

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебдодаток стоматологічної клініки»**

Виконав студент групи ВТп-22-1  
БОЖКО Кирило Дмитрович

Керівник роботи:  
ДУНЄВ Сергій Сергійович

Рецензент:  
ЄФРЕМОВ Юрій Миколайович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**«ЗАТВЕРДЖУЮ»**

Зав. кафедри інженерії  
програмного забезпечення  
Тетяна Вакалюк  
**«16» липня 2026 р.**

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **БОЖКО Кирило Дмитрович**

Керівник роботи: **ДУНЄВ Сергій Сергійович**

Тема роботи: **«Вебдодаток стоматологічної клініки»**,

затверджена Наказом закладу вищої освіти від **«16» липня 2026 р., №396/с**

Вихідні дані для роботи: **Технічне завдання на розробку вебдодатка стоматологічної клініки з автоматизацією онлайн-запису пацієнтів, ведення медичних карток та управління персоналом клініки.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант | Завдання видав | Завдання прийняв |
|--------|-------------|----------------|------------------|
| 1      | Дунєв С.С.  | 16.07.2026р.   | 16.07.2026р.     |
| 2      | Дунєв С.С.  | 16.07.2026р.   | 16.07.2026р.     |
| 3      | Дунєв С.С.  | 16.07.2026р.   | 16.07.2026р.     |

## РЕФЕРАТ

Кваліфікаційна робота складається з вебдодатка стоматологічної клініки та пояснювальної записки. Пояснювальна записка містить 80 сторінок, 21 ілюстрацію та 21 таблицю.

Метою роботи є розробка вебдодатка для стоматологічної клініки, що автоматизує онлайн-запис пацієнтів, ведення електронної медичної документації та управління роботою персоналу клініки.

У роботі використано методи об'єктно-орієнтованого проектування, системного аналізу, функціонального тестування та вебпрограмування. Реалізовано повнофункціональний вебдодаток на стеку Next.js 16, TypeScript, PostgreSQL та Prisma з підтримкою чотирьох ролей (пацієнт, лікар, асистент, адміністратор).

Отримані результати можуть бути використані у приватних стоматологічних клініках для автоматизації основних бізнес-процесів без придбання дорогих іноземних програмних рішень.

**КЛЮЧОВІ СЛОВА:** ВЕБДОДАТОК, СТОМАТОЛОГІЧНА КЛІНІКА, NEXT.JS, POSTGRESQL, АВТОМАТИЗАЦІЯ.

|           |               |          |        |      |                                    |                                            |      |        |
|-----------|---------------|----------|--------|------|------------------------------------|--------------------------------------------|------|--------|
|           |               |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ           |                                            |      |        |
| Змн.      | Арк.          | № докум. | Підпис | Дата |                                    |                                            |      |        |
| Розроб.   | Божко К. Д.   |          |        |      | Вебдодаток стоматологічної клініки | Літ.                                       | Арк. | Аркуші |
| Перевір.  |               |          |        |      |                                    |                                            | 3    | 81     |
| Керівник  | Дунєв С. С.   |          |        |      |                                    | Житомирська політехніка,<br>група ВТп-22-1 |      |        |
| Рецензент | Єфремов Ю. М. |          |        |      |                                    |                                            |      |        |
| Зав. каф. | Вакалюк Т.А.  |          |        |      |                                    |                                            |      |        |

## ABSTRACT

The bachelor's thesis consists of a dental clinic web application and an explanatory note. The explanatory note contains 80 pages, 21 illustrations and 21 tables.

The goal of the work is to develop a web application for a dental clinic that automates online patient appointment booking, electronic medical record keeping and clinic staff management.

The work employs object-oriented design, system analysis, functional testing and web programming methods. A fully functional web application has been implemented using Next.js 16, TypeScript, PostgreSQL and Prisma, supporting four user roles: patient, doctor, assistant and administrator.

The obtained results can be applied in private dental clinics to automate core business processes without purchasing expensive foreign software solutions.

KEYWORDS: WEB APPLICATION, DENTAL CLINIC, NEXT.JS, POSTGRESQL, AUTOMATION.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 4    |

## ЗМІСТ

|                                                                                                  |    |
|--------------------------------------------------------------------------------------------------|----|
| РЕФЕРАТ .....                                                                                    | 3  |
| ABSTRACT .....                                                                                   | 4  |
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....                                                                   | 7  |
| ВСТУП .....                                                                                      | 9  |
| РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ РОЗРОБКИ ВЕБДОДАТКІВ ДЛЯ<br>СТОМАТОЛОГІЧНИХ КЛІНІК.....               | 12 |
| 1.1. Постановка задачі.....                                                                      | 12 |
| 1.2. Аналіз конкурентів та аналогів вебдодатка.....                                              | 14 |
| 1.3. Обґрунтування вибору інструментальних засобів та вимоги до<br>апаратного забезпечення ..... | 20 |
| Висновки до першого розділу.....                                                                 | 24 |
| РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКА СТОМАТОЛОГІЧНОЇ<br>КЛІНІКИ .....                               | 26 |
| 2.1. Визначення варіантів використання та об'єктно-орієнтованої структури<br>системи .....       | 26 |
| 2.2. Розробка бази даних системи.....                                                            | 32 |
| 2.3. Проєктування та реалізація алгоритмів роботи системи .....                                  | 39 |
| 2.4. Реалізація функціоналу вебдодатка стоматологічної клініки .....                             | 43 |
| Висновки до другого розділу .....                                                                | 45 |
| РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ<br>СТОМАТОЛОГІЧНОЇ КЛІНІКИ.....              | 48 |
| 3.1. Порядок встановлення та налаштування параметрів системи .....                               | 48 |
| 3.2. Структура інтерфейсу вебдодатка стоматологічної клініки .....                               | 52 |
| 3.3. Тестування роботи програмного комплексу .....                                               | 62 |

|                                   |    |
|-----------------------------------|----|
| Висновки до третього розділу..... | 68 |
| ВИСНОВКИ.....                     | 69 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....   | 71 |
| ДОДАТКИ.....                      | 74 |

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 6    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс (Application Programming Interface)

БД – база даних

CDN – мережа доставки вмісту (Content Delivery Network)

CRUD – операції створення, читання, оновлення та видалення (Create, Read, Update, Delete)

CSS – каскадні таблиці стилів (Cascading Style Sheets)

FDI – Всесвітня федерація стоматологів (Fédération Dentaire Internationale)

HTTP – протокол передачі гіпертексту (Hypertext Transfer Protocol)

HTTPS – захищений протокол передачі гіпертексту (HTTP Secure)

JWT – JSON-токен авторизації (JSON Web Token)

JSON – формат обміну даними (JavaScript Object Notation)

MVC – архітектурний патерн Модель–Вигляд–Контролер (Model–View–Controller)

ORM – об'єктно-реляційний маппер (Object-Relational Mapper)

ПЗ – програмне забезпечення

REST – архітектурний стиль передачі репрезентативного стану (Representational State Transfer)

SaaS – програмне забезпечення як послуга (Software as a Service)

SQL – мова структурованих запитів (Structured Query Language)

SSR – серверний рендеринг (Server-Side Rendering)

СУБД – система управління базами даних

SVG – масштабована векторна графіка (Scalable Vector Graphics)

UI – інтерфейс користувача (User Interface)

UML – уніфікована мова моделювання (Unified Modeling Language)

URL – уніфікований локатор ресурсів (Uniform Resource Locator)

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 7    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

UX – досвід користувача (User Experience)

XSS – міжсайтовий скриптинг (Cross-Site Scripting)

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 8    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ВСТУП

Цифровізація сфери охорони здоров'я належить до пріоритетних напрямів розвитку інформаційних технологій в Україні. Запровадження електронної системи охорони здоров'я змінило принципи ведення медичної документації у первинній ланці, проте приватна стоматологія залишилася переважно поза межами цих процесів: питання автоматизації кожен заклад вирішує самостійно, орієнтуючись на власні фінансові можливості. Наявні на ринку рішення, зокрема Dentrux, Open Dental та Curve Dental [1–3], орієнтовані передусім на північноамериканський ринок і великі мережі клінік, мають ліцензійну або передплатну модель оплати та не передбачають української локалізації. Це створює потребу у доступному вебдодатку, адаптованому під потреби невеликої приватної клініки.

Значна частина невеликих стоматологічних кабінетів дотепер поєднує паперові медичні картки, телефонний запис пацієнтів і розрізнені електронні таблиці, які не утворюють єдиного інформаційного простору закладу. Через це дані про одного пацієнта зберігаються фрагментарно, і лікар не має цілісної картини попереднього лікування, а відновлення втраченої інформації потребує повторного опитування або дублювання обстежень. Запис на прийом залишається залежним від присутності адміністратора на робочому місці, тому пацієнт не може самостійно обрати зручний час поза межами робочого дня клініки. Розклад лікарів формується вручну, що призводить до накладання прийомів, простоїв кабінетів і помилок під час планування відпусток. Робота медичного асистента, який супроводжує втручання та фіксує його перебіг, здебільшого взагалі не відображена в інформаційній системі, хоча саме ці записи мають найбільше значення для подальшого контролю якості лікування. Окремим невирішеним питанням залишається зберігання дентальних фотознімків, оскільки локальні файлові сховища не забезпечують ані надійності, ані швидкого доступу з робочого місця лікаря.

Сучасний стан вебтехнологій дозволяє усунути перелічені обмеження.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 9    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Перехід від настільних застосунків до вебдодатків знімає потребу встановлювати програмне забезпечення на кожному робочому місці та забезпечує доступ до системи з будь-якого пристрою, що має браузер. Застосування серверного рендерингу, рольового розмежування доступу та токенної автентифікації дає змогу поєднати швидкодію інтерфейсу з вимогами до захисту персональних медичних даних, а орієнтація на характеристики якості програмного забезпечення, визначені стандартом ISO/IEC 25010 [24], забезпечує супроводжуваність системи та її придатність до подальшого розширення.

Актуальність обраної теми полягає у тому, що автоматизація основних бізнес-процесів стоматологічної клініки на основі веборієнтованої архітектури дає змогу об'єднати розрізнені дані в єдину електронну медичну картку, зробити запис на прийом доступним цілодобово, унеможливити накладання прийомів завдяки централізованому веденню розкладу, а також уперше вивести роботу медичного асистента та операційний протокол у цифрову площину. Веборієнтований формат системи робить ці можливості доступними для невеликих приватних закладів, які не спроможні придбати комерційні іноземні рішення.

Метою випускної кваліфікаційної роботи бакалавра є проєктування та розробка вебдодатка для стоматологічної клініки, що автоматизує онлайн-запис пацієнтів, ведення електронної медичної документації, управління розкладом лікарів і роботою медичних асистентів.

Для досягнення поставленої мети у роботі передбачено виконання комплексу взаємопов'язаних завдань. Насамперед необхідно провести аналіз предметної області стоматологічної клініки та дослідити наявні аналоги програмних продуктів для автоматизації медичних закладів, а також обґрунтувати вибір технологічного стеку та архітектурних рішень з урахуванням вимог до продуктивності й масштабованості системи. Наступним етапом є проєктування об'єктно-орієнтованої структури системи та схеми

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 10   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

реляційної бази даних, розроблення алгоритмів роботи основних модулів, зокрема авторизації користувачів, онлайн-запису на прийом, ведення дентальних карток, планування розкладу та формування операційних протоколів. Далі передбачено реалізацію повнофункціонального вебдодатка з чотирма робочими просторами для ролей пацієнта, лікаря, асистента та адміністратора. Завершальним етапом є проведення функціонального тестування реалізованого програмного комплексу методом чорного ящика, забезпечення механізмів захисту даних і валідації вхідної інформації та підготовка технічної документації.

Об'єктом дослідження є процес автоматизації управління стоматологічною клінікою за допомогою веборієнтованих інформаційних систем. Предметом дослідження є методи та засоби проєктування і реалізації вебдодатків для медичних закладів з використанням сучасних full-stack-фреймворків та хмарних сервісів. У роботі використано методи системного аналізу, порівняльного аналізу програмних рішень, об'єктно-орієнтованого проєктування, UML-моделювання, реляційного моделювання даних та функціонального тестування програмного забезпечення. Інформаційною базою роботи стали матеріали щодо веборієнтованих медичних інформаційних систем, клієнт-серверної архітектури та UML-моделювання, міжнародні стандарти якості й безпеки програмного забезпечення, а також офіційна документація обраних технологій.

Практичне значення роботи полягає у створенні готового до впровадження програмного продукту, який може бути використаний приватними стоматологічними кабінетами та клініками середнього розміру для автоматизації основних бізнес-процесів без придбання дорогих іноземних програмних рішень. Система забезпечує єдине сховище медичних даних, зменшує вплив людського фактора під час планування прийомів і надає кожній ролі окремий робочий простір з інтуїтивно зрозумілим інтерфейсом. Архітектура системи допускає подальше розширення функціоналу.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 11   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

# РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ РОЗРОБКИ ВЕБДОДАТКІВ ДЛЯ СТОМАТОЛОГІЧНИХ КЛІНІК

## 1.1. Постановка задачі

Сучасна стоматологічна клініка щоденно обробляє значні обсяги інформації: персональні дані пацієнтів, графіки роботи лікарів, результати обстежень, плани лікування, дані про проведені маніпуляції, рентгенівські знімки, рекомендації та призначення. Ручне опрацювання таких даних, ведення паперових медичних карток та узгодження часу прийому за телефоном суттєво уповільнює роботу персоналу, призводить до помилок у розкладі, втрати інформації та зниження якості обслуговування. Саме тому впровадження веборієнтованих рішень є необхідним кроком для підвищення ефективності роботи стоматологічних закладів.

Вебдодаток як форма реалізації системи був обраний з огляду на низку переваг: відсутність потреби встановлювати клієнтське програмне забезпечення, кросплатформеність (доступ з ПК, планшета чи смартфона), централізоване зберігання даних та можливість одночасної роботи кількох категорій користувачів — пацієнтів, лікарів, медичних асистентів та адміністратора клініки.

Для реалізації вебдодатка стоматологічної клініки необхідно розробити дев'ять взаємопов'язаних модулів. Модуль реєстрації та авторизації забезпечує розмежування ролей (пацієнт, лікар, асистент, адміністратор) і безпечно зберігання облікових даних з хешуванням паролів. Модуль онлайн-запису дозволяє пацієнту переглядати вільні слоти у тижневому графіку лікаря та бронювати їх. Модуль управління розкладом надає адміністратору інструменти для генерації слотів, управління відпустками лікарів та реагування на запити пацієнтів. Модуль електронних медичних карток реалізовано у формі інтерактивної дентальної карти (SVG-схема 32 зубів [30]) — лікар фіксує проблему, лікування, коментарі та фотознімки через хмарний сервіс Cloudinary [9]. Модуль медичного анамнезу зберігає медичну картку

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 12   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

пацієнта: групу крові, алергії, хронічні захворювання, препарати та попередні операції. Модуль операційних протоколів дозволяє асистенту вести хронологічний журнал подій під час прийому в режимі реального часу. Модуль рекомендацій та призначень дозволяє лікарю після прийому залишати персоналізовані поради пацієнту. Модуль обробки звернень фіксує запити від пацієнтів, які не знайшли вільного слота. Модуль адміністрування забезпечує керування лікарями та асистентами, перегляд статистики клініки, обробку запитів та управління розкладом [32].

Розробка вебдодатка стоматологічної клініки передбачає послідовне виконання кількох взаємопов'язаних етапів. Перший етап — аналіз предметної області, що включає вивчення особливостей роботи приватних стоматологічних кабінетів, виявлення типових сценаріїв взаємодії пацієнта, лікаря, асистента та адміністратора, а також формування переліку функціональних і нефункціональних вимог. На цьому ж етапі здійснюється огляд існуючих програмних рішень для стоматологічних клінік з метою визначення сильних та слабких сторін кожного з них.

Наступний етап — проектування архітектури системи. Передбачено побудову вебдодатка за клієнт-серверною моделлю з використанням підходу full-stack-фреймворку, де клієнтська частина та серверні API-маршрути реалізуються в межах єдиного застосунку. Такий підхід зменшує час розробки, спрощує підтримку та дозволяє більш ефективно ділитися типами між клієнтом і сервером.

На третьому етапі здійснюється проектування бази даних: визначаються основні сутності (користувачі, лікарі, асистенти, прийоми, слоти розкладу, дентальні записи, фото, протоколи, рекомендації, анамнез, запити, відпустки), будується ER-модель і виконується нормалізація схеми. Окрема увага приділяється захисту персональних та медичних даних, оскільки інформація про здоров'я пацієнтів належить до категорії чутливої.

Четвертий етап охоплює безпосередньо реалізацію програмного

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 13   |

забезпечення: розробку інтерфейсних компонентів, опис маршрутів API, створення сервісного шару бізнес-логіки, інтеграцію з базою даних та механізмами автентифікації. На завершальному етапі проводиться тестування системи у різних типових сценаріях та підготовка програмного продукту до експлуатації.

Результатом виконання кваліфікаційної роботи є функціонуючий вебдодаток для стоматологічної клініки, що автоматизує процес запису пацієнтів, веде електронну медичну картку, зберігає медичний анамнез та операційні протоколи прийомів, забезпечує формування рекомендацій та надає адміністратору повний інструментарій для керування роботою закладу. Система готова до експлуатації у реальному медичному закладі та може бути розширена додатковими модулями (наприклад, фінансовим обліком, інтеграцією з месенджерами, мобільним клієнтом).

## 1.2. Аналіз конкурентів та аналогів вебдодатка

Перш ніж переходити до проєктування власного програмного продукту, доцільно проаналізувати ринок наявних інформаційних систем для стоматологічних клінік. Це дозволяє виявити функціональні особливості найбільш популярних рішень, оцінити сильні і слабкі сторони, визначити очікування користувачів та сформулювати перелік вимог, на які слід орієнтуватися при розробці. Аналіз існуючих систем — обов'язковий етап інженерії програмного забезпечення, оскільки повторне винайдення вже реалізованих механізмів є економічно недоцільним, а ігнорування досвіду конкурентів призводить до помилок проєктування.

Серед найбільш поширених рішень на ринку стоматологічного програмного забезпечення можна виокремити три провідні платформи: Dentrix, Open Dental та Curve Dental. Розглянемо кожен з них детальніше.

