**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інженерії програмного забезпечення**

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____Євген ДАВИДЕНКО

«___» _____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

# ЗАСТОСУНОК СТОМАТОЛОГІЧНОЇ КЛІНІКИ З ФУНКЦІЄЮ ПРОГНОЗУВАННЯ ВІДВІДУВАННЯ ПАЦІЄНТІВ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

**Здобувач** _____ **Андрій ТЕБЕНКО**

«__» _____2025 р.

**Керівник роботи**
канд. техн. наук,
доцент _____ **Євген ДАВИДЕНКО**

«__» _____2025 р.

**Миколаїв – 2025**

Чорноморський національний університет імені Петра Могили

( повне найменування закладу вищої освіти )

| | |
|---|---|
| Факультет | Комп'ютерних наук |
| Кафедра | Інженерії програмного забезпечення |
| Рівень вищої освіти | Другий (магістерський) |
| Освітній ступінь | Магістр |
| Спеціальність | 121 Інженерія програмного забезпечення |
| Освітня програма | Інженерія програмного забезпечення |

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«___»_____2025 р.

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

**Тебенко Андрія Васильовича**

*(прізвище, ім'я, по батькові здобувача)*

1. Тема кваліфікаційної роботи
Застосунок стоматологічної клініки з функцією прогнозування відвідування пацієнтів

Затверджена наказом ЧНУ ім. Петра Могили від «02» липня 2025 р.
№ 182

2. Строк представлення кваліфікаційної роботи «23» грудня 2025 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;

- визначення архітектури для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування та оптимізації роботи програмного забезпечення;

Перелік графічних матеріалів

Презентація

Завдання до спеціальної частини

5. Консультанти:

| Консультант | Кафедра (організація) | Частина роботи |
|---|---|---|
| | | |
| | | |
| | | |

Дата видачі завдання «___» _____ 20___р.

**КАЛЕНДАРНИЙ ПЛАН**

**виконання кваліфікаційної роботи**

Тема: Застосунок стоматологічної клініки з функцією прогнозування відвідування пацієнтів

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1. | Розробка та затвердження завдання на виконання КМР | 01.09.2025 | 01.09.2025 | виконано |
| 2. | Огляд літератури за темою роботи | 05.09.2025 | 15.09.2025 | виконано |
| 3. | Складання календарного плану КМР | 17.09.2025 | 18.09.2025 | виконано |
| 4. | Аналіз предметної області | 19.09.2025 | 21.09.2025 | виконано |
| 5. | Розробка проєктних рішень | 23.09.2025 | 27.09.2025 | виконано |
| 6. | Моделювання програмного забезпечення | 05.10.2025 | 12.10.2025 | виконано |
| 7. | Кодування, тестування розробленого програмного забезпечення, оптимізація застосунку | 15.10.2025 | 06.11.2025 | виконано |
| 8. | Апробація | 12.11.2025 | 12.11.2025 | виконано |
| 8. | Оформлення КМР та презентації | 13.11.2025 | 18.11.2025 | виконано |
| 9. | Відгук керівника КМР | 18.11.2025 | 18.11.2025 | виконано |
| 10. | Попередній захист | 24.11.2025 | 24.11.2025 | виконано |
| 11. | Завершення оформлення КМР та презентації | 26.11.2025 | 03.12.2025 | виконано |
| 12. | Рецензування | 05.12.2025 | 08.12.2025 | виконано |
| 13. | Захист кваліфікаційної роботи | 23.12.2025 | 23.12.2025 | виконано |

**Здобувач** _____ **Андрій ТЕБЕНКО**

«__» _____2025 р.

**Керівник роботи**

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____2025 р.

# АНОТАЦІЯ

до кваліфікаційної роботи магістра

«Застосунок стоматологічної клініки з функцією прогнозування відвідування пацієнтів»

Здобувач 608д групи: Тебенко Андрій

Керівник: канд. техн. наук, доцент Давиденко Євген.

Дана робота присвячена розробці застосунку для стоматологічної клініки з функцією прогнозування відвідування пацієнтів.

Об’єктом кваліфікаційної роботи є процес управління відвідуваннями клієнтів у стоматологічній клініці.

Предметом роботи є методи збору, обробки та аналізу даних для прогнозування неявок пацієнтів і виявлення потенційно втрачених клієнтів.

Метою кваліфікаційної роботи є розробка застосунку, який поєднує систему керування записами пацієнтів із модулем прогнозування неявок на основі методів машинного навчання. Це дозволить підвищити ефективність роботи клініки, зменшити кількість пропущених прийомів і покращити якість взаємодії з пацієнтами.

Завдання:

1) проаналізувати сучасні підходи та аналоги систем прогнозування неявок у медичній сфері;

2) визначити набір ознак і структуру даних для побудови прогнозної моделі;

3) провести порівняльний аналіз алгоритмів машинного навчання та обрати оптимальний для реалізації;

4) розробити архітектуру вебзастосунку та реляційну базу даних для збереження інформації про пацієнтів, лікарів і візити;

5) реалізувати клієнтську й серверну частини застосунку з безпечним доступом користувачів за ролями;

6) інтегрувати модуль прогнозування відвідувань пацієнтів та протестувати його ефективність.

