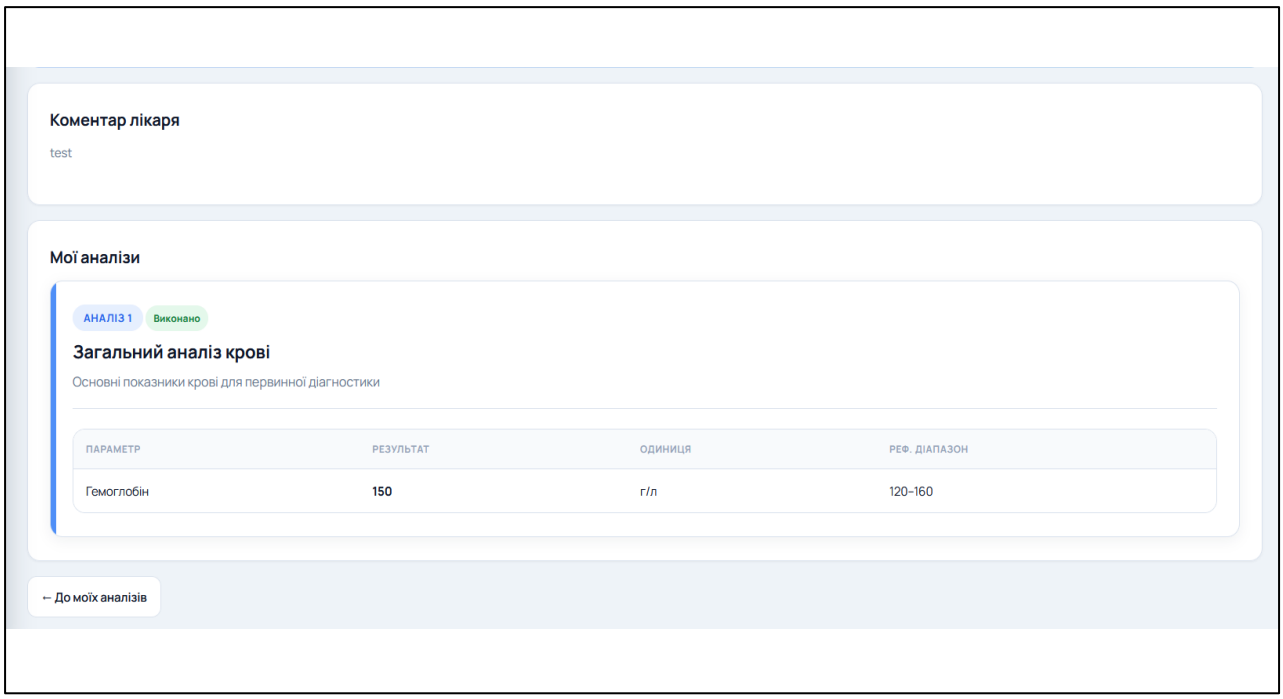


Рисунок 3.19 – Перегляд результатів аналізів

Функціональність кабінету забезпечує швидкий доступ до результатів та історії лабораторних досліджень.

3.5.3 Робота лікаря

Після авторизації лікар отримує доступ до функцій створення лабораторних замовлень та перегляду історії аналізів пацієнтів. Головна сторінка лікаря містить список створених замовлень та інструменти пошуку пацієнтів (див. рис. 3.20).

Для створення нового лабораторного замовлення лікар обирає пацієнта та натискає кнопку створення замовлення. Після цього відкривається форма, у якій необхідно вибрати типи аналізів та додаткові параметри дослідження (див. рис. 3.21).