Dentrix — одна з найбільш розповсюджених у Північній Америці систем для управління стоматологічними клініками, що належить компанії Henry Schein One. Платформа надає широкий набір інструментів: ведення електронних медичних карток, планування графіків роботи лікарів і кабінетів,

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 14   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

фінансовий облік, формування звітів для страхових компаній, інтеграцію з рентгенівським і діагностичним обладнанням. Dentrrix підтримує модуль нагадувань для пацієнтів про майбутні візити та функцію відправки SMS- і e-mail-повідомлень (рис. 1.1) [1].

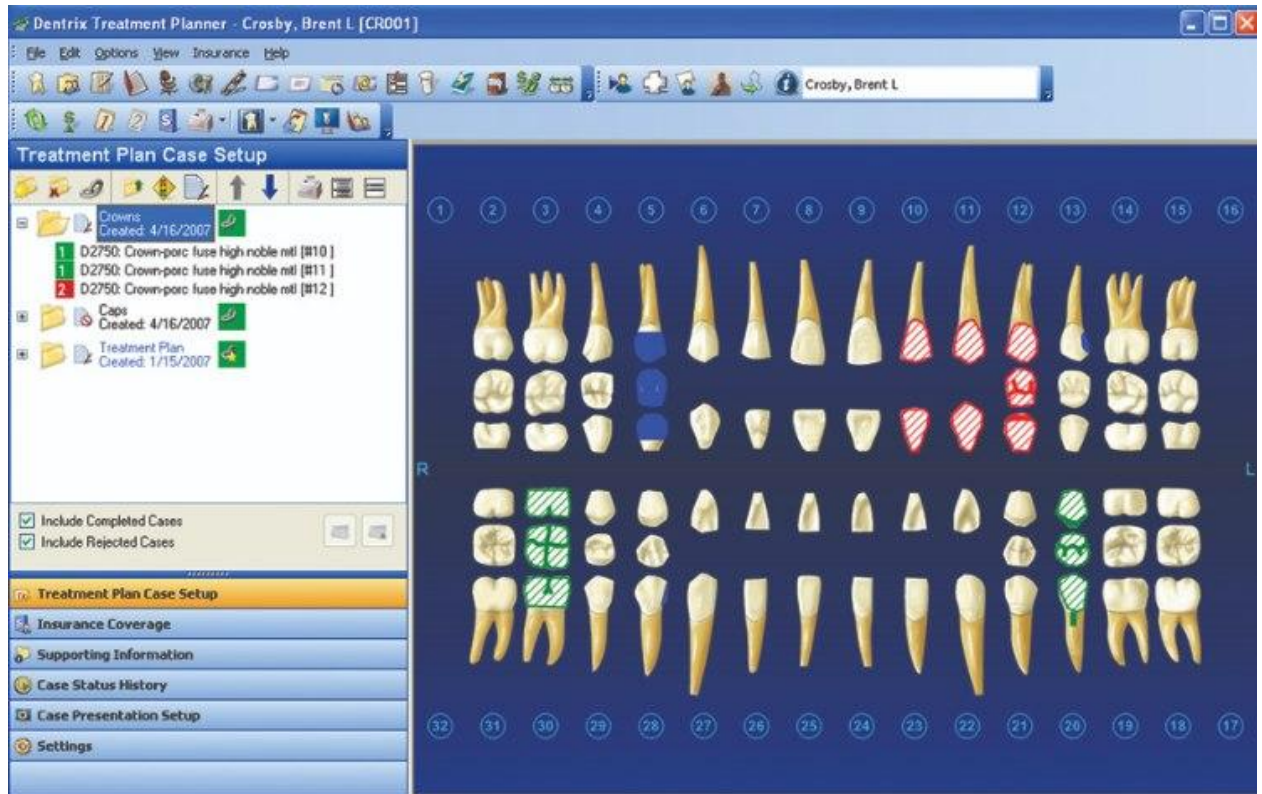


Рис. 1.1. Система управління стоматологічною клінікою Dentrrix

Серед переваг системи слід відзначити: глибокий функціонал для великих мереж клінік, тісну інтеграцію з обладнанням провідних виробників, наявність окремого мобільного додатка для лікарів, можливість роботи у локальній мережі чи у хмарі. До недоліків відносять: висока вартість ліцензії та обов'язкова щомісячна плата за підтримку, орієнтація передусім на американський ринок, складний інтерфейс із великою кількістю налаштувань, відсутність української локалізації та значне навантаження на апаратне забезпечення.

Open Dental — розповсюджується за моделлю open-source, тобто з відкритим вихідним кодом, що дає клінікам змогу самостійно адаптувати її під свої потреби. Система реалізована як настільний клієнт-серверний застосунок

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 15   |

із базою даних MySQL. Open Dental надає широкі можливості ведення карток пацієнтів, планування прийомів, контролю фінансів, обліку матеріалів та інтеграції з рентгенівським обладнанням (рис. 1.2) [2].

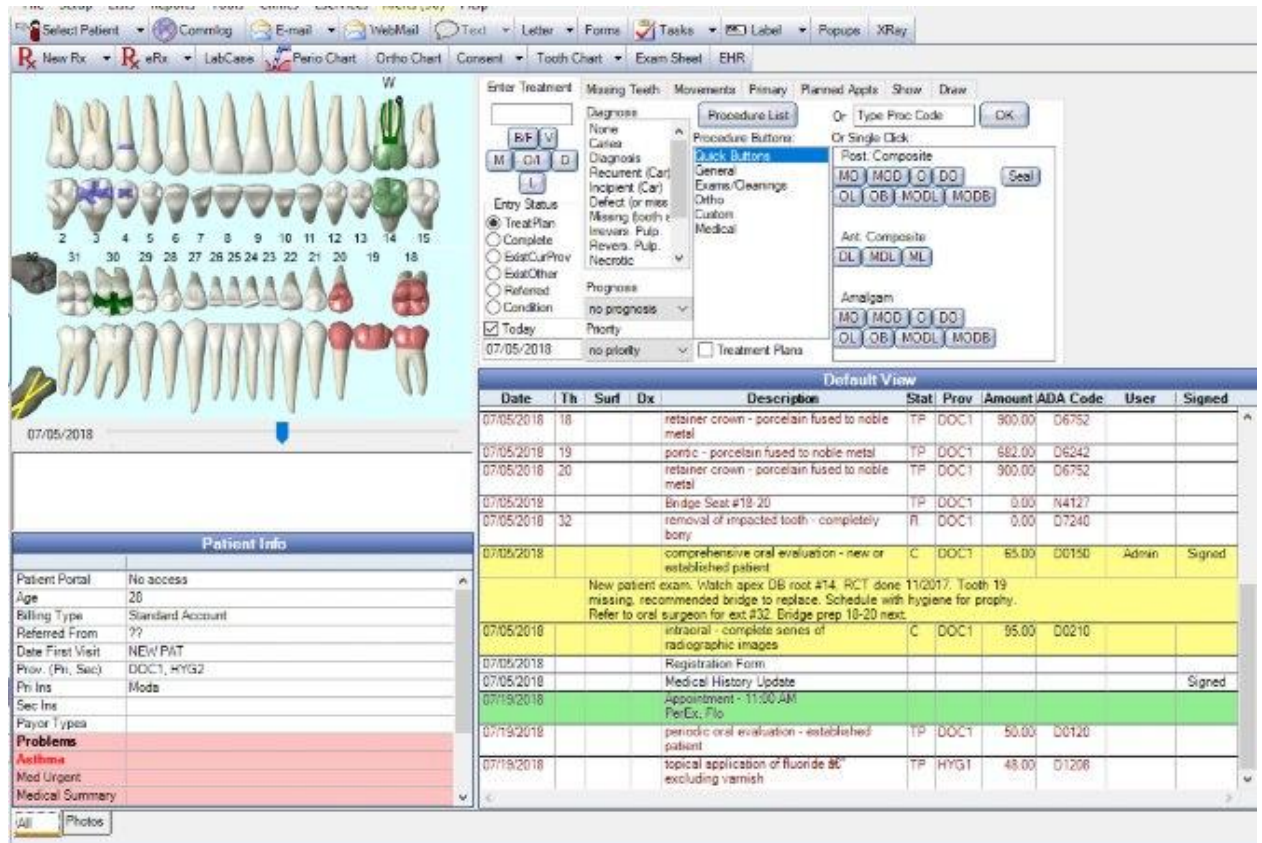


Рис. 1.2. Система управління стоматологічною клінікою Open Dental

До переваг Open Dental належать: прозорість коду, активна спільнота розробників, можливість самостійного хостингу. Водночас система має низку обмежень: відсутній зручний вебінтерфейс для пацієнтів, необхідність встановлення спеціалізованого клієнта на кожному робочому місці, потреба у штатному системному адміністраторі, морально застарілий інтерфейс та обмежена локалізація українською мовою.

Curve Dental — повністю хмарне рішення (SaaS), що дозволяє користувачам працювати через браузер без встановлення клієнтського програмного забезпечення. Curve Dental відомий зручним та сучасним інтерфейсом, продуманим UX, наявністю мобільної версії та розширеними інструментами візуалізації даних. Система пропонує функції онлайн-запису,

нагадування пацієнтам, ведення електронної медичної документації та фінансову аналітику (рис. 1.3) [3].



Рис. 1.3. Хмарна система управління стоматологічною клінікою Curve Dental

Перевагами Curve Dental є: відсутність необхідності в локальному сервері, регулярні автоматичні оновлення, інтуїтивний інтерфейс. Серед недоліків: підписочна модель оплати з прив'язкою до кількості робочих місць, відсутність можливості модифікувати логіку системи, доступ до повного функціоналу лише за вищим тарифним планом та відсутність української локалізації.

Усі розглянуті системи орієнтовані переважно на іноземний ринок та потребують суттєвої адаптації під українські реалії. Для невеликої приватної стоматологічної клініки в Україні впровадження таких рішень виявляється економічно невиправданим. Це зумовлює доцільність розробки власного вебдодатка, який поєднає сильні сторони існуючих платформ та усуне характерні для них недоліки.

При виборі критеріїв для порівняльного аналізу враховувались передусім практичні потреби стоматологічної клініки середнього розміру. До переліку увійшли 13 показників, серед яких: наявність вебінтерфейсу без

встановлення окремого клієнта, підтримка хмарного розгортання, функція онлайн-запису пацієнтів, ведення електронних медичних карток та анамнезу, управління графіком лікарів, українська локалізація, адаптивний дизайн, а також безкоштовність використання. У порівняльній таблиці застосовано умовні позначення: «+» — функцію реалізовано повністю, «–» — функція відсутня, «+/-» — функцію реалізовано частково або вона доступна лише в окремих тарифних планах. Оцінки складено на підставі офіційної технічної документації виробників та відгуків практикуючих адміністраторів клінік на спеціалізованих форумах. Розроблюваний у рамках цієї роботи вебдодаток представлено в останній колонці таблиці. Відмітною рисою є підтримка операційних протоколів асистента та персоналізованих рекомендацій для пацієнтів — функцій, відсутніх у жодній з розглянутих комерційних систем. Додаток повністю орієнтований на вітчизняний ринок, що відображається у відповідній локалізації та безкоштовній моделі розповсюдження.

Підсумкові результати порівняльного аналізу наведено у таблиці 1.1.

Аналіз виявив, що жодна з комерційних систем не підтримує одночасно всі 13 визначених критеріїв, тоді як розроблюваний додаток забезпечує повну відповідність кожному з них. Особливо суттєвою є перевага у критеріях українськомовного інтерфейсу, безкоштовного доступу та підтримки ролі асистента — функцій, що є критично важливими для вітчизняних стоматологічних закладів. Це підтверджує актуальність та практичну цінність розробки.

Таблиця 1.1

## Порівняльний аналіз вебсервісів для стоматологічних клінік

| Характеристика                  | Dentrix | Open Dental | Curve Dental | Розроблюваний |
|---------------------------------|---------|-------------|--------------|---------------|
| Вебінтерфейс (без клієнта)      | +/-     | -           | +            | +             |
| Хмарне розгортання              | +       | -           | +            | +             |
| Онлайн-запис пацієнтів          | +       | -           | +            | +             |
| Інтерактивна дентальна карта    | +       | +           | +            | +             |
| Електронна медична документація | +       | +           | +            | +             |
| Медичний анамнез пацієнта       | +       | +           | +/-          | +             |
| Операційні протоколи асистента  | -       | -           | -            | +             |
| Персоналізовані рекомендації    | +/-     | -           | +/-          | +             |
| Управління відпустками лікарів  | +       | -           | +            | +             |
| Українська локалізація          | -       | -           | -            | +             |
| Адаптивний дизайн               | +/-     | -           | +            | +             |
| Безкоштовне використання        | -       | +           | -            | +             |
| Гнучкість бізнес-логіки         | -       | +           | -            | +             |

Як видно з порівняльної таблиці, жодне з розглянутих рішень не повністю задовольняє вимогам приватної стоматологічної клініки в Україні. Найбільш гнучкою є Open Dental, проте застарілий клієнт і відсутність вебінтерфейсу роблять її непривабливою. Curve Dental пропонує сучасний UX, але закрита система не дозволяє вносити специфічні правки. Dentrrix — потужний інструмент для великих мереж, проте недоступний для малих клінік. Жодна з платформ не підтримує модуль операційних протоколів для асистентів.

Розробка власного програмного продукту дозволить реалізувати точно ті функціональні можливості, що потрібні цільовій аудиторії, забезпечити повну українську локалізацію, мінімізувати витрати на ліцензування та надати клініці незалежність у подальшому розвитку системи.

### **1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення**

Розробка сучасного вебдодатка передбачає вибір технологічного стеку, який забезпечить високу продуктивність, надійність, зручність розробки та можливість подальшого масштабування системи. Архітектура вебдодатка стоматологічної клініки побудована за клієнт-серверною моделлю з використанням підходу full-stack-фреймворку, де клієнтська та серверна частини інтегровані в межах єдиного застосунку.

Як основна платформа розробки обрано Next.js — повноцінний full-stack-фреймворк на основі бібліотеки React.js версії 19 [12]. Next.js надає інструменти для створення як клієнтських сторінок (React-компоненти), так і серверних API-маршрутів (Route Handlers) у межах єдиного проєкту. Серед ключових переваг: підтримка серверного рендерингу (SSR), File-Based Routing, оптимізація завантаження, вбудована підтримка Server Components React 19. Версія Next.js 16 використовує App Router — найновіший підхід до маршрутизації [4, 16].

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 20   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Для написання коду застосунку обрано мову TypeScript — типізовану надбудову над JavaScript, розроблену компанією Microsoft. TypeScript дозволяє виявляти помилки на етапі компіляції ще до запуску програми, забезпечує авто-доповнення в середовищах розробки та значно спрощує рефакторинг у великих кодових базах. Особливо важливою перевагою є можливість спільної типізації між клієнтською та серверною частинами: типи, визначені у Prisma-схемі, автоматично поширюються на весь застосунок, що унеможливорює невідповідності форматів даних між шарами [5].

Серед альтернативних варіантів технологічного стеку розглядалися: Spring Boot (Java), Django (Python), Laravel (PHP) та Node.js у поєднанні з Express.js. Фреймворки Spring Boot і Django потребують окремого деплою клієнтської та серверної частин, що ускладнює архітектуру і підвищує операційні витрати. Laravel є потужним рішенням, однак потребує окремої екосистеми PHP. Next.js обрано з огляду на можливість розробляти весь стек однією мовою (TypeScript), підтримувати спільну типізацію між шарами та використовувати єдиний інструмент розгортання — платформу Vercel, яка оптимізована саме для Next.js-проектів [28].

Як база даних обрано PostgreSQL — реляційну СУБД з відкритим вихідним кодом, що вирізняється надійністю, дотриманням ACID-принципів та потужним механізмом обмежень цілісності. PostgreSQL підтримує складні запити, індексування, зовнішні ключі та тригери, що є критично важливим для медичної інформаційної системи, де цілісність даних є пріоритетом. Для підключення у продакшн-середовищі використовується хмарна платформа Neon — serverless-рішення на основі PostgreSQL, яке підключається через адаптер @neondatabase/serverless та не потребує виділеного сервера бази даних [6].

Для взаємодії застосунку з PostgreSQL обрано ORM Prisma — сучасний об'єктно-реляційний маппер з декларативним описом схеми у файлі schema.prisma. Prisma автоматично генерує TypeScript-типи на основі схеми

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 21   |

бази даних, що забезпечує повну типізацію результатів запитів і унеможливорює звернення до неіснуючих полів. Автоматична система міграцій дозволяє вносити зміни до структури бази даних безпечно та відтворювано, зберігаючи повну історію змін схеми у вигляді SQL-файлів міграцій [7].

Інтерфейс застосунку побудовано з використанням бібліотеки React 19 та фреймворку Tailwind CSS 4 [8]. Tailwind реалізує підхід utility-first: стилі задаються безпосередньо через утилітарні CSS-класи прямо у JSX-розмітці компонентів, що дозволяє створювати адаптивні інтерфейси з мінімальними витратами часу та без необхідності писати окремі CSS-файли. Вбудований механізм tree-shaking автоматично виключає невикористані стилі з продакшн-збірки, що суттєво зменшує обсяг фінального CSS-файлу. Завдяки адаптивним префіксам (sm:, md:, lg:) інтерфейс коректно відображається на пристроях будь-якого розміру екрана.

Для зберігання фотознімків, що прикріплюються до дентальних записів пацієнтів, використовується хмарний сервіс Cloudinary [9]. Сервіс надає зручний REST API для програмного завантаження зображень із серверної частини застосунку, автоматично оптимізує формат і розмір файлів, а також генерує постійні публічні URL для подальшого відображення зображень у браузері. Використання спеціалізованого CDN для медіафайлів розвантажує базу даних і зменшує витрати на зберігання, оскільки у PostgreSQL зберігаються лише посилання на файли, а не самі бінарні дані.

Авторизацію у системі реалізовано на основі стандарту JSON Web Token (JWT) [17]. Після успішної автентифікації сервер формує підписаний токен, який містить зашифровані ідентифікатор користувача та його роль, і передає його клієнту. Токен зберігається у HTTP-only cookie, що унеможливорює його зчитування через JavaScript і захищає від XSS-атак. Для формування та верифікації токенів використовується бібліотека jose [10], яка підтримує сучасні алгоритми підпису. Паролі користувачів зберігаються виключно у хешованому вигляді за алгоритмом bcrypt [11, 31] з адаптивним фактором

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 22   |

складності, що забезпечує стійкість до атак методом перебору.