Кваліфікаційна робота складається зі вступу, 4 розділів, висновків, переліку джерел посилань та додатків.

У вступі обґрунтовано актуальність теми, визначено мету, завдання, об’єкт і предмет дослідження. Розглянуто проблематику неявок пацієнтів, наслідки для медичних закладів та доцільність застосування прогнозної аналітики у стоматології.

У першому розділі проведено аналіз предметної області та існуючих рішень для прогнозування неявок. Визначено основні технологічні підходи, алгоритми машинного навчання та питання інформаційної безпеки при роботі з медичними даними.

У другому розділі розроблено вимоги до системи, описано структуру бази даних та набір ознак, використаних для моделі. Проведено аналіз даних обсягом 3000 записів, побудовано та випробувано декілька моделей машинного навчання. Найкращий результат продемонстрував алгоритм Random Forest, який забезпечив високу точність прогнозування.

У третьому розділі виконано проєктування архітектури вебзастосунку. Створено діаграми прецедентів, станів і класів, а також побудовано реляційну базу даних із чотирнадцяти таблиць, що описують усі сутності системи (пацієнти, лікарі, записи, процедури, прогнози).

У четвертому розділі наведено процес розробки вебзастосунку. Клієнтська частина реалізована з використанням Next.js 15, TypeScript і Tailwind CSS. Серверна частина побудована на Node.js із Prisma ORM та інтеграцією модуля прогнозування. Реалізовано автентифікацію користувачів за допомогою NextAuth.js, сторінку прогнозування відвідувань та оптимізацію продуктивності API, що скоротила час відгуку до 94 %.

У висновках проводиться аналіз виконаних робіт та отриманих результатів.

Ключові слова: *стоматологічна клініка, прогнозування, машинне навчання, вебзастосунок, пацієнт, аналітика, неявка.*

# ABSTRACT

to the Master's degree qualification work

«Dental Clinic Application with a Patient Visit Prediction Function»

Student of 608d group: Andrii Tebenko

Supervisor: Ph.D. in Engineering, Associate Professor Yevhen Davydenko

This work is devoted to the development of a application for a dental clinic with a patient visit prediction function.

The object of the qualification work is the process of managing patient visits in a dental clinic.

The subject of the work is the methods of data collection, processing, and analysis for predicting patient no-shows and identifying potentially lost clients.

The aim of the qualification work is to develop a application that combines a patient appointment management system with a prediction module based on machine learning methods. This approach increases the efficiency of the clinic's operations, reduces the number of missed appointments, and improves patient interaction quality.

Tasks:

1) to analyze modern approaches and analogs of no-show prediction systems in the medical field;

2) to determine the set of features and data structure for building a predictive model;

3) to conduct a comparative analysis of machine learning algorithms and select the optimal one for implementation;

4) to develop the architecture of the web application and a relational database for storing patient, doctor, and visit information;

5) to implement the client and server parts of the application with secure, role-based user access;

6) to integrate the patient visit prediction module and test its effectiveness.

The qualifying work consists of an introduction, 4 chapters, a conclusion, a list of references, and appendices.

The introduction justifies the relevance of the topic, defines the purpose, objectives, object, and subject of the study. It examines the issue of patient no-shows, their impact on medical institutions, and the feasibility of applying predictive analytics in dentistry.

The first chapter analyzes the subject area and existing solutions for predicting no-shows. It defines key technological approaches, machine learning algorithms, and data security aspects in working with medical information.

The second chapter develops the system requirements, describes the database structure and the set of features used for the model. A dataset of 3,000 records was analyzed, several machine learning models were built and tested. The Random Forest algorithm demonstrated the best result, providing high prediction accuracy.

The third chapter presents the design of the web application architecture. Use case, state, and class diagrams were created. A relational database consisting of fourteen tables was built to describe all system entities (patients, doctors, appointments, procedures, predictions).

The fourth chapter describes the process of developing the web application. The client part was implemented using Next.js 15, TypeScript, and Tailwind CSS. The server part was built on Node.js with Prisma ORM and integrated with the prediction module. User authentication was implemented with NextAuth.js, and a visit prediction page was developed. API performance optimization reduced response time by up to 94%.

The conclusions summarize the conducted work and obtained results.

Keywords: *dental clinic, prediction, machine learning, web application, patient, analytics, no-show.*

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних
ЕМК – електронна медична картка
МОЗ – Міністерство охорони здоров’я
ПЗ – програмне забезпечення
ПК – персональний комп’ютер
СКБД – система керування базами даних
ШІ – штучний інтелект

CLI – Command Line Interface
CRM – Customer Relationship Management
ERP – Enterprise Resource Planning
GDRP – General Data Protection Regulation
HTTP – HyperText Transfer Protocol
HTTPS – HyperText Transfer Protocol Secure
IDE – Integrated Development Environment
ISR – Incremental Static Regeneration
KNN – K-Nearest Neighbors
ML – Machine Learning
ORM – Object-Relational Mapping
SEO – Search Engine Optimization
SSL – Secure Sockets Layer
SSG – Static Site Generation
SSR – Server-Side Rendering
UI – User Interface