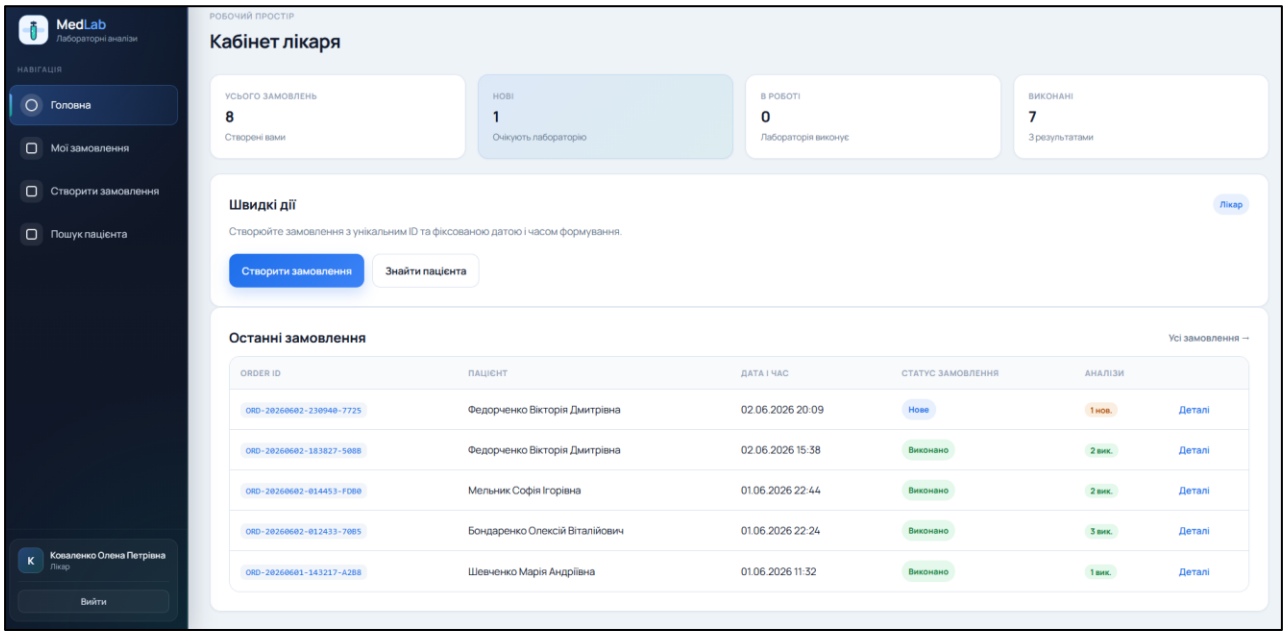


Рисунок 3.20 – Робоча сторінка лікаря

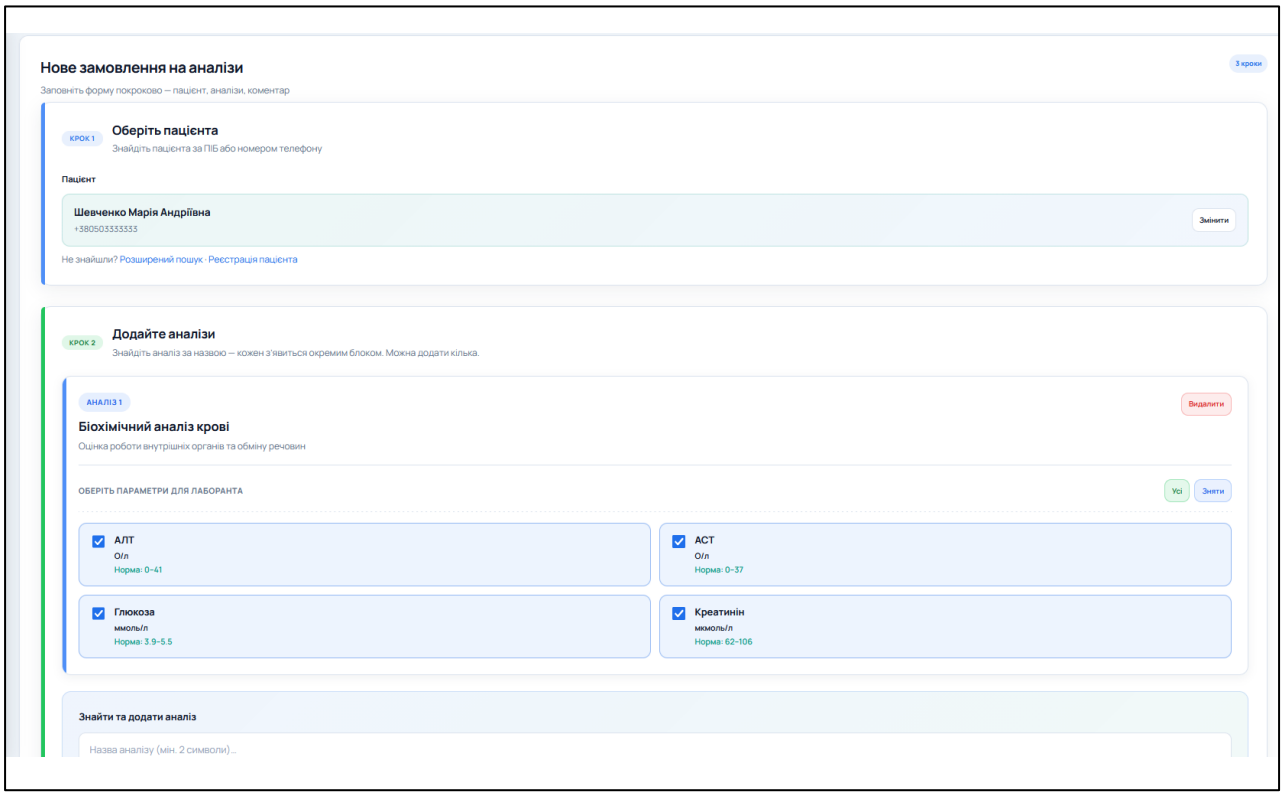


Рисунок 3.21 – Створення лабораторного замовлення

Після заповнення форми та підтвердження введених даних система створює нове замовлення та присвоює йому унікальний ідентифікатор.

Створене замовлення автоматично стає доступним для перегляду лаборантами (див. рис. 3.22).

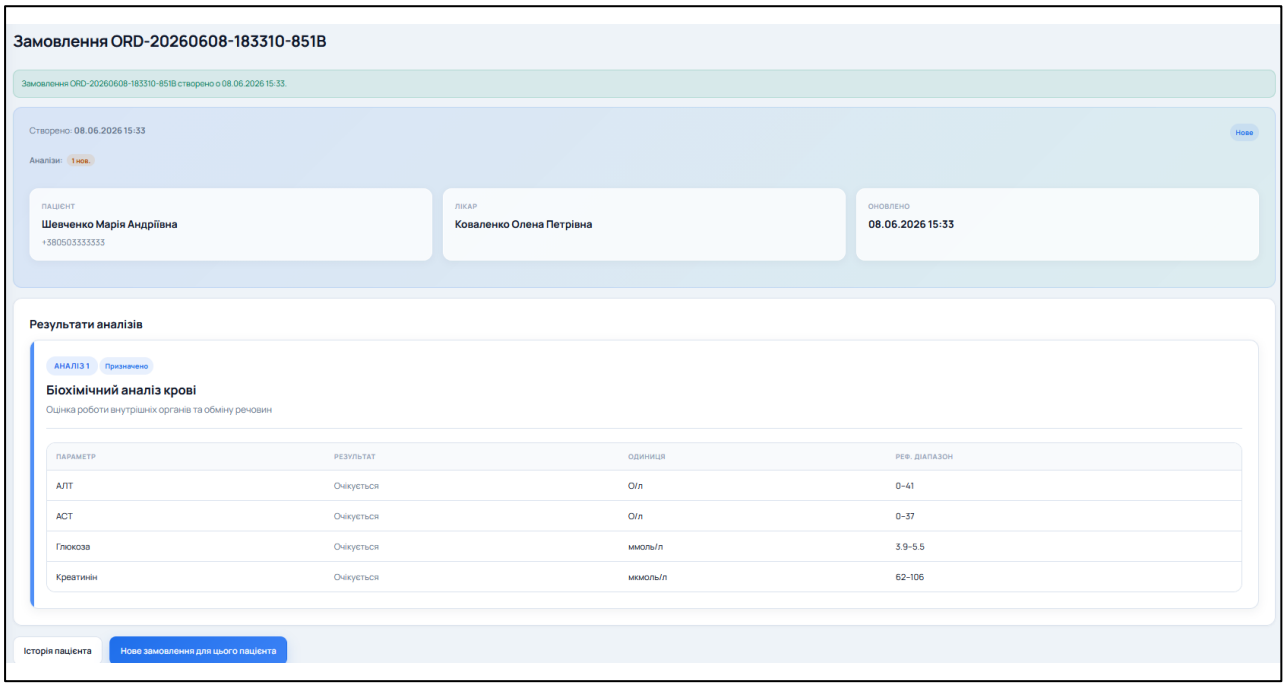


Рисунок 3.22 – Успішне створення лабораторного замовлення

Для перегляду попередніх досліджень лікар може відкрити історію аналізів пацієнта, де відображаються результати проведення досліджень (див. рис. 3.23).