До складу допоміжних інструментів розробки входять кілька компонентів. ESLint використовується для статичного аналізу коду: він автоматично виявляє потенційні помилки, порушення стилю та небезпечні конструкції ще на етапі написання коду, що підвищує загальну якість кодової бази [15]. Управління залежностями проєкту здійснюється через npm — стандартний менеджер пакетів екосистеми Node.js, що забезпечує відтворюване встановлення бібліотек через файл package-lock.json. Як середовище розробки обрано Visual Studio Code — легковагий редактор з підтримкою TypeScript, вбудованим відладчиком та широкою екосистемою розширень. Контроль версій вихідного коду ведеться за допомогою Git та платформи GitHub, що дозволяє відстежувати повну історію змін і організовувати командну розробку.

Таблиця 1.2

Системні вимоги до серверної частини

| Параметр           | Мінімальна вимога                                  |
|--------------------|----------------------------------------------------|
| Процесор           | 2,0 ГГц, 2 ядра                                    |
| Оперативна пам'ять | 4 ГБ                                               |
| Накопичувач (SSD)  | 20 ГБ вільного місця                               |
| Операційна система | Ubuntu Server 22.04 LTS / Windows Server 2019/2022 |
| Мережа             | 10 Мбіт/с                                          |
| Node.js            | версія 20+                                         |
| PostgreSQL         | версія 14+                                         |

Вимоги до клієнтської частини є мінімальними: для роботи з вебдодатком достатньо будь-якого сучасного пристрою — персонального комп'ютера, планшета або смартфона — з підтримкою протоколу HTTPS та підключенням до мережі Інтернет. Завдяки адаптивній верстці на основі

Tailwind CSS інтерфейс коректно відображається у діапазоні розмірів екрана від 320 пікселів (смартфон) до 1920 пікселів і більше (широкоформатний монітор), що забезпечує однаково зручну роботу для всіх категорій користувачів системи.

Обраний технологічний стек повністю відповідає як функціональним, так і нефункціональним вимогам [24] до системи. Усі основні компоненти стеку є проєктами з відкритим вихідним кодом, що виключає витрати на ліцензування та забезпечує доступ до активних спільнот розробників і великого масиву документації. Поєднання Next.js, TypeScript, Prisma та PostgreSQL є перевіреним і широко вживаним стеком у промисловій веброботі, що гарантує довгострокову підтримку та сумісність компонент.

### **Висновки до першого розділу**

У першому розділі кваліфікаційної роботи проведено комплексний аналіз предметної області — діяльності сучасних стоматологічних клінік, особливостей організації роботи персоналу та обсягу інформації, що підлягає обробці. Визначено, що автоматизація бізнес-процесів за допомогою вебдодатка є актуальною задачею.

Сформульовано постановку задачі: визначено дев'ять ключових модулів системи для чотирьох ролей (пацієнт, лікар, асистент, адміністратор), окреслено основні етапи розробки програмного продукту.

Виконано порівняльний аналіз Dentrix, Open Dental та Curve Dental. Виявлено, що жодна система не має повної відповідності вимогам українського ринку та не підтримує модуль операційних протоколів для асистентів. Обґрунтовано доцільність розробки власного рішення.

Обрано технологічний стек: Next.js 16, React 19, TypeScript, PostgreSQL (Neon), Prisma, Tailwind CSS 4, Cloudinary, jose, bcryptjs. Визначено вимоги до апаратного та програмного забезпечення. Обраний стек є сучасним, продуктивним та економічно ефективним.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 24   |

Таким чином, у першому розділі сформовано теоретичну й технологічну базу для подальшого проектування та реалізації вебдодатка стоматологічної клініки.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 25   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКА СТОМАТОЛОГІЧНОЇ КЛІНІКИ

### 2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Будь-яка інформаційна система розробляється для задоволення потреб конкретних груп користувачів. Для вебдодатка стоматологічної клініки виокремлено три основні ролі (актори), які взаємодіють із системою: пацієнт, лікар та адміністратор. Кожна з цих ролей має власні цілі, очікування і доступ до різних функцій застосунку, тому розмежування прав доступу та зон відповідальності було ключовим аспектом проєктування.

Пацієнт — кінцевий споживач послуг клініки. Його основні задачі: створити обліковий запис у системі, переглянути доступні часові слоти, обрати лікаря та забронювати прийом, переглянути історію власних візитів, ознайомитися з рекомендаціями та призначеннями, що залишив лікар, а також звернутися до клініки із додатковим запитом, якщо стандартний механізм запису не задовольнив його потреби.

Лікар — медичний працівник стоматологічної клініки. Його роль у системі полягає у веденні професійної діяльності: перегляд власного розкладу прийомів, опрацювання карток пацієнтів, занесення інформації про проведене лікування у дентальну карту, формування персоналізованих рекомендацій, відмітка пацієнтів, які не з'явилися на прийом і потребують повторного дзвінка, ведення списку пацієнтів, з якими лікар взаємодіє.

Адміністратор — особа, відповідальна за загальне функціонування клініки та правильність організаційних процесів. У межах системи він керує переліком лікарів (додає новий персонал, редагує дані, припиняє роботу), формує і модифікує розклад, обробляє запити, які надходять від пацієнтів, реагує на нестандартні ситуації та переглядає узагальнену статистику діяльності закладу.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 26   |

Для формального опису сценаріїв взаємодії користувачів із системою побудовано діаграму прецедентів використання (use case-діаграму) [19, 25], яка відображає основні функції, доступні кожному з акторів. Система підтримує такі прецеденти: реєстрацію та авторизацію користувача; перегляд переліку лікарів та їхніх спеціалізацій; перегляд доступних слотів у графіку лікаря; бронювання прийому пацієнтом (онлайн-запис); відправку запиту до клініки через форму звернення; перегляд власних майбутніх та минулих прийомів; перегляд персональних рекомендацій від лікаря; перегляд інтерактивної дентальної карти пацієнта. З боку лікаря реалізовано: перегляд переліку власних пацієнтів та пацієнтів, що потребують повторного контакту; заповнення дентального запису після прийому; додавання рекомендацій конкретному пацієнту. Адміністратор має доступ до: управління переліком лікарів (CRUD-операції); управління розкладом (створення, зміна, видалення слотів); запису пацієнта на прийом, у тому числі гостьового запису без реєстрації; обробки запитів від пацієнтів; перегляду статистики клініки.

Детальний опис ключових прецедентів використання наведено у таблицях 2.1 — 2.6.

Таблиця 2.1

Опис прецеденту «Авторизація користувача»

|                             |                                                                                                                                                                       |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Авторизація користувача                                                                                                                                               |
| Основні актори              | Пацієнт, Лікар, Адміністратор                                                                                                                                         |
| Короткий опис               | Користувач вводить електронну пошту та пароль; система перевіряє облікові дані, видає JWT-токен і направляє користувача до робочого простору відповідно до його ролі. |

Вхід до системи є передумовою для будь-якої іншої дії користувача, тому цей прецедент описано першим (табл. 2.1). Форма входу єдина для всіх категорій користувачів: людина вводить електронну пошту та пароль, сервер

знаходить відповідний обліковий запис і зіставляє введений пароль зі збереженим хешем. Після успішної перевірки формується JWT-токен, у якому зашифровано ідентифікатор та роль. Саме роль із токена визначає, куди буде перенаправлено користувача — до кабінету пацієнта, лікаря, асистента чи адміністратора.

Таблиця 2.2

Опис прецеденту «Онлайн-запис на прийом»

|                             |                                                                                                                                                            |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Онлайн-запис на прийом                                                                                                                                     |
| Основні актори              | Пацієнт                                                                                                                                                    |
| Короткий опис               | Авторизований пацієнт обирає лікаря, переглядає доступні часові слоти на найближчі дати, обирає вільний слот, вказує тип прийому і підтверджує бронювання. |

Онлайн-запис на прийом (табл. 2.2) доступний лише авторизованому пацієнтові. Спочатку пацієнт обирає лікаря, після чого на екрані з'являється тижнева сітка годинних слотів, де вільні позиції виділено кольором. Бронювання виконується однією транзакцією: слот позначається як зайнятий і водночас створюється запис про прийом. Завдяки цьому два пацієнти не можуть зайняти той самий час, навіть якщо натиснуть кнопку одночасно. Окремо перевіряється, чи не припадає обрана дата на відпустку лікаря.

Таблиця 2.3

Опис прецеденту «Відправка запиту до клініки»

|                             |                                                                                                                                                                             |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Відправка запиту до клініки                                                                                                                                                 |
| Основні актори              | Пацієнт                                                                                                                                                                     |
| Короткий опис               | Пацієнт заповнює форму довільного звернення (опис проблеми, побажання щодо часу прийому), запит зберігається у системі і потрапляє до адміністратора для подальшої обробки. |

Не завжди пацієнт готовий одразу обрати конкретний час — інколи потрібна попередня консультація або узгодження. Для таких випадків передбачено прецедент відправки запиту до клініки, наведений у табл. 2.3. Звернення зберігається зі статусом NEW і потрапляє до черги адміністратора, який або записує пацієнта на конкретний слот, або просто переводить запит у статус PROCESSED.

Таблиця 2.4

Опис прецеденту «Заповнення дентального запису»

|                             |                                                                                                                                                                                            |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Заповнення дентального запису                                                                                                                                                              |
| Основні актори              | Лікар                                                                                                                                                                                      |
| Короткий опис               | Лікар відкриває картку пацієнта, обирає на дентальній карті номер зуба, що підлягає лікуванню, вносить інформацію про проблему, обране лікування, додає коментар та за потреби фотознімок. |

Заповнення дентального запису — основний робочий сценарій лікаря (табл. 2.4). Лікар обирає пацієнта зі списку своїх прийомів і відкриває інтерактивну дентальну карту, де кожен зуб позначено номером за міжнародною системою FDI. Обравши потрібний зуб, лікар вносить опис проблеми та проведеного лікування, за потреби додає коментар і фотознімок. Саме зображення завантажується до хмарного сховища, а в базі зберігається лише посилання на нього. Кожен запис прив'язується одночасно до пацієнта і до лікаря, тому завжди відомо, хто саме вніс зміни до медичної документації.

Таблиця 2.5

Опис прецеденту «Управління розкладом»

|                             |                      |
|-----------------------------|----------------------|
| Назва варіанту використання | Управління розкладом |
|-----------------------------|----------------------|

|                |                                                                                                                                                                              |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Основні актори | Адміністратор                                                                                                                                                                |
| Короткий опис  | Адміністратор створює нові часові слоти для конкретних лікарів, редагує дату та час існуючих слотів, видаляє слоти, що більше не актуальні, та контролює зайнятість графіка. |

Управління розкладом належить до зони відповідальності адміністратора (табл. 2.5). Він формує часові слоти для кожного лікаря на обраний період, редагує їх і видаляє ті, що втратили актуальність. Видалення діє тільки на вільні слоти — позиції з уже записаними пацієнтами залишаються недоторканими. Оскільки лікар може створювати записи лише в межах днів, для яких сформовано розклад, цей прецедент фактично є передумовою для решти сценаріїв роботи з прийомами.

Таблиця 2.6

## Опис прецеденту «Гостьовий запис пацієнта»

|                             |                                                                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Гостьовий запис пацієнта                                                                                                                                                         |
| Основні актори              | Адміністратор                                                                                                                                                                    |
| Короткий опис               | Адміністратор записує на прийом пацієнта, який не має облікового запису у системі (наприклад, дзвінок по телефону), вказуючи його ім'я та номер телефону як гостьового пацієнта. |

Частина пацієнтів звертається до клініки телефоном і не має облікового запису. Такий випадок описує останній із розглянутих прецедентів — гостьовий запис (табл. 2.6). Адміністратор фіксує прийом, зазначаючи ім'я та номер телефону особи, і запис зберігається без прив'язки до зареєстрованого користувача. Якщо згодом пацієнт зареєструється сам, вказавши той самий телефон або ім'я, попередні візити автоматично прив'яжуться до нового облікового запису, і історія лікування не переривається.

Об'єктно-орієнтований підхід дозволяє виокремити основні сутності системи та пов'язати їх відповідно до бізнес-логіки. Центральною сутністю є User — узагальнений користувач системи, який має облікові дані, ім'я, контактну інформацію та роль (PATIENT або ADMIN). Сутність Doctor представлена окремою моделлю, оскільки лікар має додаткові поля (спеціалізація, графік роботи, відносини з пацієнтами через прийоми та дентальні записи) і власну схему автентифікації.

Сутність ScheduleSlot описує одиничний часовий слот у розкладі лікаря: дату та час, статус зайнятості. Сутність Appointment представляє факт запису пацієнта на прийом і пов'язує між собою пацієнта, лікаря та конкретний слот. Особливістю реалізації є можливість зберігати у Appointment гостьові поля (guestName, guestPhone), що дозволяє створювати запис без обов'язкової реєстрації пацієнта у системі.

Сутність DentalRecord представляє електронний запис у дентальній карті пацієнта: номер зуба, проблема, проведене лікування, коментарі, посилання на фото. Сутність Recommendation зберігає текстові рекомендації лікаря пацієнту (наприклад, режим харчування після видалення зуба, рекомендоване полоскання, графік контрольних оглядів). Сутність PatientRequest описує довільне звернення пацієнта до клініки і має статус NEW або PROCESSED залежно від того, чи опрацював адміністратор запит.

Узагальнена об'єктно-орієнтована структура системи відображена на діаграмі класів (рис. 2.1). На діаграмі показано основні сутності, їхні поля та відносини між ними у вигляді асоціацій типу «один до багатьох» (наприклад, один лікар має багато слотів розкладу, один пацієнт має багато записів у дентальній карті).

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 31   |

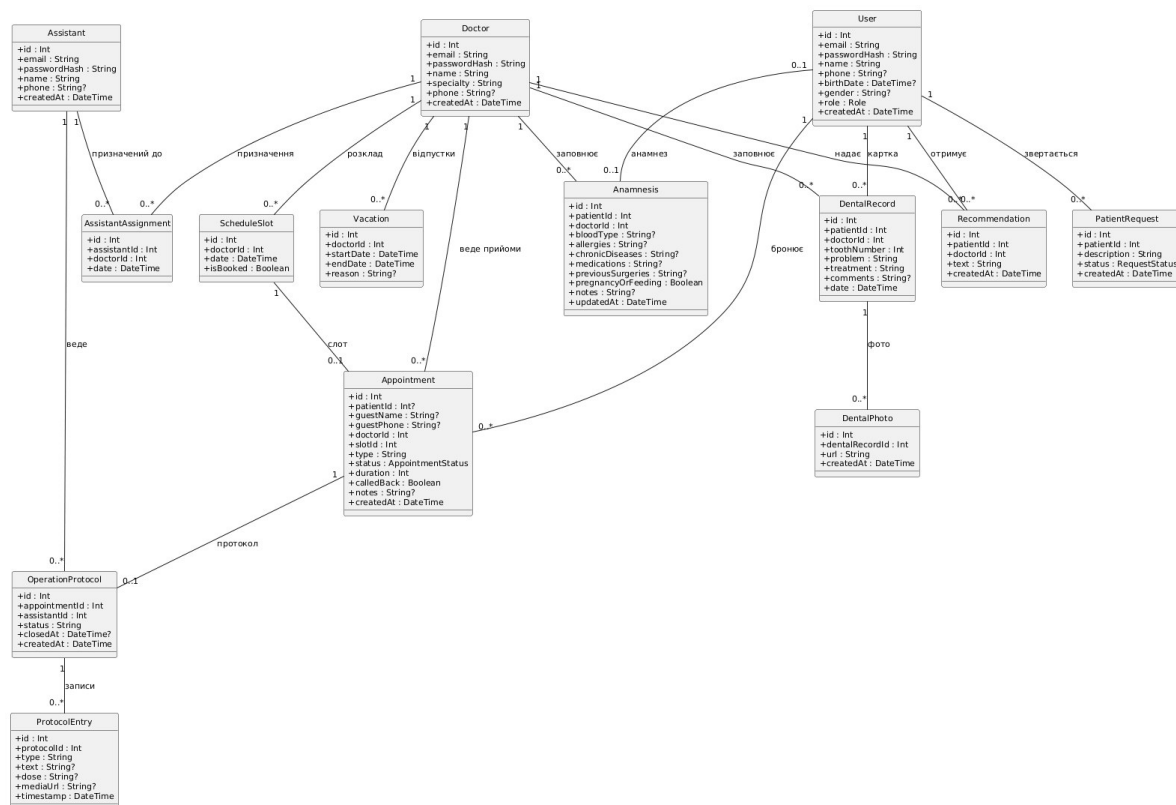


Рис. 2.1. Діаграма класів вебдодатка стоматологічної клініки

Об'єктно-орієнтований підхід забезпечує низку переваг при розробці вебдодатка: чітке розділення відповідальностей між сутностями, спрощення подальшого розширення системи новими модулями, можливість повторного використання типів між клієнтом і сервером (через TypeScript та Prisma Client), інкапсуляцію логіки роботи з даними у сервісному шарі та полегшення тестування. Дотримання принципів SOLID [21] при проєктуванні класів і модулів робить кодову базу більш зрозумілою для майбутніх розробників і знижує ризик появи помилок при внесенні змін.

## 2.2. Розробка бази даних системи

Для зберігання даних у вебдодатку стоматологічної клініки спроектовано реляційну базу даних під керуванням системи PostgreSQL. Реляційний підхід обрано з огляду на чітко визначений набір сутностей, наявність множини зв'язків між ними, потребу у строгому контролі цілісності та необхідність дотримання ACID-властивостей при роботі з медичною

інформацією. NoSQL-рішення, які надають більшу гнучкість схеми, у даному випадку не є оптимальним вибором, оскільки структура даних стоматологічної клініки добре формалізується і змінюється рідко.

Опис структури бази даних виконано у декларативному форматі за допомогою файла `schema.prisma` [26], що належить до ORM Prisma. Це дозволяє автоматично генерувати міграції бази даних, типобезпечний клієнт для роботи з нею у TypeScript та документацію зі схемою. Файл `schema.prisma` [26] виступає єдиним джерелом істини щодо структури даних: будь-яка зміна полів моделі автоматично відображається в типах TypeScript, що використовуються у клієнтських компонентах і API-маршрутах.

База даних містить сім основних таблиць (моделей): `User`, `Doctor`, `ScheduleSlot`, `Appointment`, `PatientRequest`, `DentalRecord`, `Recommendation`. Структура кожної з них описана в таблицях 2.7 — 2.13.