# ВСТУП

**Актуальність теми**. У сучасних умовах сфери охорони здоров'я, зокрема стоматологічні послуги, переживає період активної цифрової трансформації. Пацієнти очікують зручного та швидкого запису на прийом, прозорої комунікації з клінікою та нагадувань про майбутні візити. Разом з тим, для стоматологічних клінік існує серйозна проблема – неявка пацієнтів на прийом або втрата постійних клієнтів. За даними досліджень, рівень відмови від прийому у медичних закладах може коливатися від 10% до 30%, що призводить до значних фінансових втрат, неефективного використання робочого часу лікарів та зниження доступності медичних послуг для інших пацієнтів. Традиційні методи нагадування, такі як телефонні дзвінки чи SMS-повідомлення, знижують ризик неявки, проте не завжди дають очікуваний результат. Часто вони застосовуються без урахування індивідуальних особливостей пацієнта – його віку, статі, історії відвідувань, типових днів та годин запису тощо. Саме тому важливим завданням стає використання методів аналізу даних та машинного навчання для прогнозування ризику неявки клієнтів [1]. Системи прогнозування дозволяють завчасно виявляти пацієнтів із високою ймовірністю відмови від візиту та застосовувати цільові заходи: індивідуальні нагадування, персоналізовані знижки чи повторні пропозиції запису. Такий підхід не лише збільшує рівень відвідування, але й сприяє утриманню клієнтів, що є стратегічно важливим для розвитку стоматологічної клініки.

Крім того, актуальність теми підсилюється загальними тенденціями у сфері медицини та бізнесу:

1) зростанням конкуренції між приватними клініками, де якість сервісу та оптимізація бізнес-процесів стають ключовими факторами успіху;

2) підвищеним попитом пацієнтів на цифрові сервіси, які забезпечують простоту взаємодії з клінікою;

3) розвитком напрямку прогнозна аналітика (predictive analytics) у медицині, що дозволяє будувати прогнози поведінки пацієнтів та розробляти персоналізовані стратегії взаємодії.

**Об’єкт роботи** – процес управління відвідуваннями клієнтів у стоматологічній клініці.

**Предмет роботи** – методи збору та обробки даних для прогнозування неявок і виявлення потенційно втрачених клієнтів.

**Метою кваліфікаційної роботи** є розробка вебзастосунку для стоматологічної клініки з модулем прогнозування неявок пацієнтів.

**На основі заданої теми було створено наступні завдання:**

1) проаналізувати існуючі аналоги для прогнозування неявок у медицині;
2) визначити набір ознак і структуру даних;
3) побудувати і оцінити моделі;
4) розробити бекенд та фронтенд: аутентифікація ролей (адміністратор, лікар), календар візитів, панель аналітики;
5) протестувати вебзастосунок.

**Очікуваний результат:** вебзастосунок для стоматологічної клініки, що поєднує інструменти управління записами пацієнтів, аутентифікацію користувачів за ролями (адміністратор, лікар), інтерактивний календар візитів та панель аналітики. Система дозволятиме не лише автоматизувати процеси реєстрації та збереження історії відвідувань, а й формуватиме аналітичні звіти щодо динаміки відвідування пацієнтів у клініці. Використання методів аналізу даних забезпечить прогнозування ризиків пропусків, чи відмін візитів, що дасть можливість завчасно реагувати на потенційні проблеми.

**Апробація результатів** КМР відбулась під час XXVIII Всеукраїнської науково-практичної конференції «Могилянські читання – 2025», Миколаїв, 10-14 листопада, 2025 р. (Додаток А).

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Особливості роботи стоматологічних клінік

Стоматологічні клініки займають важливе місце у сфері охорони здоров’я, оскільки забезпечують не лише лікування, а й профілактику захворювань порожнини рота, що безпосередньо впливають на загальний стан організму людини. Особливістю їхньої діяльності є поєднання високотехнологічних медичних процедур із сервісним підходом до пацієнтів [2]. Це зумовлює необхідність не лише професійної роботи лікарів та використання сучасного обладнання, але й ефективної організації управлінських та комунікаційних процесів.

Насамперед варто відзначити, що стоматологічні клініки характеризуються високим рівнем конкуренції на ринку медичних послуг. У багатьох містах діє значна кількість приватних клінік, які пропонують схожі види послуг: терапевтичну стоматологію, ортопедію, ортодонтію, хірургію, імплантологію, тощо. В умовах такої конкуренції визначальними факторами успіху стають якість сервісу, ефективність управління записами, доступність консультацій та рівень комфорту для пацієнта.

Однією з ключових особливостей роботи стоматологічних клінік є висока залежність від відповідальності пацієнтів. На відміну від деяких інших медичних сфер, стоматологічні послуги переважно плануються заздалегідь: пацієнт записується на прийом у зручний для нього час, а клініка резервує кабінет, обладнання та лікаря. Будь-які неявки або запізнення без попередження призводять до простою, втрати фінансових ресурсів і зниження загальної ефективності діяльності закладу. За даними досліджень, у середньому близько 10–30% пацієнтів можуть відмовлятися від візиту чи не приходити взагалі, що створює для клініки суттєві ризики.