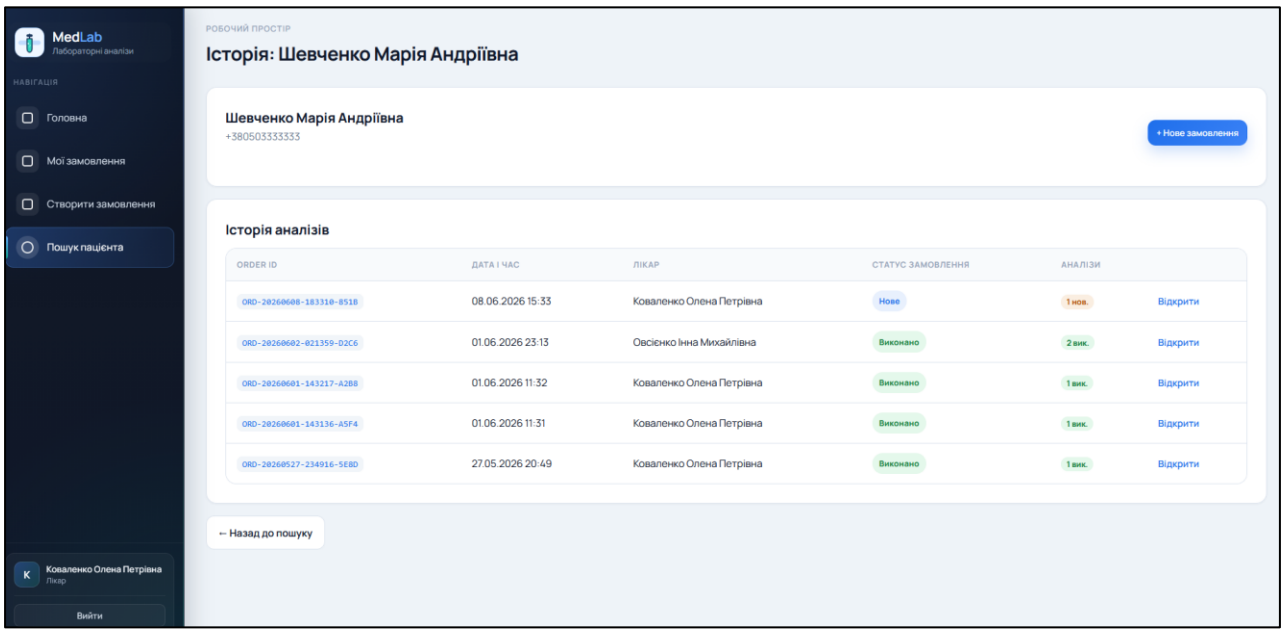


Рисунок 3.23 – Історія лабораторних досліджень пацієнта

Також лікар може переглядати результати завершених досліджень для встановлення діагнозу та контролю стану пацієнта.

3.5.4 Робота лаборанта

Після авторизації лаборант отримує доступ до черги лабораторних замовлень із зазначенням номера, пацієнта та статусу виконання (рис. 3.24). Для початку роботи лаборант обирає необхідне замовлення зі списку. Після відкриття замовлення система відображає перелік призначених аналізів та параметрів, які необхідно заповнити (рис. 3.25).