Центральною моделлю схеми є `User` — саме навколо неї побудовано більшість зв'язків. Лікарі, записи на прийом, медичні картки й рекомендації — усе це прив'язане до конкретного користувача. Логіка роботи зі слотами дещо відрізняється: таблиця `ScheduleSlot` зберігає доступні часові проміжки у розкладі лікаря, а під час оформлення запису відповідний слот отримує прив'язку до `Appointment`. Завдяки цьому подвійне бронювання виключено вже на рівні бази даних, без додаткових перевірок у коді. Щодо технічних деталей схеми — усі первинні ключі є автоінкрементними `Int`-полями. Поля, що можуть бути порожніми, позначено знаком питання (наприклад, `guestName String?`). Часові мітки `createdAt` заповнюються автоматично через `@default(now())`, тому передавати їх з боку застосунку не потрібно. Перелічувані типи `Role` і `AppointmentStatus` фіксують допустимий набір значень на рівні схеми і при генерації клієнта Prisma транслюються у відповідні типи TypeScript.

Таблиця 2.7

Опис таблиці User

| Поле         | Тип             | Призначення                     |
|--------------|-----------------|---------------------------------|
| id           | Int (PK, auto)  | Унікальний ідентифікатор        |
| email        | String (unique) | Електронна пошта (логін)        |
| passwordHash | String          | Хеш пароля (bcrypt)             |
| name         | String          | Повне ім'я користувача          |
| phone        | String?         | Контактний номер телефону       |
| birthDate    | DateTime?       | Дата народження                 |
| gender       | String?         | Стать                           |
| role         | Role enum       | Роль (PATIENT / DOCTOR / ADMIN) |
| createdAt    | DateTime        | Дата створення запису           |

Поля моделі User перелічено в табл. 2.7. Електронна пошта оголошена унікальною і водночас виконує роль логіна, тому двох користувачів з однаковою адресою в системі бути не може. Пароль у відкритому вигляді не зберігається — у базі є лише його хеш, обчислений bcrypt. Телефон, дата народження та стать необов'язкові: пацієнт може зареєструватися з мінімальним набором даних і доповнити профіль пізніше.

Таблиця 2.8

Опис таблиці Doctor

| Поле         | Тип             | Призначення              |
|--------------|-----------------|--------------------------|
| id           | Int (PK, auto)  | Унікальний ідентифікатор |
| email        | String (unique) | Електронна пошта лікаря  |
| passwordHash | String          | Хеш пароля               |
| name         | String          | Повне ім'я лікаря        |
| specialty    | String          | Спеціалізація лікаря     |
| phone        | String?         | Контактний номер         |
| createdAt    | DateTime        | Дата створення запису    |

Лікар винесений в окрему таблицю, а не реалізований як різновид користувача. Причина видно з табл. 2.8: у лікаря є власні атрибути, передусім

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 34   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

спеціалізація, і власні зв'язки — розклад, дентальні записи, рекомендації, відпустки, — які для пацієнта не мають сенсу. Логін і збереження пароля влаштовані так само, як у моделі User.

Таблиця 2.9

Опис таблиці ScheduleSlot

| Поле     | Тип               | Призначення                    |
|----------|-------------------|--------------------------------|
| id       | Int (PK, auto)    | Унікальний ідентифікатор слота |
| doctorId | Int (FK → Doctor) | Посилання на лікаря            |
| date     | DateTime          | Дата та час слота              |
| isBooked | Boolean           | Чи заброньований слот          |

Найдрібнішою одиницею планування є слот, що відповідає одній годині робочого часу конкретного лікаря (табл. 2.9). Зовнішній ключ doctorId пов'язує слот із лікарем, а логічне поле isBooked показує його поточний стан. Перевірка та зміна цього поля в межах однієї транзакції і забезпечує систему від подвійного бронювання.

Таблиця 2.10

Опис таблиці Appointment

| Поле       | Тип                             | Призначення                                 |
|------------|---------------------------------|---------------------------------------------|
| id         | Int (PK, auto)                  | Унікальний ідентифікатор прийому            |
| patientId  | Int? (FK → User)                | Посилання на пацієнта (якщо зареєстрований) |
| guestName  | String?                         | Ім'я гостьового пацієнта                    |
| guestPhone | String?                         | Телефон гостьового пацієнта                 |
| doctorId   | Int (FK → Doctor)               | Посилання на лікаря                         |
| slotId     | Int (FK → ScheduleSlot, unique) | Слот, до якого прив'язано прийом            |

|            |                        |                                                |
|------------|------------------------|------------------------------------------------|
| type       | String                 | Тип прийому (огляд, лікування тощо)            |
| status     | AppointmentStatus enum | Статус (PENDING/CONFIRMED/COMPLETED/CANCELLED) |
| duration   | Int                    | Тривалість прийому у слотах                    |
| calledBack | Boolean                | Прапорець потреби зворотного дзвінка           |
| notes      | String?                | Додаткові примітки                             |
| createdAt  | DateTime               | Дата створення запису                          |

Appointment — центральна модель усього процесу обслуговування, бо поєднує пацієнта, лікаря і часовий слот (табл. 2.10). Поле patientId необов'язкове: коли прийом створює адміністратор для незареєстрованої особи, замість нього заповнюються guestName і guestPhone. Зв'язок зі слотом оголошено унікальним, тож до одного слота неможливо прив'язати більше ніж один прийом. Поле duration потрібне для тривалих втручань, що займають кілька годин поспіль. Статус відображає стан прийому в межах життєвого циклу, а прапорець calledBack лікар використовує, щоб позначити пацієнтів, яких уже запросили до кабінету.

Таблиця 2.11

## Опис таблиці PatientRequest

| Поле        | Тип                | Призначення                     |
|-------------|--------------------|---------------------------------|
| id          | Int (PK, auto)     | Унікальний ідентифікатор запиту |
| patientId   | Int (FK → User)    | Посилання на пацієнта           |
| description | String             | Текст звернення                 |
| status      | RequestStatus enum | Статус (NEW / PROCESSED)        |
| createdAt   | DateTime           | Дата створення                  |

Структура звернень пацієнтів проста (табл. 2.11): посилання на пацієнта, текст опису і статус обробки. Записи впорядковано за датою створення, тому

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 36   |

адміністратор бачить чергу в порядку надходження. Перехід статусу з NEW у PROCESSED фіксує факт опрацювання.

Таблиця 2.12

Опис таблиці DentalRecord

| Поле        | Тип               | Призначення                            |
|-------------|-------------------|----------------------------------------|
| id          | Int (PK, auto)    | Унікальний ідентифікатор запису        |
| patientId   | Int (FK → User)   | Посилання на пацієнта                  |
| doctorId    | Int (FK → Doctor) | Посилання на лікаря                    |
| toothNumber | Int               | Номер зуба за міжнародною системою FDI |
| problem     | String            | Опис стоматологічної проблеми          |
| treatment   | String            | Опис проведеного лікування             |
| comments    | String?           | Додаткові коментарі лікаря             |
| photoUrl    | String?           | Посилання на фотознімок                |
| date        | DateTime          | Дата виконання запису                  |

Електронний запис у дентальній карті описує модель DentalRecord (табл. 2.12). Номер зуба зберігається за міжнародною системою FDI — це дозволяє однозначно зіставити запис із конкретною позицією на інтерактивній карті. Опис проблеми та проведеного лікування обов'язкові, коментар і фотознімок заповнюються за потреби. Два зовнішні ключі, на пацієнта і на лікаря, роблять кожен запис персоніфікованим.

Таблиця 2.13

Опис таблиці Recommendation

| Поле      | Тип               | Призначення              |
|-----------|-------------------|--------------------------|
| id        | Int (PK, auto)    | Унікальний ідентифікатор |
| patientId | Int (FK → User)   | Посилання на пацієнта    |
| doctorId  | Int (FK → Doctor) | Посилання на лікаря      |
| text      | String            | Текст рекомендації       |
| createdAt | DateTime          | Дата створення           |

Останньою в переліку йде модель Recommendation (табл. 2.13). Вона зберігає текстові призначення, які лікар формує для пацієнта після прийому, разом із датою створення. У кабінеті пацієнта такі записи показуються хронологічним списком із зазначенням лікаря, що їх виписав.

Зв'язки між таблицями реалізовані за допомогою зовнішніх ключів та обмежень цілісності, що гарантує узгодженість даних при будь-яких операціях. Зокрема, видалення лікаря викликає каскадне видалення пов'язаних з ним слотів розкладу (onDelete: Cascade), що відображає реальну логіку: якщо лікар припинив роботу в клініці, його розклад також стає неактуальним. Видалення пацієнта, на відміну від цього, не призводить до видалення дентальних записів, оскільки медична документація повинна зберігатися в архіві навіть після того, як пацієнт перестав користуватися послугами клініки.

Особливу увагу під час проєктування бази даних приділено забезпеченню унікальності ключових полів. Поле email у таблицях User та Doctor має обмеження UNIQUE, що унеможливорює створення двох користувачів з однаковим логіном. Поле slotId у таблиці Appointment також є унікальним, що гарантує неможливість одночасного бронювання одного й того ж часового слота двома різними прийомами — критично важлива бізнес-вимога, що запобігає накладанню записів у графіку лікаря.

ER-діаграма бази даних представлена на рис. 2.2. Вона ілюструє основні таблиці, ключові поля та зв'язки між ними. Видно, що центральне місце в схемі займає таблиця Appointment, яка пов'язує між собою пацієнта (User), лікаря (Doctor) та слот розкладу (ScheduleSlot). Допоміжні таблиці DentalRecord, Recommendation та PatientRequest безпосередньо посиляються на User та Doctor, описуючи різні аспекти взаємодії пацієнта з клінікою.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 38   |



введений пароль із хешем, що зберігається у полі passwordHash. У разі успіху сервер генерує підписаний JWT-токен (бібліотека jose), у який вкладає id користувача та роль, і відправляє цей токен клієнту через cookie з прапорцем HttpOnly. Клієнт перенаправляється до робочого простору відповідно до своєї ролі: пацієнт — на /patient/dashboard, лікар — на /doctor/dashboard, адміністратор — на /admin/dashboard.

Алгоритм онлайн-запису пацієнта реалізований таким чином. Авторизований пацієнт відкриває сторінку «Записатися на прийом» (/patient/book), де представлений перелік усіх лікарів клініки. Після обрання лікаря клієнтська частина запитує доступні слоти у /api/doctor/book/slots з ідентифікатором лікаря. Сервер повертає список вільних слотів (де isBooked = false) у вигляді масиву об'єктів зі значенням дати, часу та ідентифікатора. Пацієнт обирає зручний для нього слот, вказує тип прийому (наприклад, «огляд» або «лікування») і підтверджує бронювання. Клієнт надсилає POST-запит на /api/patient/book, сервер у межах однієї транзакції створює новий запис у таблиці Appointment, оновлює прапорець isBooked = true у відповідному ScheduleSlot та повертає клієнту інформацію про успішне бронювання. Якщо протягом виконання транзакції інший користувач встиг забронювати цей же слот, сервер повертає помилку, і клієнтська сторона повідомляє пацієнта про необхідність обрати інший час.

Алгоритм заповнення дентального запису побудований аналогічним чином. Лікар відкриває сторінку «Дентальна карта» (/doctor/dental-chart), вибирає пацієнта зі списку та натискає на конкретний зуб на інтерактивній схемі. На екрані з'являється модальне вікно з полями для заповнення: проблема, обране лікування, коментарі, можливість додати фото. Після підтвердження клієнт відправляє POST-запит на /api/doctor/dental-record. Серверний обробник перевіряє, що поточний користувач має роль DOCTOR, валідує введені дані і створює новий запис у таблиці DentalRecord із прив'язкою до пацієнта, лікаря та номера зуба. У разі додавання фотознімка

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 40   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

файл зберігається у відповідному сховищі, а у поле `photoUrl` записується посилання.

Алгоритм формування рекомендацій лікарем запускається з відповідної сторінки робочого простору лікаря. Лікар обирає пацієнта, вводить текст рекомендації (наприклад, «прополіскувати ротову порожнину розчином хлоргексидину 3 рази на день протягом 5 днів») і натискає кнопку «Зберегти». Клієнт надсилає POST-запит на `/api/doctor/recommendation`, сервер створює новий запис у таблиці `Recommendation`. Пацієнт надалі бачить ці рекомендації у власному кабінеті на сторінці `/patient/prescriptions`, де реалізовано хронологічний список усіх отриманих призначень.

Алгоритм гостьового запису, доступний адміністратору, дозволяє зафіксувати прийом для пацієнта, який ще не зареєстрований у системі (наприклад, телефонний дзвінок). Адміністратор переходить на сторінку `/admin/schedule`, обирає вільний слот, заповнює поля «Ім'я гостя» та «Телефон» і підтверджує запис. Клієнт надсилає POST-запит на `/api/admin/book`. Сервер створює запис у таблиці `Appointment` з порожнім `patientId` та заповненими полями `guestName` і `guestPhone`, оновлює статус слота і повертає підтвердження. Така архітектура дозволяє адміністратору працювати з системою як у режимі повної реєстрації, так і у спрощеному режимі, що є важливим для невеликих клінік, де частина пацієнтів звертаються по телефону.

Алгоритм обробки запиту пацієнта також орієнтований на роботу адміністратора. Запит, відправлений пацієнтом через форму на `/patient/request`, потрапляє у таблицю `PatientRequest` зі статусом `NEW`. Адміністратор переглядає перелік нових запитів на сторінці `/admin/requests`. Після зв'язку з пацієнтом адміністратор натискає «Опрацьовано», що змінює статус запиту на `PROCESSED` через PATCH-запит до `/api/admin/requests/[id]`. Це дозволяє підтримувати чергу необроблених звернень і не пропустити жодне з них.

Кожен з описаних алгоритмів супроводжується перевітками на стороні

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 41   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

сервера: підтвердження автентифікації, валідація вхідних даних, контроль ролі користувача та цілісності бази. Помилки опрацьовуються централізовано: серверний обробник повертає клієнту HTTP-статус та повідомлення про помилку у форматі JSON, а клієнтський компонент відображає його через зручне модальне вікно або повідомлення-нотифікацію.

Узагальнена діаграма активності, що ілюструє послідовність кроків при типовій взаємодії пацієнта з системою (від відкриття головної сторінки до перегляду рекомендацій), наведена на рис. 2.3.

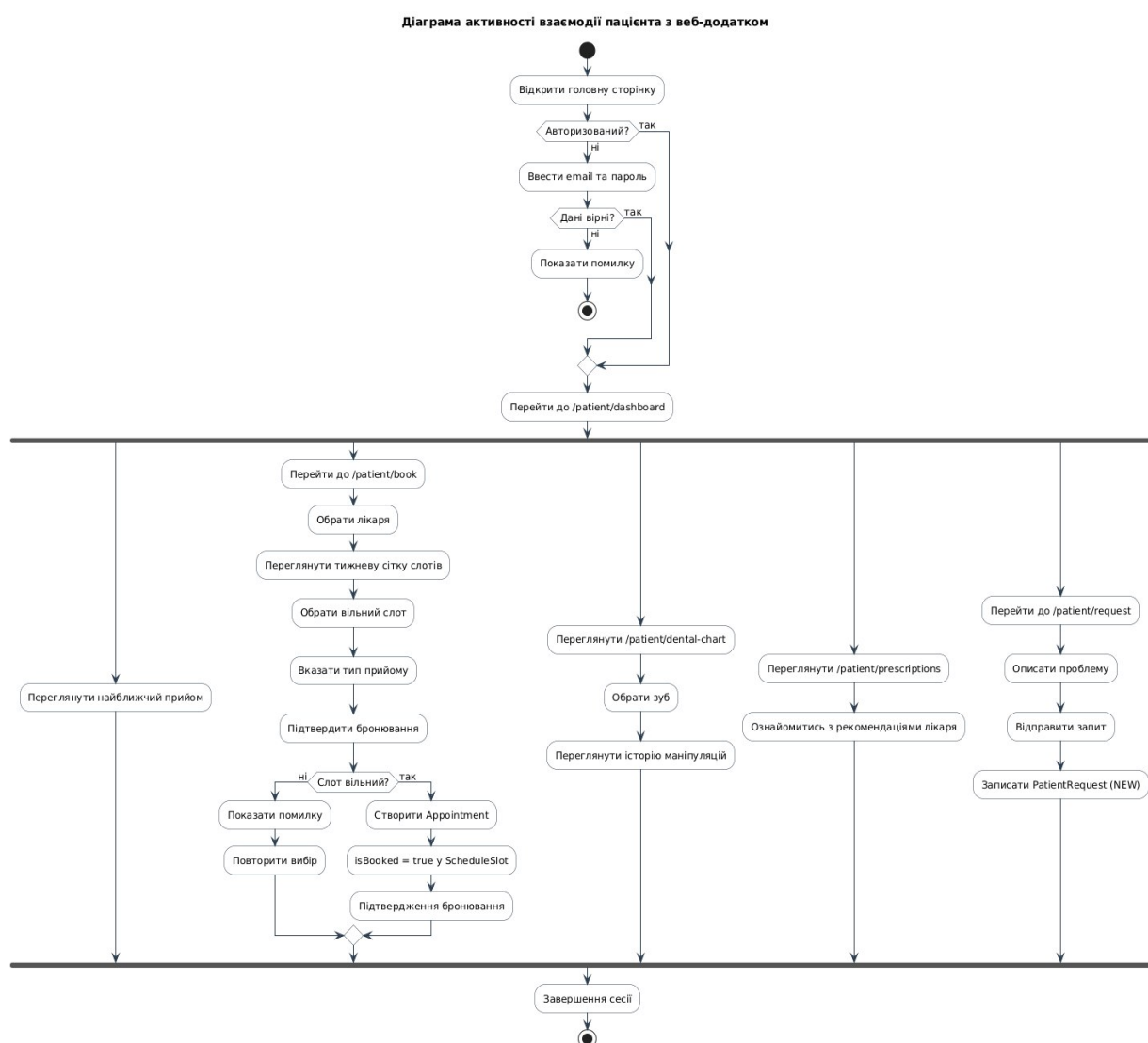


Рис. 2.3. Діаграма активності взаємодії пацієнта з вебдодатком

## 2.4. Реалізація функціоналу вебдодатка стоматологічної клініки

Реалізація функціоналу вебдодатка стоматологічної клініки виконана з дотриманням архітектурного шаблону Model-View-Controller [20, 23] у його сучасній адаптації для full-stack-фреймворків. Роль моделей виконують Prisma-сутності та сервісний шар роботи з базою даних; представлення реалізовано у вигляді React-компонентів, що рендеряться на сервері та доповнюються інтерактивністю на клієнті; роль контролерів виконують серверні API-маршрути REST [22], описані як Route Handlers у каталозі `src/app/api`.