Ще однією важливою особливістю є потреба у збереженні довготривалих відносин із пацієнтами. Стоматологічні послуги мають системний характер: вони включають не лише одноразові процедури, а й регулярні профілактичні огляди, курси лікування або ортодонтичні втручання, які тривають місяцями чи навіть роками. Відтак, клієнтська база клініки має довготривалу цінність, і утримання пацієнтів є не менш важливим завданням, ніж залучення нових.

Окрему роль у діяльності стоматологічних клінік відіграє управління ресурсами. Це стосується як кадрового забезпечення (графіки лікарів, розподіл навантаження), так і матеріально-технічної бази (використання стоматологічних установок, витратних матеріалів, діагностичного обладнання). Оптимізація таких процесів напряму впливає на якість послуг і задоволеність пацієнтів.

Сучасні тенденції також підкреслюють зростання значення цифрових сервісів у стоматології. Пацієнти очікують можливості швидкого запису на прийом, отримання нагадувань через електронні канали комунікації, перегляду історії відвідувань та результатів лікування у власному кабінеті. Для клініки це означає необхідність впровадження інформаційних систем, які не лише зберігатимуть дані, але й забезпечуватимуть інструменти для прогнозування поведінки пацієнтів та оптимізації бізнес-процесів.

Окремо слід підкреслити важливість питань безпеки персональних даних та дотримання етичних стандартів. У стоматологічних клініках опрацьовується широкий спектр персональних даних пацієнтів – від базової контактної інформації до відомостей про стан здоров’я, результати обстежень та повну історію лікування. Відповідно, на такі дані поширюються вимоги законодавства щодо медичної таємниці та захисту персональної інформації (зокрема, положення GDPR у країнах ЄС та аналогічні норми в Україні). Недотримання правил конфіденційності може призвести до юридичної відповідальності та втрати довіри пацієнтів. Тому сучасні клініки повинні забезпечувати багаторівневий захист даних: обмеження доступу

для сторонніх осіб, шифрування інформації та регулярне оновлення політик безпеки.

Таким чином, особливості роботи стоматологічних клінік полягають у високій конкурентності ринку, необхідності підтримання довготривалих відносин, важливості ефективного управління ресурсами, впровадженні цифрових рішень та дотриманні вимог щодо безпеки персональних даних. Усі ці фактори формують потребу в сучасних інформаційних технологіях, що дозволяють автоматизувати облік, покращувати комунікацію та прогнозувати ризики неявки пацієнтів, водночас гарантуючи конфіденційність і захист інформації.

## 1.2 Правові та нормативні вимоги у сфері охорони здоров’я

Сфера охорони здоров’я належить до найбільш регульованих галузей суспільного життя, адже безпосередньо пов’язана з правами людини на життя, здоров’я та медичну допомогу. Будь-яка діяльність медичних закладів, у тому числі стоматологічних клінік, має базуватися на чіткій системі правових і нормативних вимог [3]. Ці вимоги спрямовані на захист прав пацієнтів, забезпечення безпеки надання медичних послуг, регламентацію професійної діяльності лікарів, а також упровадження сучасних технологій відповідно до стандартів конфіденційності та безпеки.

В умовах цифровізації та активного впровадження інформаційних технологій правове поле постійно розширюється. Поряд із класичними вимогами щодо ліцензування, сертифікації та ведення документації, усе більшого значення набувають норми, які регулюють захист персональних даних, електронний документообіг, телемедицину та використання інтелектуальних систем у сфері медицини.

Ці вимоги виконують кілька ключових функцій:

1) захист прав пацієнтів: забезпечення їх доступу до медичних послуг, дотримання стандартів лікування, конфіденційності та інформованої згоди на медичні втручання;

2) гарантія безпеки: чітка регламентація умов надання медичної допомоги, контролю за якістю ліків, обладнання та процедур;

3) визначення професійної діяльності медичних працівників: встановлення вимог до освіти, кваліфікації та сертифікації лікарів;

4) регулювання інновацій та цифрових рішень: створення умов для впровадження сучасних інформаційних технологій у відповідності до міжнародних стандартів безпеки.

### 1.2.1 Конституційно-правові основи охорони здоров’я

Конституція України (стаття 49) закріплює право кожної людини на охорону здоров’я, медичну допомогу та медичне страхування [4]. Це базова норма, яка визначає, що держава зобов’язана створювати умови для функціонування системи охорони здоров’я.

Для стоматологічних клінік це означає:

1) гарантію доступності стоматологічних послуг для населення;

2) забезпечення їх належної якості відповідно до сучасних медичних стандартів;

3) можливість поєднання державних та приватних форм надання послуг.

Конституційна норма також підкреслює, що держава фінансує медичні програми, але при цьому дозволяє існування приватних клінік, які повинні діяти виключно в межах правового поля.

### 1.2.2 Ліцензування та регулювання медичної практики

Будь-який медичний заклад в Україні має право здійснювати діяльність лише після отримання ліцензії. Це регулюється Законом України «Про ліцензування видів господарської діяльності».

Ліцензійні умови передбачають:

1) наявність кваліфікованих лікарів із підтвердженими дипломами, сертифікатами та спеціалізаціями;

2) відповідність матеріально-технічної бази: сучасне обладнання, стоматологічні кабінети, засоби стерилізації та дезінфекції;

3) дотримання санітарних правил та вимог інфекційного контролю;

4) ведення медичної документації у встановленому порядку.