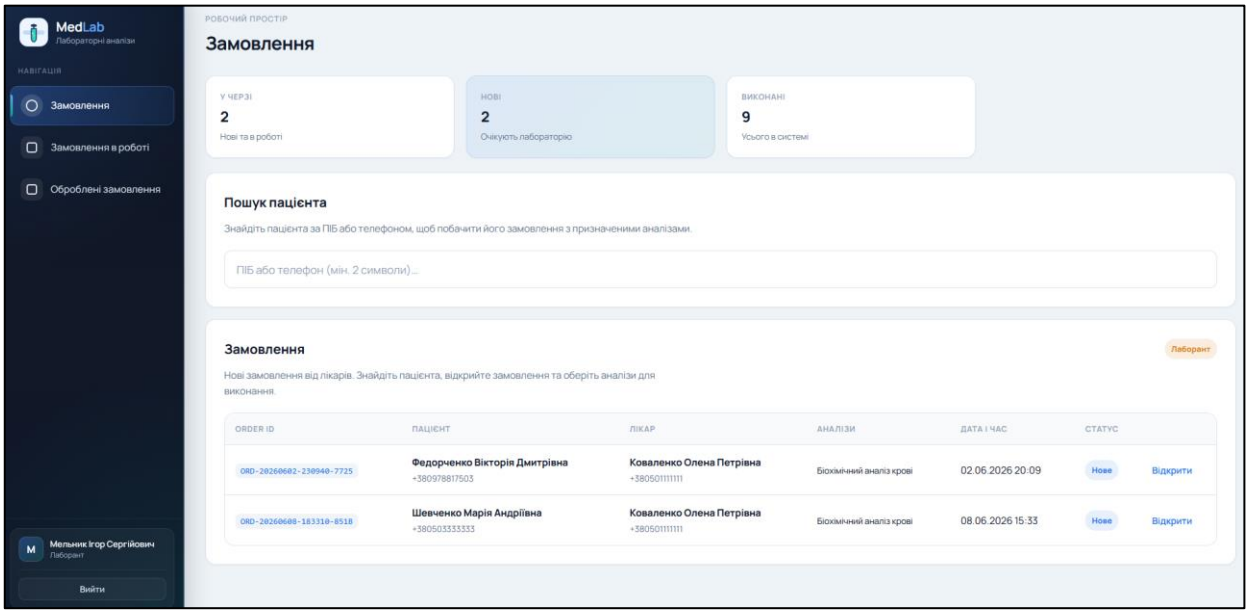


Рисунок 3.24 – Черга лабораторних замовлень

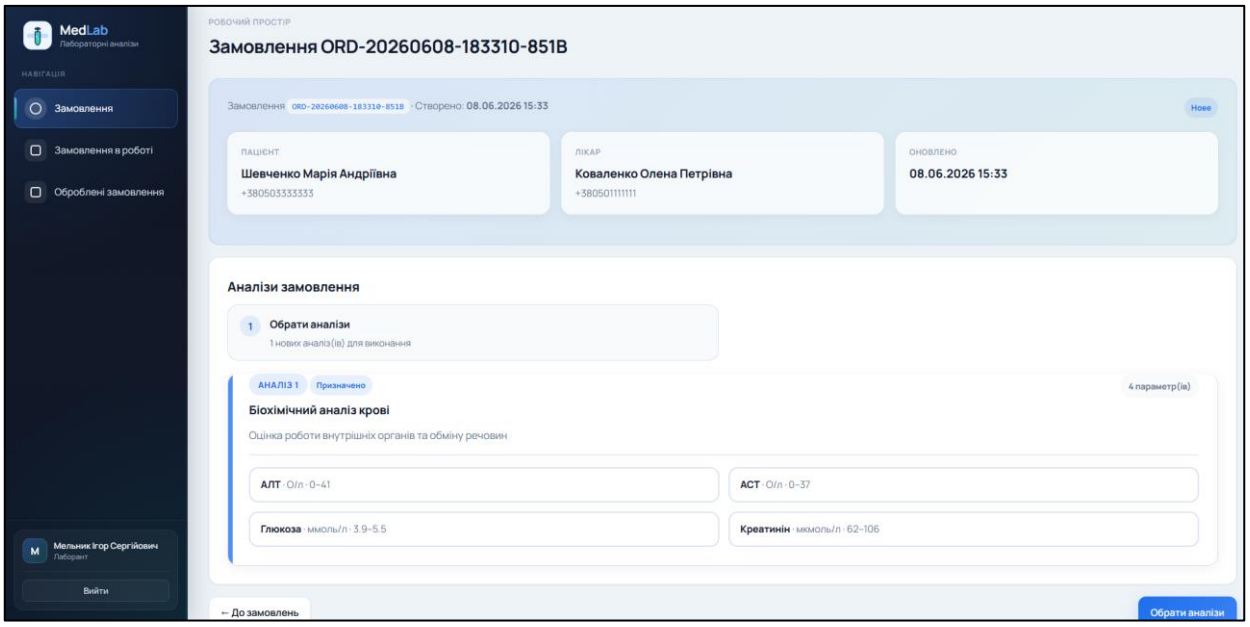


Рисунок 3.25 – Перегляд лабораторного замовлення лаборантом

Після проведення дослідження лаборант вносить отримані результати у відповідні поля форми. Для кожного параметра вказується значення показника, а також за потреби можуть бути додані додаткові коментарі (рис. 3.26).

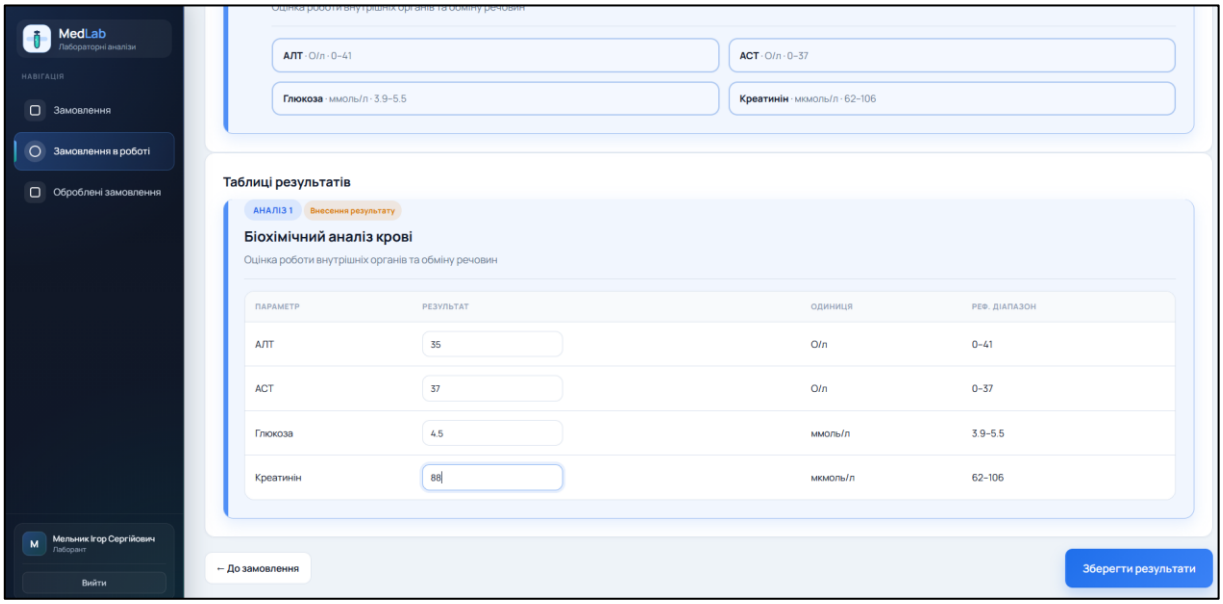


Рисунок 3.26 – Внесення результатів аналізів

Після перевірки введених даних результати зберігаються в базі даних і стають доступними лікарю та пацієнту (рис. 3.27).

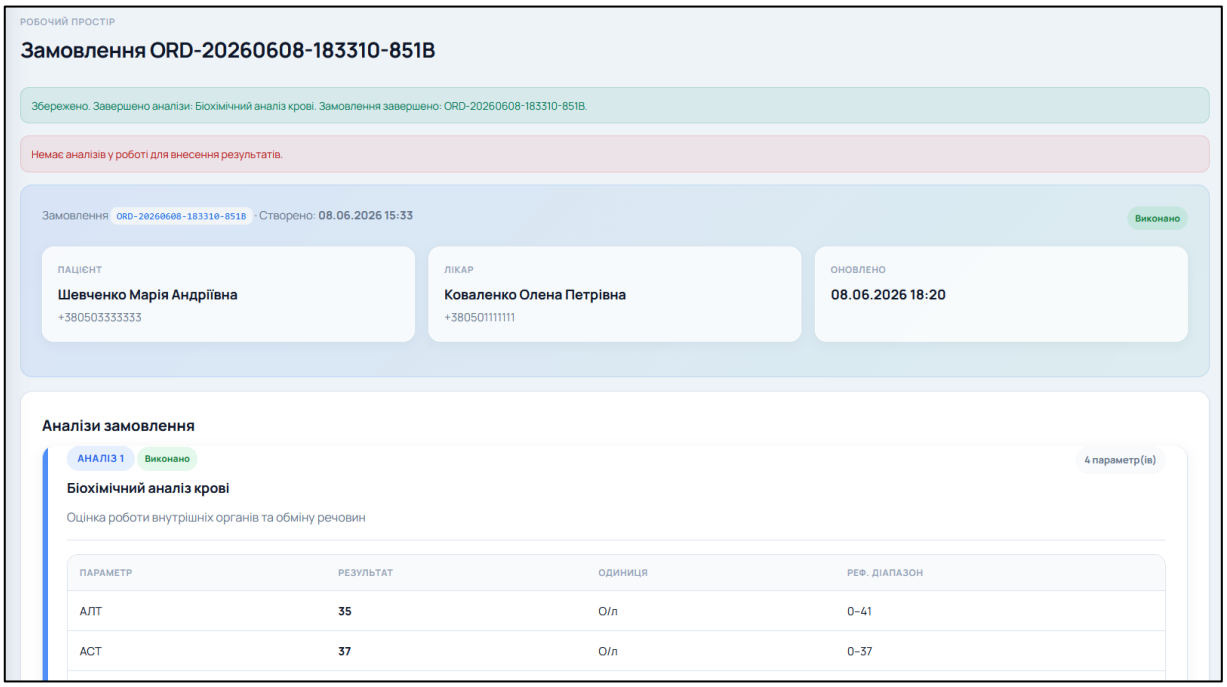


Рисунок 3.27 – Успішне збереження результатів аналізів

3.5.5 Робота адміністратора

Після авторизації адміністратор отримує доступ до функцій керування користувачами та контролю роботи системи (рис. 3.28).