Структура проєкту відповідає стандартам Next.js App Router: каталог `src/app` містить як сторінки інтерфейсу (наприклад, `src/app/patient/dashboard/page.tsx`), так і API-маршрути (`src/app/api/...`). Сторінки організовані за функціональними роздільниками — окремі підкаталоги для робочих просторів пацієнта (`/patient`), лікаря (`/doctor`) та адміністратора (`/admin`). Кожен робочий простір має власний `layout`-компонент, який відповідає за загальну структуру сторінок (бічне меню, верхня панель, перевірка авторизації).

Серверна логіка розділена на тематичні групи маршрутів. Група `/api/auth` містить ендпоїнти реєстрації, входу, виходу та отримання даних поточного користувача; група `/api/patient` — функції, доступні пацієнту (бронювання, перегляд прийомів, дентальної карти, рекомендацій, відправка запитів); група `/api/doctor` — функції лікаря (управління дентальними записами, рекомендаціями, переглядом пацієнтів); група `/api/admin` — функції адміністратора (управління лікарями, графіком, обробка запитів, статистика). Такий поділ полегшує навігацію за кодом та забезпечує чітке розмежування зон відповідальності.

Користувацький інтерфейс побудований з використанням бібліотеки React 19 та Tailwind CSS 4. Компоненти інтерфейсу розбито на дрібні самостійні елементи (кнопки, форми, картки, модальні вікна) [27], що

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 43   |

забезпечує повторне використання коду та полегшує підтримку. Особливу увагу приділено інтерактивній дентальній карті — ключовому компоненту медичного функціоналу. Карта реалізована у вигляді SVG-схеми з 32 зубами, кожен із яких є інтерактивним і відображає поточний статус (здоровий, лікувався, видалений) кольорами та іконками.

Робочий простір пацієнта (/patient/dashboard) надає узагальнене зведення: найближчий запис на прийом, перелік минулих візитів, кількість непрочитаних рекомендацій. Сторінка /patient/book реалізує процес онлайн-запису: вибір лікаря у вигляді карток зі спеціалізаціями, перегляд календаря з вільними слотами, бронювання обраного слота. Сторінка /patient/appointments відображає історію всіх прийомів пацієнта з можливістю фільтрації за статусом. Сторінка /patient/dental-chart надає пацієнту перегляд власної дентальної карти з історією маніпуляцій по кожному зубі. Сторінка /patient/prescriptions містить список усіх рекомендацій, наданих лікарями.

Робочий простір лікаря (/doctor/dashboard) орієнтований на повсякденні задачі стоматолога. На головній сторінці відображається розклад прийомів на поточний день, перелік пацієнтів, які потребують повторного дзвінка (calledBack = true), та швидкі переходи до основних функцій. Сторінка /doctor/dental-chart дозволяє обрати пацієнта зі списку та редагувати його дентальну карту у режимі реального часу. Сторінка /doctor/recommendations надає інтерфейс для створення нових призначень, а сторінка /doctor/book — інструмент для самостійного планування додаткових прийомів.

Робочий простір адміністратора (/admin/dashboard) забезпечує повне керування діяльністю клініки. На головній сторінці виведено основні показники: кількість прийомів за поточний місяць, число активних лікарів, динаміка записів за тиждень, кількість необроблених запитів. Сторінка /admin/doctors реалізує управління штатом лікарів через стандартні CRUD-операції (Create, Read, Update, Delete). Сторінка /admin/schedule дозволяє керувати загальним розкладом клініки: створювати нові слоти для лікарів,

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 44   |

редагувати існуючі, видаляти неактуальні. Сторінка `/admin/requests` відображає чергу запитів пацієнтів, відсортовану за датою надходження, з можливістю позначити запит як опрацьований.

Усі форми у системі використовують клієнтську валідацію з негайним відображенням помилок (порожні поля, невірний формат email, надто короткий пароль). Серверна частина дублює всі перевірки, оскільки клієнтську валідацію легко обійти, що неприйнятно для системи з медичними даними. Помилки серверної частини передаються клієнту у структурованому форматі JSON, де поле `message` містить локалізований текст для відображення користувачу.

Безпека вебдодатка забезпечується кількома механізмами. По-перше, всі паролі зберігаються у хешованому вигляді (`bcrypt` [31] з фактором складності 10). По-друге, JWT-токени передаються у `cookie` з прапорцями `HttpOnly` та `Secure` [29], що захищає їх від доступу через JavaScript [18] і від відправки у незахищеному з'єднанні. По-третє, кожний серверний обробник API перевіряє роль поточного користувача перед виконанням операції — пацієнт не може звернутися до маршрутів адміністратора, навіть якщо знає їхні URL. По-четверте, для роботи з базою даних використовується ORM `Prisma`, що автоматично екранує всі параметри запитів та запобігає SQL-ін'єкціям.

Результатом виконаної роботи є цілісний прототип вебдодатка стоматологічної клініки, який реалізує повний цикл взаємодії пацієнта, лікаря та адміністратора з інформаційною системою. Прототип повністю функціональний, відповідає сформульованим у першому розділі вимогам та готовий до подальшого тестування і розгортання у реальному медичному закладі.

## Висновки до другого розділу

У другому розділі кваліфікаційної роботи виконано повноцінне проєктування та реалізацію архітектури вебдодатка стоматологічної клініки. Проведена робота охоплює всі ключові аспекти створення сучасного

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 45   |

програмного продукту: визначення варіантів використання, проєктування об'єктно-орієнтованої структури, розробка бази даних, опис алгоритмів роботи системи та реалізація функціоналу.

Сформульовано перелік основних акторів системи (пацієнт, лікар, адміністратор) і детально описано прецеденти використання, що відповідають їхнім задачам. Побудовано діаграму класів, на якій відображено сім ключових сутностей системи та зв'язки між ними. Спроектовано схему реляційної бази даних на основі PostgreSQL, описано структуру кожної таблиці та обмеження цілісності, що забезпечують узгодженість даних.

Реалізовано серверну частину застосунку у вигляді API-маршрутів Next.js, згрупованих за функціональними областями. Розроблено клієнтську частину з адаптивним інтерфейсом для трьох робочих просторів — пацієнта, лікаря та адміністратора. Особливу увагу приділено реалізації інтерактивної дентальної карти, формуванню персоналізованих рекомендацій, обробці гостей записів та запитів від пацієнтів.

Описано ключові алгоритми системи: авторизація користувача на основі JWT-токенів, онлайн-запис на прийом із транзакційним оновленням розкладу, заповнення дентального запису, формування рекомендацій, обробка запитів і гостьовий запис пацієнтів. Кожен алгоритм спроектовано з урахуванням вимог надійності, безпеки та зручності користувача.

Безпека системи забезпечена комплексом заходів: хешування паролів за алгоритмом bcrypt, передача JWT-токенів через захищені HTTP-only cookie, перевірка ролі користувача на кожному API-маршруті, використання ORM Prisma для уникнення SQL-ін'єкцій. Архітектура системи є модульною та готовою до розширення додатковими компонентами в майбутньому.

Результатом виконання другого розділу є повноцінний програмний прототип вебдодатка стоматологічної клініки, що повністю відповідає функціональним вимогам, сформульованим у першому розділі. Прототип

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 46   |

може бути використаний як основа для подальшого впровадження у реальному медичному закладі та розширений додатковими модулями (наприклад, інтеграцією з месенджерами для нагадувань, модулем фінансового обліку чи системою CRM-аналітики).

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 47   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ СТОМАТОЛОГІЧНОЇ КЛІНІКИ

### 3.1. Порядок встановлення та налаштування параметрів системи

Розгортання вебдодатка стоматологічної клініки здійснюється шляхом виконання послідовності підготовчих кроків, що включають налаштування середовища виконання, конфігурацію зовнішніх сервісів та ініціалізацію бази даних.

Для опису архітектури розгортання додатку побудовано діаграму розгортання (рис. 3.1). Вона відображає взаємодію між клієнтською частиною (браузером), серверною частиною (Next.js), базою даних (Neon PostgreSQL) та хмарним сховищем медіафайлів (Cloudinary).

Обрана архітектура розгортання відповідає концепції serverless-хостингу: серверна частина не вимагає виділеного фізичного сервера та обслуговується платформою автоматично. Це суттєво знижує операційні витрати та спрощує масштабування у разі зростання навантаження. Взаємодія між компонентами відбувається виключно через HTTPS. Браузер надсилає запити до Next.js-сервера, який залежно від маршруту або повертає готову HTML-сторінку (SSR), або обробляє API-запит і повертає дані у форматі JSON. Запити до бази даних виконуються через Prisma ORM [26] з пулінгом з'єднань, що підвищує продуктивність при одночасній роботі кількох користувачів. Усі зовнішні сервіси — база даних Neon та сховище Cloudinary — підключаються через захищені з'єднання з автентифікацією за допомогою API-ключів, що зберігаються у змінних оточення та не потрапляють до публічного репозиторію.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 48   |

**Діаграма розгортання веб-додатка стоматологічної клініки**

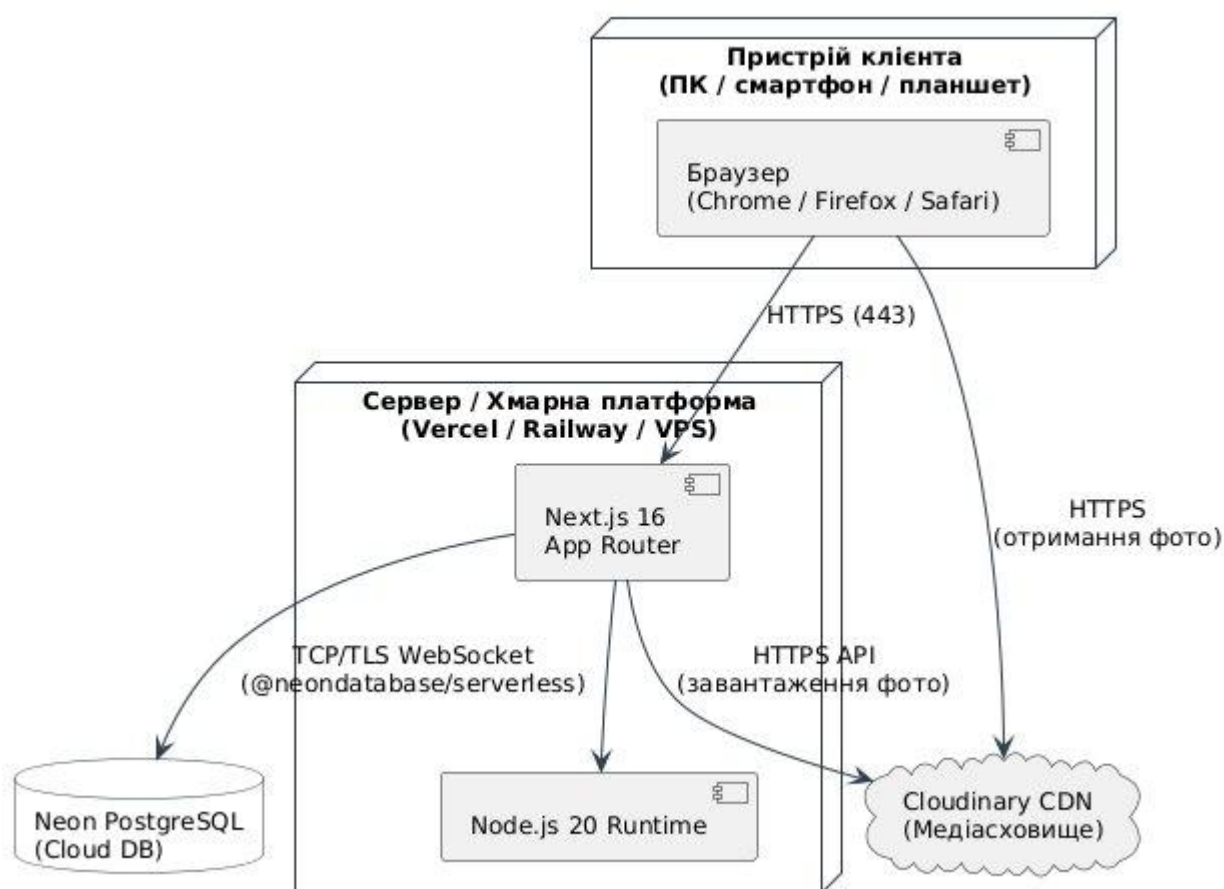


Рис. 3.1. Діаграма розгортання вебдодатка стоматологічної клініки

Архітектура розгортання вебдодатка охоплює чотири ключові компоненти, кожен з яких виконує власну роль у загальному стеку. Браузер клієнта звертається до вебдодатка через захищений протокол HTTPS і не потребує встановлення жодного додаткового програмного забезпечення на стороні користувача — для коректної роботи достатньо будь-якого сучасного браузера. Next.js-сервер є центральним компонентом архітектури: він обробляє вхідні HTTP-запити, виконує серверний рендеринг сторінок (SSR) та реалізує API-маршрути у форматі REST. Сервер може розгортатися як локально у середовищі Node.js, так і на хмарній платформі Vercel без додаткової конфігурації. Neon PostgreSQL є хмарною реляційною базою даних [13], яка підтримує стандарт SQL та забезпечує транзакційну цілісність даних; взаємодія з сервером здійснюється через захищене WebSocket-

з'єднання за допомогою адаптера `@neondatabase/serverless`. Cloudinary CDN є хмарним сервісом зберігання та оптимізованої доставки зображень: він використовується для збереження фотознімків, що прикріплюються до дентальних записів пацієнтів, і автоматично оптимізує розмір і формат зображень при завантаженні.

Перед початком розгортання необхідно підготувати середовище, встановивши відповідне програмне забезпечення. Середовище виконання Node.js [14] версії 20 LTS або новіше є обов'язковою умовою запуску Next.js-сервера; використання версії LTS гарантує стабільність та отримання патчів безпеки. Менеджер пакетів `npm` входить до складу дистрибутива Node.js і не потребує окремого встановлення — він використовується для завантаження та управління залежностями проєкту. Для зберігання та обробки реляційних даних необхідна СУБД PostgreSQL версії 14 або вище; в якості альтернативи можна скористатися обліковим записом хмарного сервісу Neon (<https://neon.tech>), що не вимагає локальної установки. Для роботи з медіафайлами потрібен обліковий запис сервісу Cloudinary — безкоштовний тариф є повністю достатнім на етапі розробки та тестування. Система контролю версій Git необхідна для отримання вихідного коду з репозиторію та подальшого управління змінами у проєкті.

Процес встановлення та налаштування виконується у визначеній послідовності. Спочатку вихідний код проєкту отримують із репозиторію командою `git clone`. Потім у кореневій директорії виконується `npm install` — менеджер пакетів завантажує всі залежності, перелічені у файлі `package.json`, серед яких `next 16.2.3`, `react 19.2.4`, `prisma 7.7.0`, `jose 6.0`, `bcryptjs 3.0`, `tailwindcss 4.0` та `cloudinary 2.6`. Наступним кроком налаштовують змінні оточення: у корені проєкту створюється файл `.env`, до якого вносяться рядок підключення до бази даних, секретний ключ для підпису токенів та облікові дані сервісу Cloudinary.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 50   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Перелік обов'язкових змінних оточення наведено у таблиці 3.1.

Таблиця 3.1

Змінні оточення вебдодатка стоматологічної клініки

| Змінна оточення       | Призначення                                                       | Приклад значення                   |
|-----------------------|-------------------------------------------------------------------|------------------------------------|
| DATABASE_URL          | Рядок підключення до PostgreSQL (Neon cloud або локальний сервер) | postgresql://user:pass@host/dental |
| JWT_SECRET            | Секретний ключ для підпису JWT-токенів (мін. 32 символи)          | supersecretkeymin32chars           |
| CLOUDINARY_CLOUD_NAME | Назва хмарного акаунту Cloudinary                                 | my-cloud-name                      |
| CLOUDINARY_API_KEY    | Публічний ключ API Cloudinary                                     | 123456789012345                    |
| CLOUDINARY_API_SECRET | Секретний ключ API Cloudinary                                     | abcXYZsecret...                    |

Коли файл .env заповнено, виконується команда `npm run prisma migrate dev`. Вона застосовує всі підготовлені міграції та створює у базі даних необхідні таблиці разом зі зв'язками й обмеженнями цілісності. За потреби базу можна наповнити початковими даними за допомогою сценарію `seed`, який створює обліковий запис адміністратора, а також окремого сценарію для облікових записів медичних асистентів. Запуск застосунку в режимі розробки здійснюється командою `npm run dev`, після чого сервер стає доступним на

порту 3000. Для промислового використання проєкт спочатку збирається командою `npm run build`, а потім запускається через `npm start`.

Після успішного розгортання система автоматично розмежовує доступ на основі ролей. Перший вхід для ролей Адміністратор, Лікар та Асистент виконується за обліковими даними, які занесено до бази через команду `seed` або вручну. Пацієнти реєструються самостійно через форму реєстрації (`/register`). Після авторизації сервер видає підписаний JWT-токен із зашифрованою роллю та ідентифікатором, який зберігається у `HttpOnly cookie`, захищаючи від XSS-атак. На підставі ролі система перенаправляє користувача до відповідного робочого простору.

### 3.2. Структура інтерфейсу вебдодатка стоматологічної клініки

Інтерфейс вебдодатка побудований за єдиним архітектурним принципом: бічна панель навігації зліва та робоча область із контентом справа. Кожна роль має власну кольорову схему панелі: синій — пацієнт, бірюзовий — лікар, темно-сірий — адміністратор, фіолетовий — асистент. Такий підхід дозволяє миттєво ідентифікувати поточний контекст роботи без додаткових підказок.

Сторінка входу є єдиною точкою входу для всіх категорій користувачів. Вона містить форму з двома полями (`email` та `пароль`), підпис «Вхід до системи» та логотип клініки. Система автоматично визначає роль на основі облікових даних і перенаправляє до відповідного робочого простору. У разі помилки форма відображає повідомлення. Посилання на реєстрацію доступне лише для нових пацієнтів (рис. 3.2).