Порушення ліцензійних умов може призвести до штрафів, анулювання ліцензії чи навіть закриття клініки.

### 1.2.3 Законодавство про захист персональних даних

Закон України №2297–VI «Про захист персональних даних» набуває особливого значення в умовах цифровізації медицини. Стоматологічні клініки використовують конфіденційні дані [5], тому стоматологічні клініки повинні:

1) отримувати згоду пацієнтів на обробку їхніх даних;

2) забезпечувати збереження карток і результатів обстежень у захищених електронних системах;

3) обмежувати доступ до даних лише уповноваженим співробітникам;

4) розробляти внутрішні політики безпеки.

Ці вимоги особливо актуальні у зв’язку з використанням електронних медичних карток, CRM-систем та інтегрованого діагностичного обладнання.

### 1.2.4 Правові вимоги до електронних медичних систем

Електронні медичні картки (ЕМК):

1) зберігання всієї історії лікування пацієнта;
2) захищений доступ і контроль автентифікації;
3) відповідність стандартам МОЗ і системі eHealth [6].

Електронний документообіг:

1) записи лікарів у цифровій формі;
2) сертифікація програмного забезпечення;
3) інтеграція з державними реєстрами.

Телемедицина:

1) можливість дистанційних консультацій;
2) важлива для стоматології у випадках післяопераційного моніторингу.

### 1.2.5 Стандарти та протоколи лікування

Клініки зобов’язані дотримуватися затверджених протоколів лікування, як у терапевтичній, так і в хірургічній стоматології [7]. Відхилення можливе лише у випадках клінічної необхідності та має бути обґрунтованим у документації.

### 1.2.6 Виклики та перспективи

Одним із найбільших викликів є повільна адаптація законодавства до швидкого розвитку технологій. На сьогодні детального врегулювання потребують:

1) телемедицина;
2) використання штучного інтелекту та великих обсягів даних;
3) питання кібербезпеки.

Таким чином, правові та нормативні вимоги у сфері охорони здоров’я формують основу діяльності медичних закладів і задають вектор розвитку галузі. Для стоматологічних клінік вони є не лише інструментом контролю, а й запорукою

довіри пацієнтів, високої якості медичних послуг та легального впровадження інновацій.

### 1.3 Сучасні підходи до цифровізації медичних закладів

У сучасних умовах розвитку інформаційних технологій цифровізація стала невід'ємною складовою ефективної діяльності медичних закладів [8]. Вона охоплює широкий спектр процесів: від автоматизації внутрішньої документації до впровадження інтелектуальних систем аналізу даних. Зокрема, у стоматологічних клініках цифрові технології відіграють ключову роль у забезпеченні якісного сервісу для пацієнтів, оптимізації роботи персоналу та підвищенні конкурентоспроможності закладу.

Одним із головних напрямів цифровізації є електронний документообіг. Використання електронних медичних карток дозволяє зберігати та швидко обробляти інформацію про пацієнтів – історію хвороби, результати обстежень, плани лікування. Лікар у будь-який момент має доступ до повної динаміки лікування: від первинного огляду до завершення курсу, включно з цифровими знімками, 3D-моделями та лабораторними висновками. Це мінімізує ризик втрати даних, спрощує доступ до актуальної інформації та підвищує оперативність прийняття клінічних рішень.

Особливу роль у цифровізації відіграє інтеграція з діагностичним обладнанням. Сучасні рентген-апарати та 3D-сканери можуть безпосередньо передавати отримані дані в електронну карту пацієнта. Це дозволяє лікарю одночасно переглядати знімки, порівнювати їх із попередніми дослідженнями та швидко приймати клінічні рішення. Така інтеграція скорочує час обробки даних, підвищує точність діагностики та забезпечує повну історію обстежень у цифровому вигляді.

Важливим аспектом є автоматизація адміністративних процесів. Працівники реєстратури та адміністрації використовують сучасні CRM та ERP-системи, які дозволяють вести розклад та завантаженість лікарів, контролювати фінансові потоки та формувати звіти для керівництва. Онлайн-запис на прийом, інтеграція з чат-ботами, SMS нагадування про майбутні візити, суттєво зменшують навантаження на адміністративний персонал та знижують ризик пропусків.

Для лікаря цифрові системи – це зручний інструмент планування. Він може переглядати власний календар прийомів у реальному часі, формувати план лікування для пацієнта з покроковим описом процедур та навіть узгоджувати графік із колегами через інтегровані календарі. У складних випадках дані зберігаються централізовано, що дозволяє кільком спеціалістам одночасно працювати над одним клінічним випадком.

Окремо варто виділити аналітичні та прогностичні можливості. Керівництво клініки отримує доступ до інтерактивних звітів щодо завантаженості лікарів, динаміки відвідувань, фінансових результатів. Системи з елементами машинного навчання можуть виявляти закономірності у поведінці пацієнтів, прогнозувати ризики відмін або перенесення візитів та надавати рекомендації для оптимізації графіків. Це дозволяє ефективніше управляти ресурсами, уникати перевантажень і втрат часу.

Отже, цифровізація у стоматологічних клініках – це не лише автоматизація рутинних процесів, а й створення комплексної інфраструктури, де пацієнти, лікарі та адміністрація працюють у єдиній інформаційній екосистемі. Це підвищує якість медичних послуг, полегшує управління закладом та забезпечує його стратегічний розвиток у конкурентному середовищі.