У розділі «Користувачі» відображається список зареєстрованих користувачів із зазначенням основних даних та ролей (рис. 3.29). Ця сторінка дозволяє адміністратору переглядати облікові записи користувачів та контролювати розподіл ролей у системі.

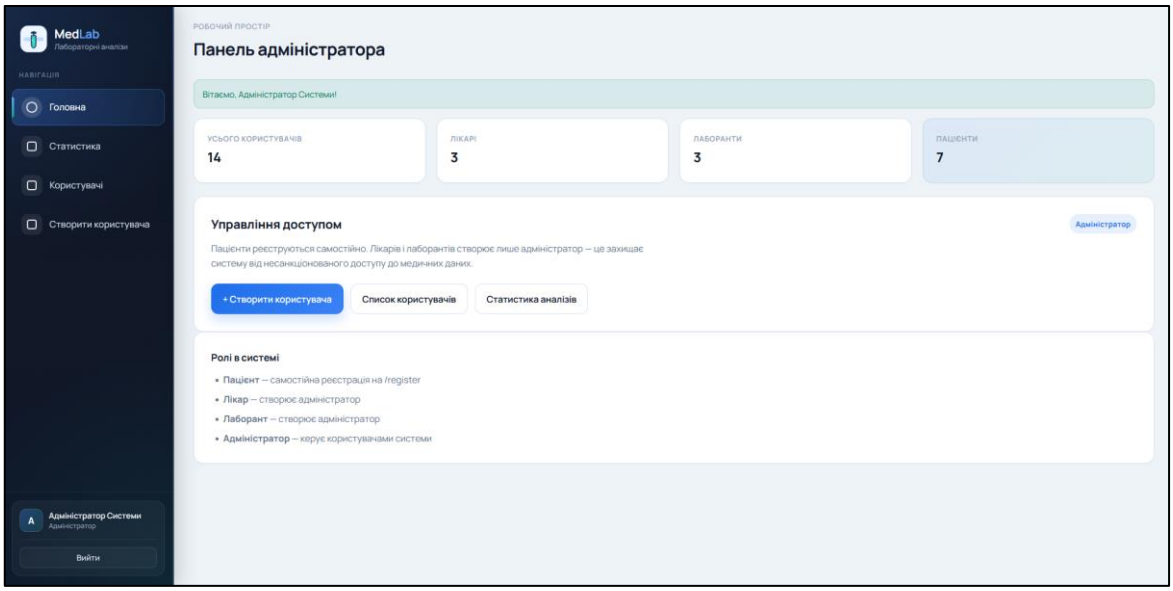


Рисунок 3.28 – Робоча сторінка адміністратора

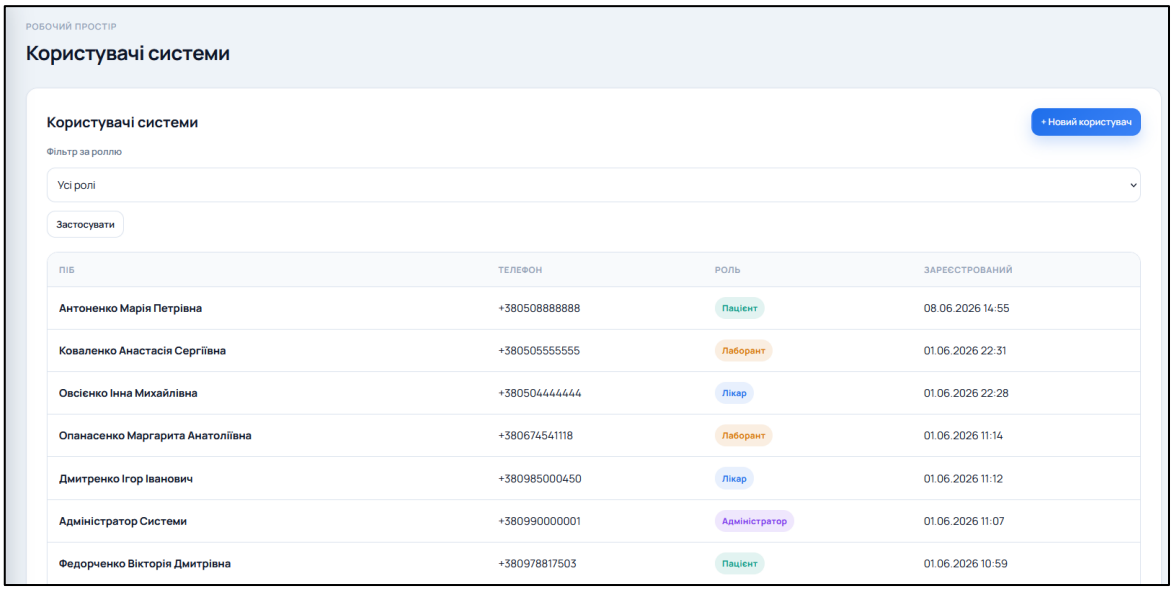


Рисунок 3.29 – Список користувачів системи

Адміністратор може створювати нові облікові записи, заповнивши відповідну форму та обравши роль користувача (рис. 3.30).