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 52   |

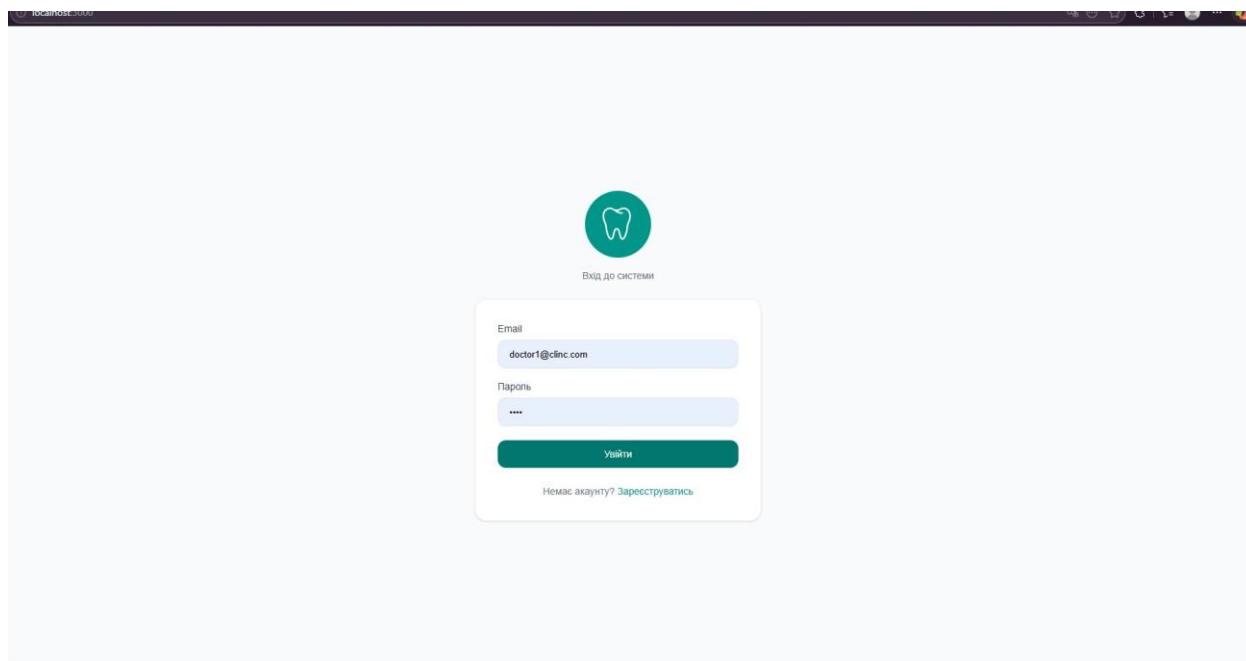


Рис. 3.2. Сторінка входу до системи

Сторінка реєстрації (/register) призначена виключно для нових пацієнтів. Форма містить поля: ім'я та прізвище, email, пароль, телефон, дата народження та стать. Після успішної реєстрації система автоматично авторизує користувача і перенаправляє до особистого кабінету (рис. 3.3).

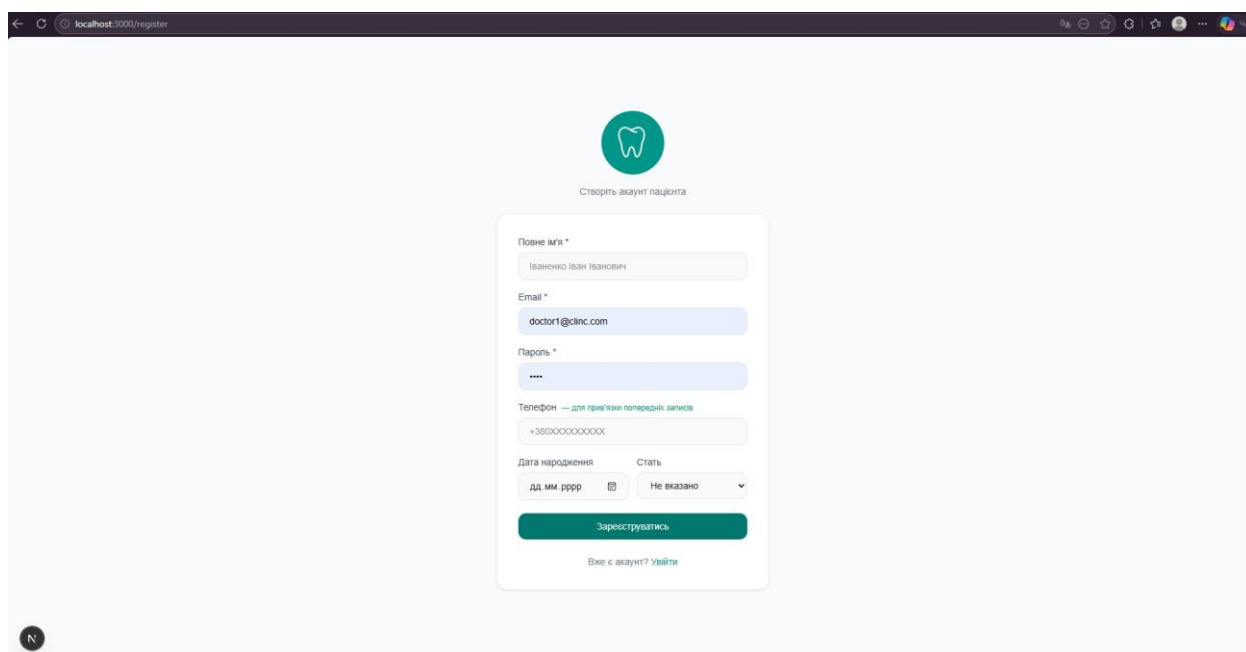


Рис. 3.3. Сторінка реєстрації пацієнта

Робочий простір пацієнта (синя бічна панель) складається із шести розділів. Головна сторінка (/patient/dashboard) надає персоналізоване

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 53   |

зведення: найближчий запланований прийом із датою, часом, ім'ям лікаря та типом прийому; список майбутніх візитів (до 4 позицій); останні рекомендації від лікарів. За відсутності запланованих прийомів система пропонує перейти до бронювання (рис. 3.4).

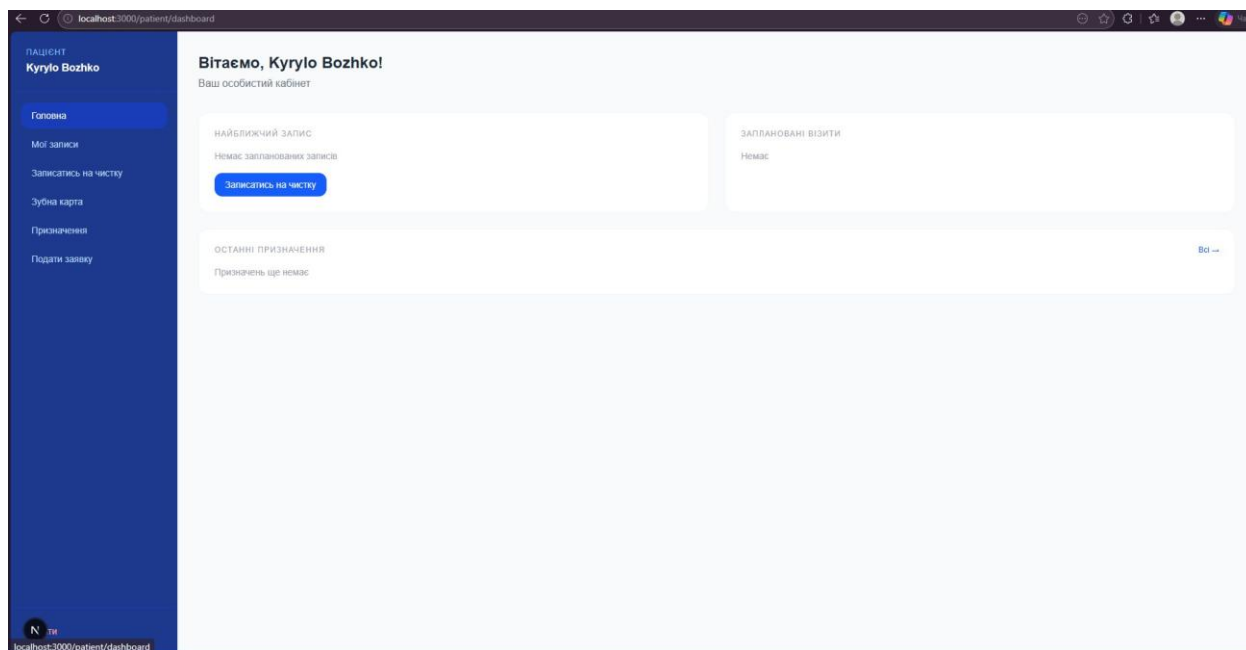


Рис. 3.4. Головна сторінка кабінету пацієнта

Сторінка «Записатись» (/patient/book) реалізує процес онлайн-запису на прийом. Вона містить випадаючий список для вибору лікаря та тижневий календар-сітку з годинними слотами (8:00–18:00). Доступні слоти виділяються кольором і є клікабельними. При виборі слота відкривається вікно підтвердження з вибором типу прийому (рис. 3.5).

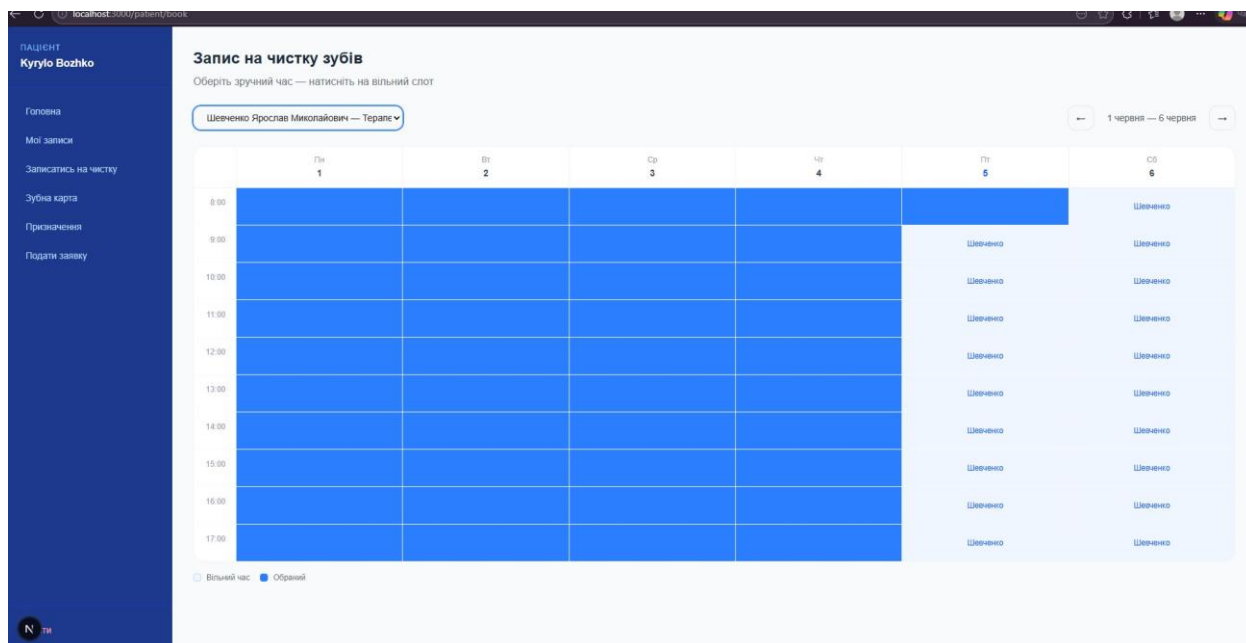


Рис. 3.5. Сторінка онлайн-запису пацієнта на прийом

Сторінка «Зубна карта» (/patient/dental-chart) надає пацієнту перегляд власної електронної медичної карти у формі інтерактивної SVG-схеми 32 зубів. Зуби з наявними медичними записами виділяються бірюзовим кольором. Натискання на зуб розкриває список усіх маніпуляцій: дата, лікар, діагноз, лікування та фотознімки (рис. 3.6).

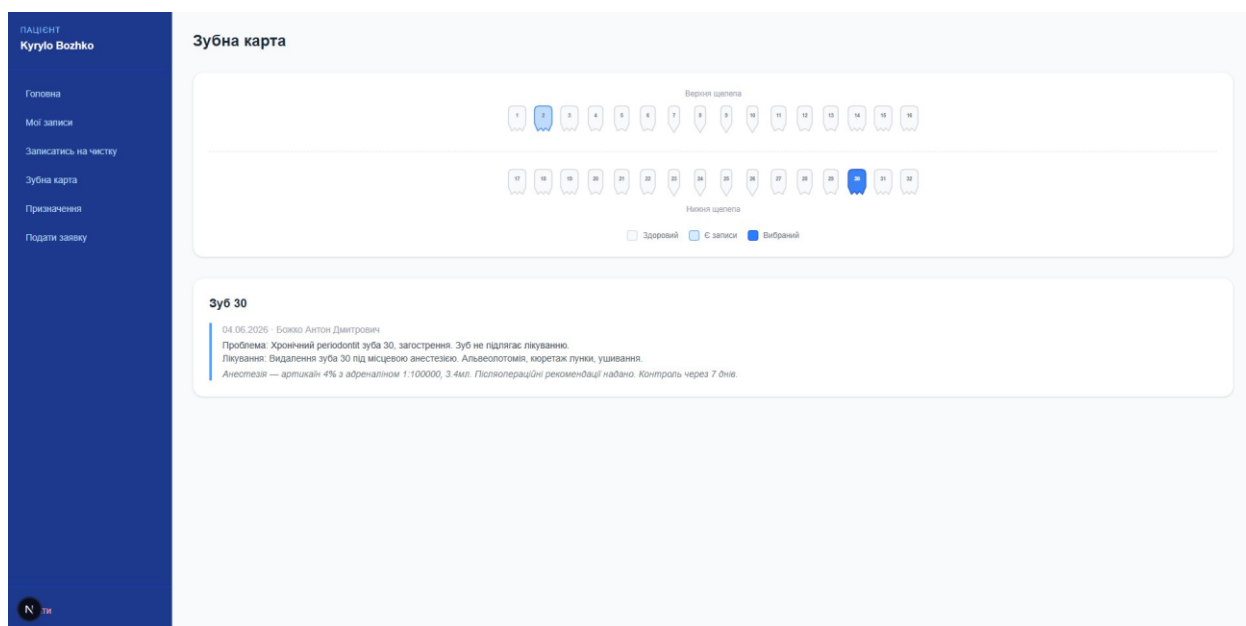


Рис. 3.6. Зубна карта пацієнта (режим перегляду)

Сторінка «Призначення» (/patient/prescriptions) відображає хронологічний список усіх рекомендацій від лікарів — текст призначення, ім'я лікаря та дата. Сторінка «Подати заявку» (/patient/request) — форма довільного звернення до клініки без прив'язки до конкретного слота розкладу (наприклад, термінове звернення або прохання перенести прийом).

Робочий простір лікаря (бірюзова бічна панель) містить сім розділів. Головна сторінка «Мої пацієнти» (/doctor/dashboard) показує перелік прийомів на поточний день: ім'я пацієнта (або мітка «без акаунту» для гостьових записів), час прийому, тип, телефон та дата народження. Кнопка «Зателефонував?» відмічає підтвердження запису; кнопка «Протокол» відкриває сторінку операційного протоколу (рис. 3.7).

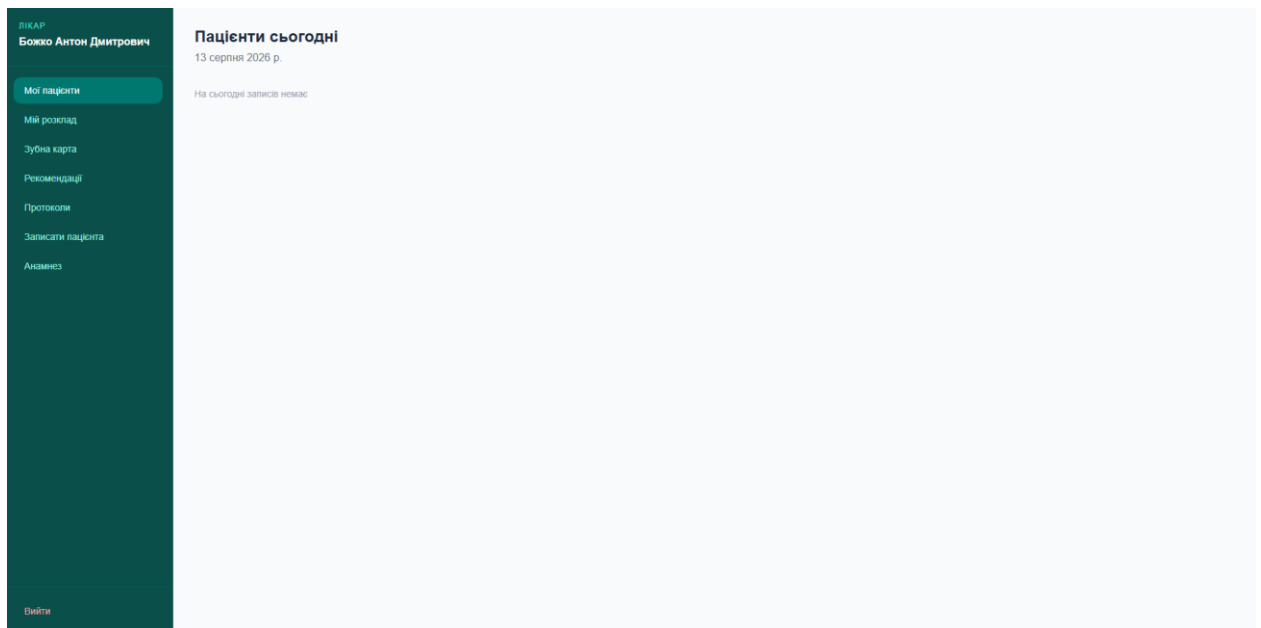


Рис. 3.7. Головна сторінка кабінету лікаря — прийоми поточного дня

Сторінка Dental Chart (/doctor/dental-chart) є центральним інструментом медичної документації. Лікар обирає пацієнта зі списку, після чого відображається інтерактивна SVG-схема 32 зубів із трьома різними формами коронок (молярна, різцева, стандартна). Зуби з наявними записами виділені кольором. Натискання на зуб розкриває форму додавання нового запису та перелік попередніх маніпуляцій. Фотознімки завантажуються через Cloudinary і відображаються у вбудованому перегляді з режимом lightbox (рис. 3.8).