## 1.4 Тенденції використання штучного інтелекту та аналітики у медицині

На сьогоднішній день штучний інтелект (ШІ) та сучасні аналітичні технології стають ключовими факторами у розвитку медицини. Вони дозволяють перейти від традиційної реактивної моделі охорони здоров’я [9], коли лікар втручався вже після виникнення проблеми, до прогностичної та персоналізованої медицини, де захворювання можна передбачити, а лікування адаптувати під кожного пацієнта [10]. У стоматології, яка поєднує клінічні процедури, діагностику, візуалізацію та організацію роботи з пацієнтами, ці тенденції мають особливо велике значення.

Основні напрями впровадження ШІ та аналітики:

1) Прогностична аналітика та прогнозування поведінки пацієнтів:

– використання алгоритмів машинного навчання для передбачення неявок на прийоми [11];

– аналіз факторів ризику: час запису, день тижня, вік і стать пацієнта, попередній досвід відвідувань;

– можливість автоматичного надсилання нагадувань у зручний час, адаптація комунікації під індивідуальні звички пацієнта.

2) Оптимізація організації роботи клініки:

– ШІ допомагає створювати динамічні графіки прийомів, що враховують складність процедур, тривалість операцій, а також індивідуальний темп роботи кожного лікаря.

3) Комп’ютерний зір у стоматології:

– стоматологічні клініки все активніше використовують цифрові рентген-знімки та 3D-сканери;

– алгоритми ШІ можуть автоматично: виявляти карієс на ранніх стадіях; оцінювати стан кісткової тканини перед імплантацією; планувати

ортодонтичне лікування; аналізувати динаміку захворювань, що особливо актуально для країн, що розвиваються [12].

4) Персоналізоване лікування та аналітика ризиків:

– ШІ дозволяє створювати індивідуальні плани лікування на основі медичної історії пацієнта;

– алгоритми враховують: генетичні фактори, звички, рівень стресу, особливості харчування, дотримання попередніх рекомендацій;

– персоналізована профілактика: система може нагадувати пацієнтові про гігієнічні процедури або пропонувати ранні профілактичні візити.

5) Обробка медичних даних:

– системи здатні аналізувати електронні медичні записи, автоматично структурувати діагнози та процедури;

– лікар може отримати готові звіти, а адміністрація зрозумілі аналітичні показники. Це зменшує рутинне навантаження на персонал і дозволяє зосередитися на роботі з пацієнтами.

6) Аналітика для керівництва клініки:

– панелі моніторингу показують динаміку доходів, кількість відвідувань, ефективність лікарів.

7) Дистанційний моніторинг та телемедицина:

– пацієнти можуть надсилати фото чи відео після операції, а система ШІ автоматично оцінює ризики ускладнень;

– лікар отримує повідомлення лише у випадках відхилень від норми, що значно знижує навантаження. У стоматології це важливо для контролю процесу загоєння після імплантації чи хірургічних втручань.

8) Економічний ефект від використання ШІ:

– автоматизація процесів знижує адміністративні витрати;

– персоналізована комунікація сприяє збереженню пацієнтів і формуванню довготривалих відносин із клієнтами.

## 1.5 Огляд застосунків-аналогів

Огляд застосунків-аналогів є ключовим етапом у процесі розробки програмного забезпечення для стоматологічної клініки. Він дозволяє визначити наявність рішень, що вже існують на ринку, та можуть частково або повністю задовольнити потреби користувачів, а також виявити можливості для створення конкурентних переваг у новому продукті. Аналіз аналогів допомагає окреслити критерії функціональності (ведення електронних карток пацієнтів, календар відвідувань, автоматичні нагадування, аналітика відвідуваності), зручності інтерфейсу, швидкодії, масштабованості та інтеграції з іншими медичними системами. У рамках дослідження було розглянуто такі програмні рішення, як: PracticeAnalytics, Dentalogic та Genware. Кожен із них має свої переваги у сфері прогнозування та аналізу відвідувань пацієнтів, проте разом із тим виявлені й недоліки, які необхідно врахувати під час створення власного застосунку. Зокрема, це стосується адаптивності інтерфейсу, можливостю інтеграції з локальними системами управління клінікою.

**Назва:** PracticeAnalytics (рис. 1.1).

**Архітектура:** client-server.

**Функції:**

1) аналітика відвідуваності пацієнтів;
2) прогнозування записів та неявок;
3) інтеграція з системами управління клінікою;
4) фінансові та операційні звіти;
5) візуалізація даних у вигляді діаграм і графіків.

**Переваги:**

1) глибока аналітика з прогнозними моделями;
2) розширені можливості інтеграції;
3) підтримка різних форматів звітів.

**Недоліки:**

1) висока вартість підписки;
2) складність у налаштування для невеликих клінік;
3) наявність тільки англомовного інтерфейсу.

https://practiceanalytics.com/


Рисунок 1.1 – Вебзастосунок Practice Analytics

**Назва:** Dentalogic (рис. 1.2) [13].

**Архітектура:** client-server, локальна версія.

**Функції:**

1) ведення електронних карток пацієнтів;
2) календар відвідування та управління розкладом;
3) управління лікувальними планами;
4) фінансовий облік та звітність.