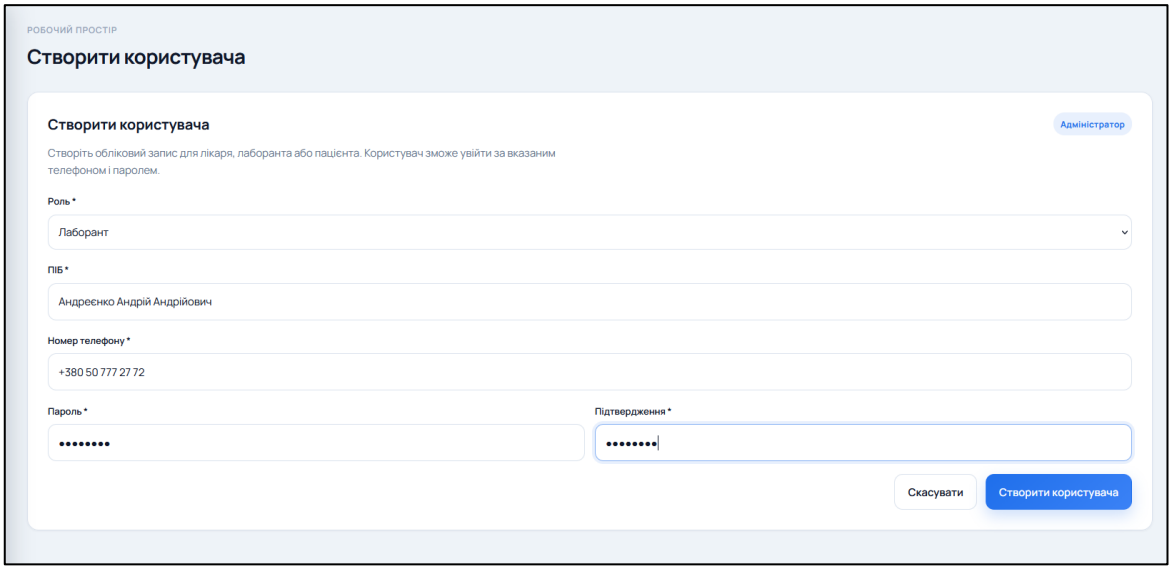


Рисунок 3.30 – Створення нового користувача

3.5.6 Перегляд статистики системи

Для перегляду основних показників роботи системи адміністратор використовує сторінку статистики (рис. 3.31).

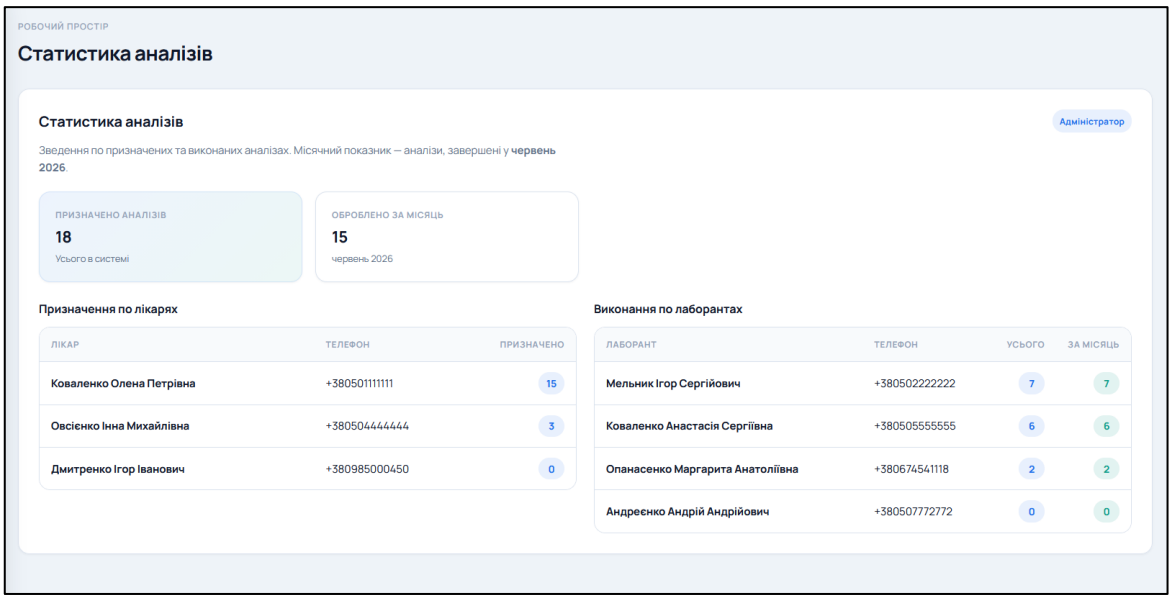


Рисунок 3.31 – Сторінка статистики системи

На сторінці представлено узагальнену статистику роботи системи, зокрема розподіл користувачів за ролями, кількість оформлених лабораторних замовлень та результати виконаних досліджень. Відображення цих показників дозволяє оперативно оцінювати поточний стан роботи лабораторії, аналізувати активність користувачів і контролювати основні процеси, що виконуються в системі

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено інформаційну систему для обліку лабораторних аналізів MedLab, яка автоматизує створення лабораторних замовлень, внесення та перегляд результатів досліджень.

Система підтримує ролі пацієнта, лікаря, лаборанта та адміністратора. Користувачі можуть переглядати результати аналізів, створювати замовлення, вносити результати досліджень і керувати роботою системи.

Для реалізації програмного продукту використано PHP 8.3, MySQL 8.4, Nginx, Docker, HTML5, CSS3, JavaScript та Tailwind CSS, що забезпечило створення сучасного веборієнтованого застосунку.

Відповідно до поставленої мети та задач роботи було сформовано функціональні та нефункціональні вимоги до системи, проведено аналіз предметної області медичної лабораторії та існуючих програмних рішень. У процесі проєктування розроблено структуру бази даних, побудовано ER-модель та UML-діаграми основних процесів системи. Реалізовано веборієнтовану інформаційну систему з розмежуванням доступу між користувачами різних ролей та підготовлено керівництво користувача.

Таким чином, поставлену мету роботи досягнуто. Розроблена інформаційна система забезпечує автоматизацію процесів обліку лабораторних аналізів та може бути використана як основа для подальшого розвитку й впровадження в діяльність медичних лабораторій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Nixon R. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites : 7th Edition. Sebastopol : O'Reilly Media, 2025. 840 p.