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 56   |

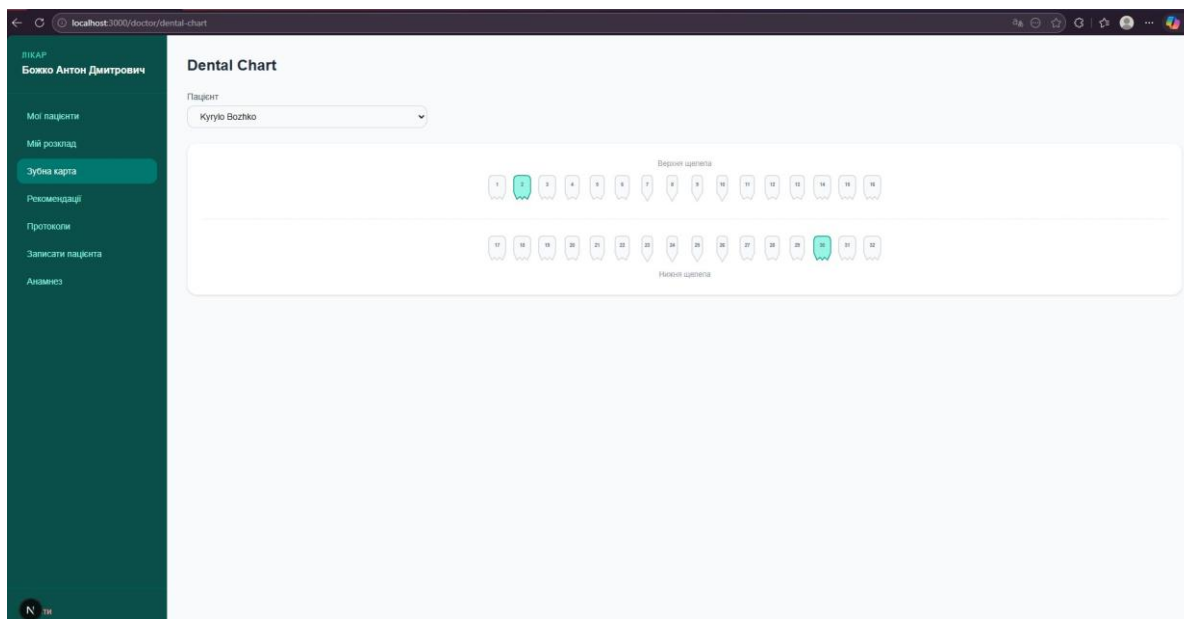


Рис. 3.8. Інтерактивна зубна карта (робочий простір лікаря)

Сторінка «Анамнез» (/doctor/anamnesis) містить форму медичного анамнезу: група крові (O(I)+, O(I)–, A(II)+, A(II)–, B(III)+, B(III)–, AB(IV)+, AB(IV)–), алергії, хронічні захворювання, препарати, попередні операції, ознака вагітності/годування та нотатки. Анамнез зберігається один раз на пацієнта і доступний для редагування у будь-який момент (рис. 3.9).

Рис. 3.9. Сторінка анамнезу пацієнта (робочий простір лікаря)

Сторінка «Протоколи» (/doctor/protocols) надає лікарю перелік операційних протоколів усіх його прийомів. Кожен протокол фіксує

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 57   |

хронологію подій під час прийому (введення анестетиків, медикаментів, виконані процедури), занесених асистентом у реальному часі.

Робочий простір адміністратора (темно-сіра бічна панель) включає шість розділів. Головна сторінка (/admin/dashboard) відображає п'ять статистичних карток: кількість лікарів, асистентів, пацієнтів, нових заявок та прийомів сьогодні (рис. 3.10).

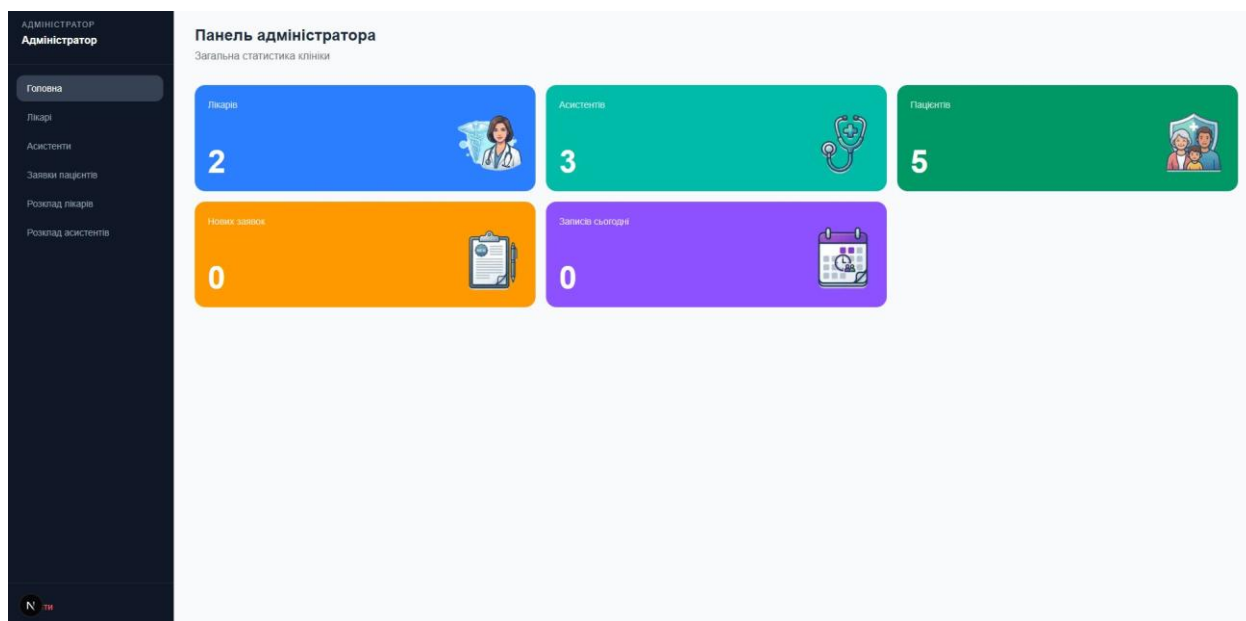


Рис. 3.10. Панель адміністратора — загальна статистика клініки

Сторінка «Лікарі» (/admin/doctors) реалізує повне CRUD-управління штатом лікарів: перегляд переліку з ім'ям, спеціалізацією та телефоном; додавання нового лікаря через форму; редагування та видалення. Аналогічну структуру має сторінка «Асистенти» (/admin/assistants) — для управління обліковими записами медичних асистентів (рис. 3.11).

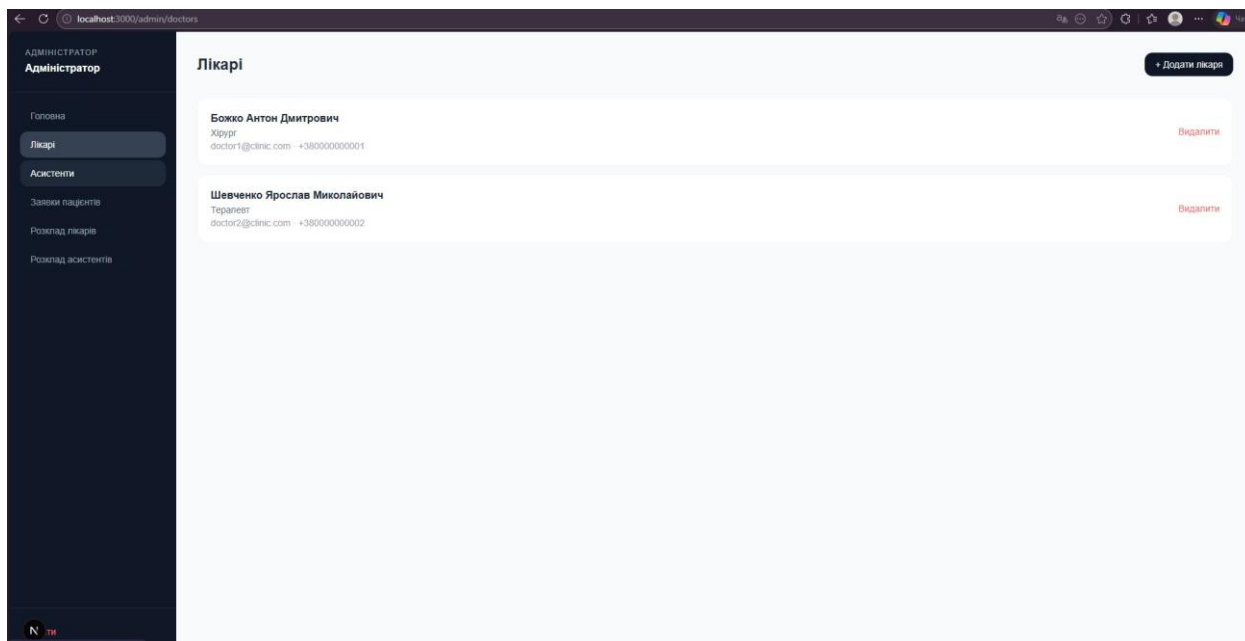


Рис. 3.11. Сторінка управління лікарями (адміністратор)

Сторінка «Розклад лікарів» (/admin/schedule) — централізоване управління часовими слотами. Адміністратор може створювати нові слоти (лікар + дата + час), переглядати та видаляти слоти, записувати зареєстрованих і гостьових пацієнтів, а також управляти відпустками лікарів (блокування дат). Сторінка «Розклад асистентів» (/admin/assistant-schedule) дозволяє призначати асистентів до конкретних лікарів на конкретні дати (рис. 3.12).

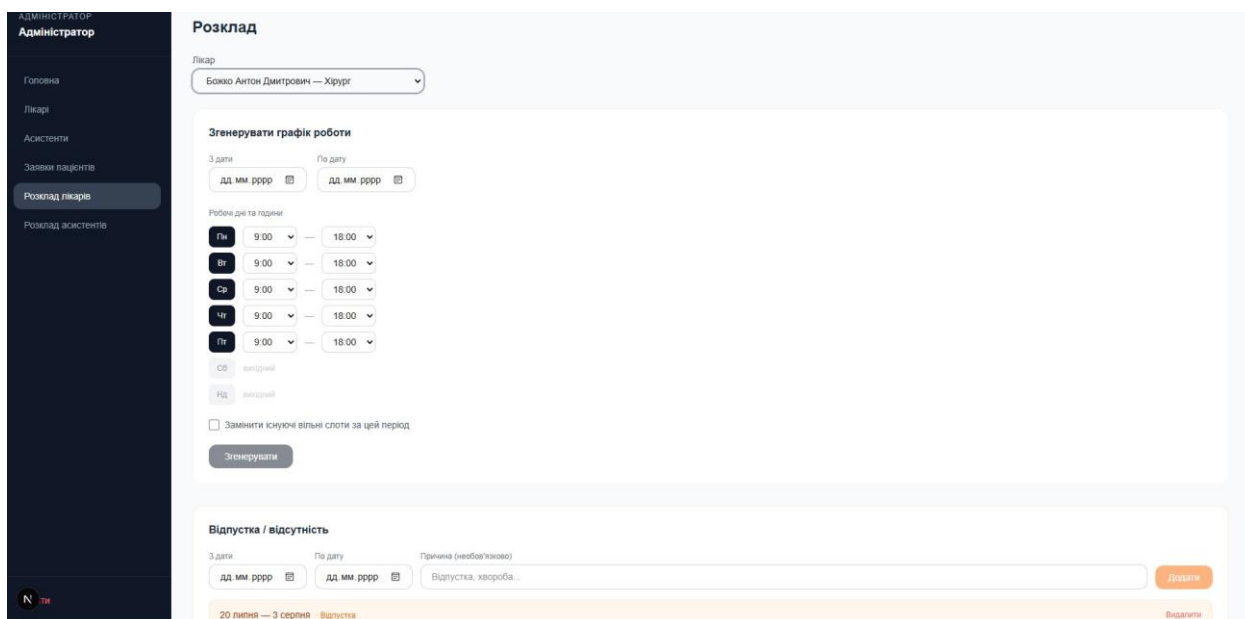


Рис. 3.12. Сторінка управління розкладом лікарів

Сторінка «Заявки пацієнтів» (/admin/requests) відображає чергу необроблених звернень, відсортовану за датою. Кнопка «Опрацьовано» змінює статус заявки з NEW на PROCESSED і прибирає її з черги активних звернень (рис. 3.13).



Рис. 3.13. Черга заявок пацієнтів (адміністратор)

Робочий простір асистента (фіолетова бічна панель) складається з двох компонентів. Головна сторінка «Мій розклад» (/assistant/dashboard) відображає тижневий перелік призначень з навігацією між тижнями. Кожне призначення показує: дату (поточний день виділяється фіолетовим), ім'я та спеціалізацію лікаря, список прийомів із часом і кнопкою переходу до протоколу (рис. 3.14).

Адміністратор

Адміністратор

Головна

Лікарі

Асистенти

Записи пацієнтів

Розклад лікарів

Розклад асистентів

N 78

Графіки асистентів

Асистент

Бондаренко Андрій Олександрович

1 червня — 7 червня

Призначити графік

Лікар

Оберіть лікаря

З дати

По дати

ДД.ММ.РРРР

ДД.ММ.РРРР

Робочі дні

Пн

Вт

Ср

Чт

Пт

Сб

Нд

Призначити

Відпустка / відсутність

З дати

По дати

ДД.ММ.РРРР

ДД.ММ.РРРР

Причина (необов'язково)

Відпустка, хвороба...

Додати

23 червня — 30 червня

Відсутність

середа, 3 червня

Ковальчук Олена Василівна — Божко Антон Дмитрович Хірург

×

четвер, 4 червня

Ковальчук Олена Василівна — Божко Антон Дмитрович Хірург

×

Ковальчук Олена Василівна — Божко Антон Дмитрович Хірург

×

Рис. 3.14. Тижневий розклад асистента

Сторінка операційного протоколу (/assistant/protocol/[id]) — основний робочий інструмент асистента. У режимі реального часу фіксуються події прийому: введення анестетику (з дозою), виконані маніпуляції, медикаменти, медіафайли. Кожен запис отримує автоматичну мітку часу. Після завершення прийому протокол закривається і стає доступним лише для читання (рис. 3.15).

Асистент

Ковальчук Олена Василівна

Мій день

N 78

← Назад

Протокол операції

Прийом #11 8 проєкт

Закрити протокол

Дія

Медикамент

Фото/Відео

Назва препарату

Артикаїн 4%

Доза

1.7мл

Додати медикамент

11:21 Дія

Проведено огляд порожнини рота, зуб 36 має глибокий карієс з ураженням пульпи

11:22 Дія

Нанесено ізоляцію робочого поля, встановлено кофердами

11:22 Медикамент

Артикаїн 4% — 1.7 мл — 1.7 мл

Рис. 3.15. Сторінка операційного протоколу (асистент)

Усі робочі простори захищені однаковим механізмом: при кожному завантаженні сторінки перевіряється JWT-токен через /api/auth/me. Відсутній

або прострочений токен, а також невідповідність ролі — ведуть до перенаправлення на сторінку входу. Це унеможливорює несанкціонований доступ навіть при прямому введенні URL у адресний рядок браузера.

### 3.3. Тестування роботи програмного комплексу

Тестування вебдодатка стоматологічної клініки проводилося методом функціонального чорного ящика (black-box testing): перевірялася відповідність поведінки системи специфікованим вимогам без врахування внутрішньої реалізації. Цей метод дозволяє верифікувати коректність роботи з позиції кінцевого користувача.

Тестування охоплювало всі чотири ролі (пацієнт, лікар, адміністратор, асистент) та ключові сценарії: автентифікацію, онлайн-запис, роботу з дентальною картою, управління розкладом, обробку заявок і ведення операційних протоколів. Для кожного сценарію визначалися вхідні дані, очікуваний і фактичний результати.

Результати тестування модуля автентифікації наведено у таблиці 3.2.

Модуль автентифікації тестувався за такими сценаріями: реєстрація нового пацієнта із валідацією форми, вхід до системи з коректними та некоректними обліковими даними, захист маршрутів від несанкціонованого доступу, автоматичне оновлення JWT-токена та розмежування прав доступу між чотирма ролями. Усього виконано 18 тест-кейсів, з яких 18 пройдено успішно.

Таблиця 3.2

Тестування модуля автентифікації

| № | Тестовий сценарій                | Вхідні дані                                     | Очікуваний результат                     | Факт. результат |
|---|----------------------------------|-------------------------------------------------|------------------------------------------|-----------------|
| 1 | Вхід пацієнта<br>(коректні дані) | Email:<br>patient@test.com,<br>Пароль: Test1234 | Перенаправлення на<br>/patient/dashboard | Пройдено        |

Продовж. табл. 3.2

|    |                                            |                                               |                                                      |          |
|----|--------------------------------------------|-----------------------------------------------|------------------------------------------------------|----------|
| 2  | Вхід з некоректним паролем                 | Email: patient@test.com,<br>Пароль: wrong     | Повідомлення «Невірний пароль»                       | Пройдено |
| 3  | Вхід неіснуючого користувача               | Email: no@test.com,<br>Пароль: Test1234       | Повідомлення «Користувача не знайдено»               | Пройдено |
| 4  | Вхід лікаря                                | Email: doctor@test.com,<br>Пароль: DocPass1   | Перенаправлення на /doctor/dashboard                 | Пройдено |
| 5  | Вхід адміністратора                        | Email: admin@clinic.com,<br>Пароль: admin1234 | Перенаправлення на /admin/dashboard                  | Пройдено |
| 6  | Реєстрація нового пацієнта (коректні дані) | Усі поля заповнені коректно                   | Авторизація та перенаправлення на /patient/dashboard | Пройдено |
| 7  | Реєстрація з існуючим email                | Email уже наявний у базі                      | Повідомлення «Email вже використовується»            | Пройдено |
| 8  | Вихід із системи                           | Кнопка «Вийти»                                | Видалення cookie, перенаправлення на сторінку входу  | Пройдено |
| 9  | Доступ до /admin без авторизації           | Прямий URL без токена                         | Перенаправлення на сторінку входу                    | Пройдено |
| 10 | Доступ пацієнта до /admin                  | Пацієнт вводить URL адмін-панелі              | Перенаправлення на сторінку входу                    | Пройдено |

Результати тестування онлайн-запису наведено у таблиці 3.3.