**Переваги:**

1) повноцінна система для управління клінікою;
2) можливість мати локальну версію застосунку;
3) автоматичні нагадування для пацієнтів;
4) модулі для бухгалтерського обліку;
5) підтримка декількох мов інтерфейсу.

**Недоліки:**

1) обмежена аналітика щодо прогнозування неявок (система більше орієнтована на облік, ніж на прогнозування);
2) складність інтеграції з деякими зовнішніми системами (наприклад, сторонні CRM або медичні сервіси);
3) обмежені можливості гнучкого налаштування під індивідуальні потреби клініки;

https://www.dentalogic.com/

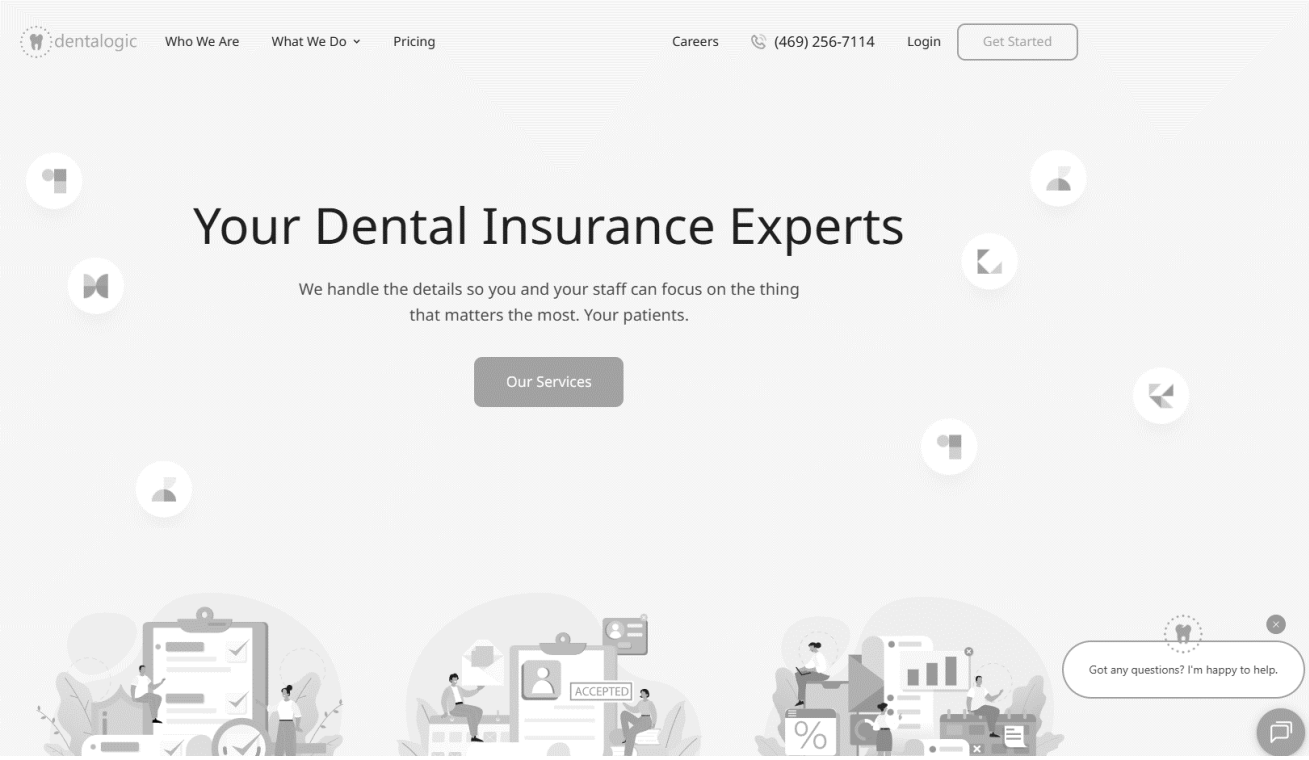



Рисунок 1.2 – Вебзастосунок Dentalogic

**Назва:** Genware (рис. 1.3).

**Архітектура:** client-server.

**Функції:**

1) прогнозування та аналіз даних на основі BI-технологій;
2) інтеграція з медичними інформаційними системами;
3) управління великими масивами даних;
4) моделі прогнозування відвідуваності пацієнтів.

**Переваги:**

1) потужні інструменти Business Intelligence;
2) можливість обробки великих даних;
3) налаштування звітів під потреби клініки;
4) масштабованість рішення;

**Недоліки:**

1) орієнтованість на середні та великі клініки;
2) складність у впровадженні та налаштуванні;
3) потребує кваліфікованих спеціалістів для адміністрування.