2. Simon M. An Introduction to PHP: Learn PHP 8 to Create Dynamic Websites. New York : Apress, 2024. 473 p.

3. Dubois P. MySQL : 5th Edition. Boston : Addison-Wesley Professional, 2014. 1152 p.

4. Duckett J. HTML and CSS: Design and Build Websites. New York : Wiley, 2011. 490 p.

5. Meyer E., Weyl E. CSS: The Definitive Guide : 5th Edition. Sebastopol : O'Reilly Media, 2023. 1100 p.

6. Flanagan D. JavaScript: The Definitive Guide : 7th Edition. Sebastopol : O'Reilly Media, 2020. 704 p.

7. Freeman A. Pro JavaScript for Web Innovations. New York : Apress, 2023. 410 p.

8. Turnbull J. The Docker Book: Startup and Development in White Boxes. New York : Turnbull Press, 2018. 420 p.

9. Miell I., Sayers A. Docker in Practice : 2nd Edition. Shelter Island : Manning Publications, 2019. 512 p.

10. Pressman R., Maxim B. Software Engineering: A Practitioner's Approach : 9th Edition. New York : McGraw-Hill Education, 2019. 912 p.

11. World Health Organization. Digital Health. Geneva : WHO, 2024. URL: https://www.who.int/health-topics/digital-health#tab=tab_1 (дата звернення: 14.02.2026).

12. Laboratory Information Management Systems (LIMS) Overview / Lab Manager. New York, 2021. URL: https://www.labmanager.com/technology-management-product-life-cycle-20974 (дата звернення: 14.02.2026).

13. ISO 15189:2022. Medical laboratories – Requirements for quality and competence. Geneva : International Organization for Standardization, 2022. 66 p. URL: https://www.iso.org/standard/76677.html (дата звернення: 15.03.2026).

14. PHP Documentation. URL: https://www.php.net/docs.php (дата звернення: 25.03.2026).

15. PDO Manual. URL: https://www.php.net/manual/en/book.pdo.php (дата звернення: 25.03.2026).

16. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 01.03.2026).

17. Nginx Documentation. URL: https://nginx.org/en/docs/ (дата звернення: 01.03.2026).

18. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 02.03.2026).

19. Composer Documentation. URL: https://getcomposer.org/doc/ (дата звернення: 02.03.2026).

20. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs/installation/using-vite (дата звернення: 24.03.2026).

ДОДАТОК А

ApiController

```
<?php

declare(strict_types=1);

namespace App\Controllers;

use App\Database;
use App\View;

final class ApiController extends Controller
{
    public function health(): void
    {
        header('Content-Type: application/json; charset=utf-8');

        $response = [
            'application' => 'System For Laboratory Test',
            'status' => 'ok',
            'environment' => getenv('APP_ENV') ?: 'local',
            'timestamp' => date('c'),
            'database' => [
                'connected' => false,
                'message' => null,
            ],
        ];

        try {
            $pdo = Database::connection();
            $version = $pdo->query('SELECT VERSION() AS version')->fetch();
            $tables = $pdo->query('SHOW TABLES')->fetchAll(\PDO::FETCH_COLUMN);

            $response['database'] = [
                'connected' => true,
                'version' => $version['version'] ?? null,
                'tables_count' => count($tables),
            ];
        } catch (\Throwable $exception) {
            http_response_code(503);
            $response['status'] = 'error';
            $response['database']['message'] = $exception->getMessage();
        }

        echo json_encode($response, JSON_UNESCAPED_UNICODE | JSON_PRETTY_PRINT);
    }
}
```

ДОДАТОК Б

OrderService

```
<?php

declare(strict_types=1);

namespace App;

final class OrderService
{
    public function __construct(
        private readonly OrderRepository $orders,
        private readonly AnalysisRepository $analysis,
        private readonly UserRepository $users
    ) {
    }

    /**
     * @param array<string, mixed> $input
     * @return array{order?: array<string, mixed>, errors?: array<string, string>}
     */
    public function createFromRequest(int $doctorId, array $input): array
    {
        $errors = [];

        $patientId = (int) ($input['patient_id'] ?? 0);
        $comment = trim((string) ($input['doctor_comment'] ?? ''));
        $selectedTypes = array_map('intval', (array) ($input['analysis_types'] ?? []));
        $selectedParameters = (array) ($input['parameters'] ?? []);

        $patient = $this->users->findPatientById($patientId);

        if ($patient === null) {
            $errors['patient_id'] = 'Оберіть пацієнта через пошук.';
        }

        if ($selectedTypes === []) {
            $errors['analysis_types'] = 'Оберіть хоча б один тип аналізу.';
        }

        $parametersByType = [];

        foreach ($selectedTypes as $typeId) {
            $typeParameterIds = array_map('intval', (array) ($selectedParameters[$typeId] ?? []));

            if ($typeParameterIds === []) {
                $errors["parameters_{$typeId}"] = 'Оберіть хоча б один параметр для аналізу.';
                continue;
            }

            $validated = [];

            foreach ($typeParameterIds as $parameterId) {
                if (!$this->analysis->parameterBelongsToType($parameterId, $typeId)) {
                    $errors['parameters'] = 'Обрано некоректні параметри аналізу.';
                    break 2;
```

```
            }

            $validated[] = $parameterId;
        }

        $parametersByType[$typeId] = array_values(array_unique($validated));
    }

    if ($errors !== []) {
        return ['errors' => $errors];
    }

    $order = $this->orders->create($doctorId, $patientId, $parametersByType, $comment);

    return ['order' => $order];
  }
}
```