Таблиця 3.3

Тестування модуля онлайн-запису на прийом

| № | Тестовий сценарій                  | Вхідні дані                               | Очікуваний результат                             | Факт. результат |
|---|------------------------------------|-------------------------------------------|--------------------------------------------------|-----------------|
| 1 | Перегляд усіх вільних слотів       | Відкриття /patient/book без вибору лікаря | Відображення слотів усіх лікарів на тиждень      | Пройдено        |
| 2 | Фільтр по конкретному лікарю       | Вибір лікаря у списку                     | Відображення лише слотів обраного лікаря         | Пройдено        |
| 3 | Успішне бронювання слота           | Вільний слот, тип «Огляд», підтвердження  | Прийом зі статусом CONFIRMED, слот isBooked=true | Пройдено        |
| 4 | Навігація між тижнями              | Кнопка «→»                                | Відображення слотів наступного тижня             | Пройдено        |
| 5 | Паралельне бронювання одного слота | Дві сесії одночасно обирають один слот    | Перший запит успішний, другий — помилка          | Пройдено        |

Результати тестування дентальної карти наведено у таблиці 3.4.

Таблиця 3.4

Тестування модуля дентальної карти

| № | Тестовий сценарій                 | Вхідні дані                                   | Очікуваний результат                            | Факт. результат |
|---|-----------------------------------|-----------------------------------------------|-------------------------------------------------|-----------------|
| 1 | Вибір пацієнта (лікар)            | Вибір зі списку на /doctor/dental-chart       | SVG-схема із виділеними зубами, що мають записи | Пройдено        |
| 2 | Додавання запису по зубу          | Зуб, поля «Проблема», «Лікування», «Коментар» | Запис у базі, зуб змінює колір на схемі         | Пройдено        |
| 3 | Завантаження фото до запису       | Вибір зображення через форму                  | Збережено у Cloudinary, URL у базі даних        | Пройдено        |
| 4 | Перегляд карти пацієнтом          | Пацієнт відкриває /patient/dental-chart       | Схема зубів у режимі лише читання               | Пройдено        |
| 5 | Збереження без обов'язкових полів | Порожнє поле «Проблема» або «Лікування»       | Помилка валідації, запис не збережено           | Пройдено        |

Результати тестування функцій адміністратора наведено у таблиці 3.5.

Таблиця 3.5

Тестування функцій адміністратора

| № | Тестовий сценарій        | Вхідні дані                         | Очікуваний результат                    | Факт. результат |
|---|--------------------------|-------------------------------------|-----------------------------------------|-----------------|
| 1 | Додавання лікаря         | Ім'я, спеціалізація, email, пароль  | Лікар у списку, може увійти до системи  | Пройдено        |
| 2 | Редагування лікаря       | Зміна спеціалізації                 | Дані оновлені в базі та у переліку      | Пройдено        |
| 3 | Видалення лікаря         | Підтвердження видалення             | Лікар та його слоти видалені (каскадно) | Пройдено        |
| 4 | Створення слота розкладу | Лікар, дата, час                    | Слот у розкладі зі статусом «вільний»   | Пройдено        |
| 5 | Гостьовий запис пацієнта | Вільний слот, ім'я та телефон гостя | Прийом створено, слот заброньовано      | Пройдено        |
| 6 | Обробка заявки пацієнта  | Кнопка «Опрацьовано»                | Статус PROCESSED, заявка зникла з черги | Пройдено        |
| 7 | Перегляд статистики      | Відкриття /admin/dashboard          | Актуальні показники всіх п'яти метрик   | Пройдено        |

Результати тестування роботи асистента наведено у таблиці 3.6.

Таблиця 3.6

Тестування функцій асистента

| № | Тестовий сценарій             | Вхідні дані                            | Очікуваний результат                           | Факт. результат |
|---|-------------------------------|----------------------------------------|------------------------------------------------|-----------------|
| 1 | Перегляд тижневого розкладу   | Відкриття<br>/assistant/dashboard      | Призначення тижня; день «сьогодні» розгорнутий | Пройдено        |
| 2 | Навігація між тижнями         | Кнопки «←» та «→»                      | Перехід до попереднього/наступного тижня       | Пройдено        |
| 3 | Відкриття протоколу прийому   | Кнопка «Протокол» у списку прийомів    | Сторінка протоколу обраного прийому            | Пройдено        |
| 4 | Додавання запису до протоколу | Тип «Анестетик», доза «2 мл лідокаїну» | Запис із міткою часу у хронологічному журналі  | Пройдено        |
| 5 | Закриття протоколу            | Кнопка «Закрити протокол»              | Статус CLOSED, форма запису заблокована        | Пройдено        |

Усі 37 тестових сценаріїв завершилися успішно — фактичні результати відповідають очікуваним. Критичних помилок у роботі системи не виявлено. Заходи безпеки (розмежування доступу, валідація даних, захист від паралельного бронювання) підтвердили свою коректність. Інтерфейс коректно відображається на всіх перевірених роздільних здатностях: 1920×1080, 1366×768 та 768×1024 пікселів.

## Висновки до третього розділу

У третьому розділі кваліфікаційної роботи описано повний порядок встановлення та налаштування вебдодатка стоматологічної клініки: перелік необхідного програмного забезпечення, конфігурацію змінних оточення, ініціалізацію бази даних через систему міграцій Prisma та порядок запуску застосунку. Побудовано діаграму розгортання, що ілюструє взаємодію клієнта, Next.js-сервера, хмарної бази даних Neon та сховища медіафайлів Cloudinary.

Детально описано структуру інтерфейсу всіх чотирьох робочих просторів: пацієнта (6 сторінок), лікаря (7 сторінок, включно з анамнезом і протоколами), адміністратора (6 сторінок) та асистента (розклад і операційні протоколи). Особливу роль займають інтерактивна SVG-схема 32 зубів, модуль операційних протоколів для хронологічного ведення записів асистентом та модуль анамнезу для зберігання медичної історії пацієнта.

Проведено функціональне тестування методом чорного ящика із 37 тестовими сценаріями у шести групах: автентифікація, онлайн-запис, дентальна карта, функції адміністратора, функції асистента та кросбраузерне відображення. Усі сценарії успішно пройдено, що підтверджує відповідність програмного комплексу функціональним вимогам та його готовність до практичного використання.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 68   |

## ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну задачу автоматизації управління стоматологічною клінікою шляхом розробки повнофункціонального вебдодатка на базі сучасного технологічного стеку.

За результатами виконання роботи отримано такі результати.

У першому розділі проведено аналіз предметної області та ринку існуючих програмних рішень. Детально розглянуто три провідні платформи — Dentrix, Open Dental і Curve Dental. Встановлено, що жодна з них не відповідає потребам вітчизняних приватних клінік через відсутність української локалізації, високу вартість ліцензування та відсутність модуля операційних протоколів для асистентів. Обґрунтовано доцільність розробки власного рішення і обрано технологічний стек: Next.js 16, TypeScript, PostgreSQL (Neon), Prisma ORM, Tailwind CSS 4, Cloudinary, jose та bcryptjs.

У другому розділі виконано проєктування системи: визначено чотири ролі користувачів (пацієнт, лікар, асистент, адміністратор) і описано вісім ключових прецедентів використання. Спроектовано реляційну базу даних із 14 таблицями — охоплено всі сутності від медичного анамнезу та дентальних фото до операційних протоколів та відпусток лікарів. Описано сім алгоритмів роботи системи та реалізовано чотири робочих простори з відповідними API-маршрутами.

У третьому розділі описано порядок розгортання системи, наведено огляд інтерфейсу та проведено функціональне тестування методом чорного ящика. Перевірено 37 тестових сценаріїв у шести групах — усі завершилися успішно. Підтверджено коректність розмежування доступу, захист від паралельного бронювання та адаптивність інтерфейсу на різних роздільних здатностях.

Реалізований програмний продукт автоматизує повний цикл роботи клініки: онлайн-запис пацієнтів, ведення дентальних карток з прикріпленням фото, медичний анамнез, операційні протоколи асистентів у реальному часі,

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 69   |

автоматичну генерацію розкладу лікарів, управління відпустками та обробку звернень пацієнтів. Система побудована на відкритому програмному забезпеченні, що робить її економічно доступною для малих та середніх клінік.

Подальший розвиток системи може включати: модуль фінансового обліку, інтеграцію з месенджерами для автоматичних нагадувань пацієнтам, мобільний клієнт на React Native та систему CRM-аналітики.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 70   |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dentrrix. URL: <https://www.dentrrix.com/> (дата звернення: 28.05.2026).
2. Open Dental. URL: <https://www.opendental.com/> (дата звернення: 28.05.2026).
3. Curve Dental. URL: <https://www.curvedental.com/> (дата звернення: 28.05.2026).
4. Next.js Documentation. URL: <https://nextjs.org/docs> (дата звернення: 15.05.2026).
5. TypeScript Handbook. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 15.05.2026).
6. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 17.05.2026).
7. Prisma ORM Documentation. URL: <https://www.prisma.io/docs/> (дата звернення: 17.05.2026).
8. Tailwind CSS Documentation. URL: <https://tailwindcss.com/docs> (дата звернення: 20.05.2026).
9. Cloudinary Documentation. URL: <https://cloudinary.com/documentation> (дата звернення: 20.05.2026).
10. jose — JSON Web Tokens library. URL: <https://github.com/panva/jose> (дата звернення: 22.05.2026).
11. bcryptjs. URL: <https://github.com/dcodeIO/bcrypt.js> (дата звернення: 22.05.2026).
12. React Documentation. URL: <https://react.dev/> (дата звернення: 15.05.2026).
13. Neon Serverless PostgreSQL Documentation. URL: <https://neon.tech/docs/> (дата звернення: 21.05.2026).
14. Node.js Documentation. URL: <https://nodejs.org/en/docs/> (дата звернення: 18.05.2026).

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 71   |

15. ESLint Documentation. URL: <https://eslint.org/docs/> (дата звернення: 18.05.2026).
16. Next.js App Router. URL: <https://nextjs.org/docs/app> (дата звернення: 16.05.2026).
17. JSON Web Token Introduction. JWT.io. URL: <https://jwt.io/introduction/> (дата звернення: 23.05.2026).
18. OWASP Top Ten. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 25.05.2026).
19. UML 2.5 Specification. Object Management Group. URL: <https://www.omg.org/spec/UML/> (дата звернення: 12.05.2026).
20. Osmani A. Learning JavaScript Design Patterns. 2nd ed. Sebastopol : O'Reilly Media, 2023. 252 p.
21. Martin R. C. Clean Craftsmanship: Disciplines, Standards, and Ethics / R. C. Martin. Boston : Addison-Wesley Professional, 2021. 416 p.
22. Fielding R. T., Nottingham M., Reschke J. HTTP Semantics : RFC 9110. Internet Engineering Task Force, 2022. URL: <https://www.rfc-editor.org/rfc/rfc9110> (дата звернення: 20.05.2026).
23. Shvets A. Dive Into Design Patterns. Refactoring.Guru, 2021. URL: <https://refactoring.guru/design-patterns> (дата звернення: 20.05.2026).
24. ISO/IEC 25010:2023. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model. Geneva : ISO, 2023. 26 p.
25. What is Use Case Diagram? Visual Paradigm. URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> (дата звернення: 12.05.2026).
26. Prisma Schema Reference. URL: <https://www.prisma.io/docs/reference/api-reference/prisma-schema-reference> (дата звернення: 17.05.2026).

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 72   |

27. React Hooks Reference. URL: <https://react.dev/reference/react> (дата звернення: 15.05.2026).

28. Vercel Deployment Documentation. URL: <https://vercel.com/docs> (дата звернення: 27.05.2026).

29. HTTP Cookies. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies> (дата звернення: 24.05.2026).

30. SVG 2 Specification. W3C. URL: <https://www.w3.org/TR/SVG2/> (дата звернення: 13.05.2026).

31. NIST Special Publication 800-63B. Digital Identity Guidelines: Authentication and Lifecycle Management. National Institute of Standards and Technology, 2024. URL: <https://pages.nist.gov/800-63-4/> (дата звернення: 22.05.2026).

32. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley Professional, 2021. 592 p.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 73   |

# ДОДАТКИ

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 74   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## Схема бази даних (schema.prisma)

```

...
...

generator client {
  provider = "prisma-client"
  output = "../src/app/generated/prisma"
}

datasource db {
  provider = "postgresql"
}

enum Role {
  PATIENT
  DOCTOR
  ADMIN
}

enum AppointmentStatus {
  PENDING
  CONFIRMED
  COMPLETED
  CANCELLED
}

enum RequestStatus {
  NEW
  PROCESSED
}

model User {
  id Int @id @default(autoincrement())
  email String @unique
  passwordHash String
  name String
  phone String?
  birthDate DateTime?
  gender String?
  role Role @default(PATIENT)
  createdAt DateTime @default(now())
  appointments Appointment[]
  requests PatientRequest[]

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 75   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```
dentalRecords DentalRecord[]
recommendations Recommendation[]
anamnesis Anamnesis?
}
```

```
model Doctor {
id Int @id @default(autoincrement())
email String @unique
passwordHash String
name String
specialty String
phone String?
createdAt DateTime @default(now())
appointments Appointment[]
scheduleSlots ScheduleSlot[]
dentalRecords DentalRecord[]
recommendations Recommendation[]
assignments AssistantAssignment[]
anamneses Anamnesis[]
vacations Vacation[]
}
```

```
model Assistant {
id Int @id @default(autoincrement())
email String @unique
passwordHash String
name String
phone String?
createdAt DateTime @default(now())
assignments AssistantAssignment[]
protocols OperationProtocol[]
vacations AssistantVacation[]
}
```

```
model AssistantAssignment {
id Int @id @default(autoincrement())
assistant Assistant @relation(fields: [assistantId], references: [id], onDelete: Cascade)
assistantId Int
doctor Doctor @relation(fields: [doctorId], references: [id], onDelete: Cascade)
doctorId Int
date DateTime
}
```

```
@@unique([assistantId, date])
```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 76   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

}

```
model ScheduleSlot {  
  id Int @id @default(autoincrement())  
  doctor Doctor @relation(fields: [doctorId], references: [id], onDelete: Cascade)  
  doctorId Int  
  date DateTime  
  isBooked Boolean @default(false)  
  appointment Appointment?  
}
```

```
model Appointment {  
  id Int @id @default(autoincrement())  
  patient User? @relation(fields: [patientId], references: [id])  
  patientId Int?  
  guestName String?  
  guestPhone String?  
  doctor Doctor @relation(fields: [doctorId], references: [id])  
  doctorId Int  
  slot ScheduleSlot @relation(fields: [slotId], references: [id])  
  slotId Int @unique  
  type String  
  status AppointmentStatus @default(CONFIRMED)  
  duration Int @default(1)  
  calledBack Boolean @default(false)  
  notes String?  
  createdAt DateTime @default(now())  
  protocol OperationProtocol?  
}
```

```
model PatientRequest {  
  id Int @id @default(autoincrement())  
  patient User @relation(fields: [patientId], references: [id])  
  patientId Int  
  description String  
  status RequestStatus @default(NEW)  
  createdAt DateTime @default(now())  
}
```

```
model DentalRecord {  
  id Int @id @default(autoincrement())  
  patient User @relation(fields: [patientId], references: [id])  
  patientId Int  
  doctor Doctor @relation(fields: [doctorId], references: [id])  
}
```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 77   |

```

doctorId Int
toothNumber Int
problem String
treatment String
comments String?
photoUrl String?
photos DentalPhoto[]
date DateTime @default(now())
}

```

```

model DentalPhoto {
id Int @id @default(autoincrement())
dentalRecord DentalRecord @relation(fields: [dentalRecordId], references: [id],
onDelete: Cascade)
dentalRecordId Int
url String
createdAt DateTime @default(now())
}

```

```

model OperationProtocol {
id Int
@id @default(autoincrement())
appointment Appointment @relation(fields: [appointmentId], references: [id])
appointmentId Int @unique
assistant Assistant @relation(fields: [assistantId], references: [id])
assistantId Int
status String @default("OPEN")
closedAt DateTime?
entries ProtocolEntry[]
createdAt DateTime @default(now())
}

```

```

model ProtocolEntry {
id Int @id @default(autoincrement())
protocol OperationProtocol @relation(fields: [protocolId], references: [id],
onDelete: Cascade)
protocolId Int
type String
text String?
dose String?
mediaUrl String?
mediaType String?
timestamp DateTime @default(now())
}

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
|      |      |          |        |      |                          | 78   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

model Vacation {
  id Int @id @default(autoincrement())
  doctor Doctor @relation(fields: [doctorId], references: [id], onDelete: Cascade)
  doctorId Int
  startDate DateTime
  endDate DateTime
  reason String?
  createdAt DateTime @default(now())
}

```

```

model AssistantVacation {
  id Int @id @default(autoincrement())
  assistant Assistant @relation(fields: [assistantId], references: [id], onDelete: Cascade)
  assistantId Int
  startDate DateTime
  endDate DateTime
  reason String?
  createdAt DateTime @default(now())
}

```

```

model Anamnesis {
  id Int @id @default(autoincrement())
  patient User @relation(fields: [patientId], references: [id])
  patientId Int @unique
  doctor Doctor @relation(fields: [doctorId], references: [id])
  doctorId Int
  bloodType String?
  allergies String?
  chronicDiseases String?
  medications String?
  previousSurgeries String?
  pregnancyOrFeeding Boolean @default(false)
  notes String?
  updatedAt DateTime @updatedAt
  createdAt DateTime @default(now())
}

```

```

model Recommendation {
  id Int @id @default(autoincrement())
  patient User @relation(fields: [patientId], references: [id])
  patientId Int
  doctor Doctor @relation(fields: [doctorId], references: [id])
}

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 79   |

```

doctorId Int
text String
createdAt DateTime @default(now())
}
...

```

## Додаток Б

```

import { SignJWT, jwtVerify } from "jose";
import { cookies } from "next/headers";

const SECRET = new TextEncoder().encode(process.env.JWT_SECRET ??
"dental-clinic-secret-key");

export async function signToken(payload: { id: number; role: string }) {
  return new SignJWT(payload)
    .setProtectedHeader({ alg: "HS256" })
    .setExpirationTime("7d")
    .sign(SECRET);
}

export async function verifyToken(token: string) {
  const { payload } = await jwtVerify(token, SECRET);
  return payload as { id: number; role: string };
}

export async function getSession() {
  const cookieStore = await cookies();
  const token = cookieStore.get("token")?.value;
  if (!token) return null;
  try {
    return await verifyToken(token);
  } catch {
    return null;
  }
}

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 80   |