ДОДАТОК В

UserRepository

```
<?php

declare(strict_types=1);

namespace App;

use PDO;

final class UserRepository
{
    public function __construct(private readonly PDO $db)
    {
    }

    public function findByPhone(string $phone): ?array
    {
        $stmt = $this->db->prepare(
            'SELECT id, phone, full_name, role, password_hash, created_at
             FROM users WHERE phone = :phone LIMIT 1'
        );
        $stmt->execute(['phone' => $phone]);

        $user = $stmt->fetch();

        return $user ?: null;
    }

    public function findById(int $id): ?array
    {
        $stmt = $this->db->prepare(
            'SELECT id, phone, full_name, role, created_at
             FROM users WHERE id = :id LIMIT 1'
        );
        $stmt->execute(['id' => $id]);

        $user = $stmt->fetch();

        return $user ?: null;
    }

    public function phoneExists(string $phone): bool
    {
        return $this->findByPhone($phone) !== null;
    }

    public function create(string $phone, string $fullName, string $passwordHash, string $role = 'patient'): int
    {
        $allowedRoles = ['patient', 'doctor', 'lab_technician', 'admin'];

        if (!in_array($role, $allowedRoles, true)) {
            throw new \InvalidArgumentException('Invalid user role.');
        }

        $stmt = $this->db->prepare(
```

```
            'INSERT INTO users (phone, full_name, role, password_hash)
             VALUES (:phone, :full_name, :role, :password_hash)'
        );
        $stmt->execute([
            'phone' => $phone,
            'full_name' => $fullName,
            'role' => $role,
            'password_hash' => $passwordHash,
        ]);

        return (int) $this->db->lastInsertId();
    }

    public function findPatientById(int $id): ?array
    {
        $stmt = $this->db->prepare(
            'SELECT id, phone, full_name, role, created_at
             FROM users
             WHERE id = :id AND role = :role
             LIMIT 1'
        );
        $stmt->execute([
            'id' => $id,
            'role' => 'patient',
        ]);

        $user = $stmt->fetch();

        return $user ?: null;
    }

    /** @return list<array<string, mixed>> */
    public function searchPatients(string $query, int $limit = 20): array
    {
        $query = $this->normalizeSearchQuery($query);

        if ($query === '') {
            return $this->getRecentPatients($limit);
        }

        if (mb_strlen($query) < 2) {
            return [];
        }

        $conditions = ['u.role = :role'];
        $params = [':role' => 'patient'];
        $filters = [];

        $filters[] = 'u.full_name LIKE :name';
        $params[':name'] = '%' . $query . '%';

        $phoneDigits = PhoneHelper::digits($query);

        if (strlen($phoneDigits) >= 3) {
            $filters[] = 'REPLACE(REPLACE(REPLACE(u.phone, "+", ""), " ", ""), "-", "") LIKE :phone_digits';
            $params[':phone_digits'] = '%' . $phoneDigits . '%';
        }

        if (strlen($phoneDigits) >= 9) {
            $lastNine = substr($phoneDigits, -9);
            $filters[] = 'u.phone LIKE :phone_last9';
            $params[':phone_last9'] = '%' . $lastNine;
        }
```

```
    $normalizedPhone = PhoneHelper::normalize($query);

    if (preg_match('/^\+380\d{9}$/', $normalizedPhone)) {
        $filters[] = 'u.phone = :exact_phone';
        $params[':exact_phone'] = $normalizedPhone;
    }

    $where = $conditions[0] . ' AND (' . implode(' OR ', $filters) . ')';

    $sql = "SELECT u.id, u.phone, u.full_name, u.created_at,
                (SELECT COUNT(*) FROM orders o WHERE o.patient_id = u.id) AS orders_count
            FROM users u
            WHERE {$where}
            ORDER BY u.full_name
            LIMIT :limit";

    $stmt = $this->db->prepare($sql);

    foreach ($params as $key => $value) {
        $stmt->bindValue($key, $value);
    }

    $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
    $stmt->execute();

    return $stmt->fetchAll();
}

/** @return list<array<string, mixed>> */
public function getRecentPatients(int $limit = 20): array
{
    $stmt = $this->db->prepare(
        'SELECT u.id, u.phone, u.full_name, u.created_at,
                (SELECT COUNT(*) FROM orders o WHERE o.patient_id = u.id) AS orders_count
         FROM users u
         WHERE u.role = :role
         ORDER BY u.created_at DESC
         LIMIT :limit'
    );
    $stmt->bindValue(':role', 'patient');
    $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
    $stmt->execute();

    return $stmt->fetchAll();
}

private function normalizeSearchQuery(string $query): string
{
    $query = trim($query);
    $query = preg_replace('/\s+/u', ' ', $query) ?? $query;

    return $query;
}

/** @return list<array<string, mixed>> */
public function findAllUsers(?string $role = null, int $limit = 100): array
{
    $sql = 'SELECT id, phone, full_name, role, created_at FROM users';
    $params = [];

    if ($role !== null && $role !== '') {
        $sql .= ' WHERE role = :role';
        $params['role'] = $role;
```

```
        }

        $sql .= ' ORDER BY created_at DESC LIMIT :limit';

        $stmt = $this->db->prepare($sql);

        foreach ($params as $key => $value) {
            $stmt->bindValue($key, $value);
        }

        $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
        $stmt->execute();

        return $stmt->fetchAll();
    }

    /** @return array<string, int> */
    public function countByRole(): array
    {
        $stmt = $this->db->query(
            'SELECT role, COUNT(*) AS total FROM users GROUP BY role'
        );

        $counts = [
            'admin' => 0,
            'doctor' => 0,
            'lab_technician' => 0,
            'patient' => 0,
            'total' => 0,
        ];

        foreach ($stmt->fetchAll() as $row) {
            $counts[$row['role']] = (int) $row['total'];
            $counts['total'] += (int) $row['total'];
        }

        return $counts;
    }
}
```

ДОДАТОК Г

Router

```
<?php

declare(strict_types=1);
namespace App;

final class Router
{
   /** @var array<string, list<array{pattern:string, handler:callable, regex:string}>> */
   private array $routes = [];

   public function get(string $path, callable $handler): self
   {
      $this->addRoute('GET', $path, $handler);

      return $this;
   }
   public function post(string $path, callable $handler): self
   {
      $this->addRoute('POST', $path, $handler);

      return $this;
   }

   public function dispatch(string $method, string $uri): void
   {
      $path = $this->normalize(parse_url($uri, PHP_URL_PATH) ?: '/');
      $match = $this->match($method, $path);

      if ($match === null) {
         http_response_code(404);
         View::render('errors.404', ['title' => 'Сторінку не знайдено'], 'auth');

         return;
      }
      $match['handler'](...array_values($match['params']));
   }

   private function addRoute(string $method, string $path, callable $handler): void
   {
      $normalized = $this->normalize($path);
      $regex = preg_replace('/\{([a-zA-Z_][a-zA-Z0-9_]*)\}/', '(?P<$1>[^/]+)', $normalized);
      $regex = '#^' . $regex . '$#';

      $this->routes[$method][] = [
         'pattern' => $normalized,
         'handler' => $handler,
         'regex' => $regex,
      ];
   }
   /** @return array{handler:callable, params:array<string, string>}|null */
   private function match(string $method, string $path): ?array
   {
      foreach ($this->routes[$method] ?? [] as $route) {
         if ($route['pattern'] === $path) {
```

```
                return [
                    'handler' => $route['handler'],
                    'params' => [],
                ];
            }
            if (!str_contains($route['pattern'], '{')) {
                continue;
            }

            if (preg_match($route['regex'], $path, $matches)) {
                $params = [];

                foreach ($matches as $key => $value) {
                    if (is_string($key)) {
                        $params[$key] = $value;
                    }
                }

                return [
                    'handler' => $route['handler'],
                    'params' => $params,
                ];
            }
        }

        return null;
    }
    private function normalize(string $path): string
    {
        $path = '/' . trim($path, '/');

        return $path === '/' ? '/' : rtrim($path, '/');
    }
}
```