https://www.genware.com/

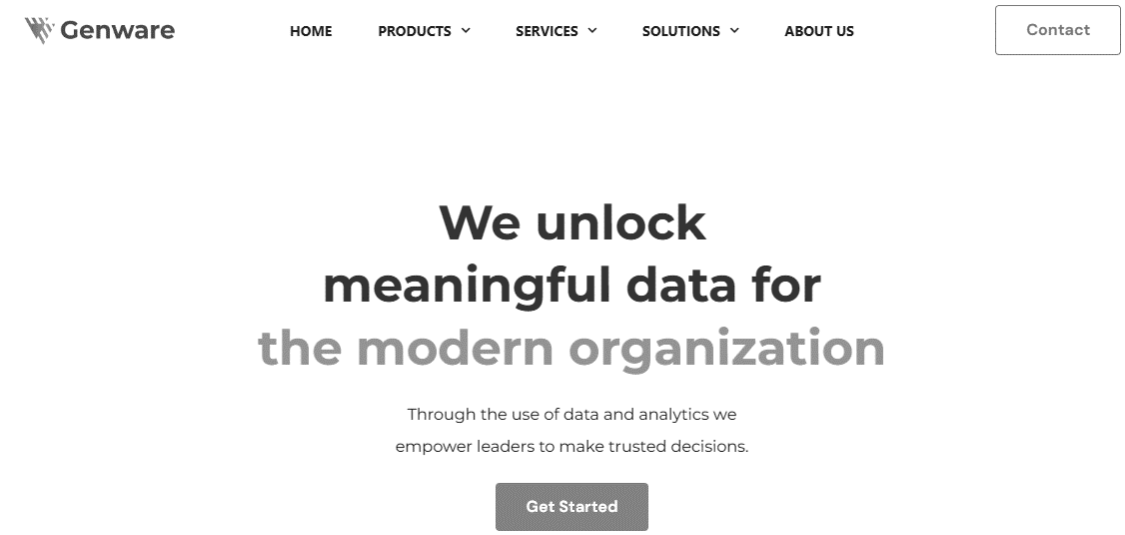



Рисунок 1.3 – Вебзастосунок Genware

## Висновки до розділу 1

У першому розділі розробка вебзастосунку стоматологічної клініки з функцією прогнозування відвідування пацієнтів розглядається як актуальне рішення у сфері цифровізації медичних послуг. У сучасних умовах ефективна робота медичного закладу залежить не лише від професіоналізму персоналу, але й від здатності оптимізувати організаційні процеси. Однією з найбільших проблем клінік є неявка пацієнтів на заплановані прийоми, що призводить до втрати часу лікаря та нераціонального використання ресурсів. Використання інтелектуальних алгоритмів прогнозування дозволяє завчасно виявити ризик відміни візиту та вчасно відреагувати та повідомити пацієнта, перенести прийом чи запропонувати вільний час іншому клієнту.

Вебзастосунок реалізує клієнт-серверну архітектуру, що складається з клієнтської частини для взаємодії користувачів, серверної частини для обробки запитів та бази даних для збереження інформації про пацієнтів, записи та історію відвідувань. Завдяки адаптивному інтерфейсу, даний проєкт є зручним у використанні як для адміністратора, так і для лікаря, забезпечуючи доступ з будь-якого пристрою з підключенням до Інтернету. Окрема увага приділяється безпеці даних: паролі зберігаються у хешованому вигляді, а доступ до адміністративних функцій обмежено лише уповноваженим особам.

Використання мови програмування TypeScript та фреймворку Next.js забезпечує гнучкість та можливість подальшого масштабування системи. База даних PostgreSQL гарантує швидке опрацювання великих обсягів інформації, що особливо важливо при роботі з розширеною клієнтською базою. Важливим аспектом є реалізація нагадувань для пацієнтів про майбутні прийоми, що збільшує рівень відвідуваності.

Впровадження цифрових рішень у медицині має відбуватися з урахуванням правових і нормативних вимог. Стоматологічні клініки зобов’язані дотримуватися

ліцензійних умов, стандартів лікування, протоколів МОЗ та законодавства про захист персональних даних [14]. Саме нормативна база визначає рамки роботи програмного забезпечення – від ведення електронної документації до гарантій конфіденційності пацієнтів.

Таким чином, вебзастосунок стоматологічної клініки з функцією прогнозування відвідування пацієнтів є ефективним інструментом для підвищення продуктивності роботи медичного закладу.

# 2 ПРОГНОЗУВАННЯ ВІДВІДУВАННЯ ПАЦІЄНТІВ

## 2.1 Специфікація вимог до програмного забезпечення

### 1. ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

#### 1.1 Призначення системи (застосунку), для якого розробляється ПЗ

Розроблюване програмне забезпечення призначене для управління відвідуванням пацієнтів у стоматологічній клініці та прогнозування можливих неявок. Система дозволяє адміністраторам і лікарям вести облік записів, планувати робочий час, отримувати аналітику щодо відвідувань пацієнтів, а також використовувати алгоритми прогнозування для зменшення кількості пропущених прийомів.

#### 1.2 Межі проєкту ПЗ

Основна увага зосереджена на створенні вебзастосунку з доступом до особистого кабінету адміністратора та лікаря. Проєкт не охоплює розробку апаратного забезпечення чи інтеграцію з медичними пристроями. Прогнозування здійснюється виключно на основі даних відвідувань, без використання зовнішніх медичних баз.

### 2. ЗАГАЛЬНИЙ ОПИС

#### 2.1 Сфера застосування

– Стоматологічні клініки малого та середнього масштабу.

– Медичні центри з відділенням стоматології.

#### 2.2 Характеристика користувачів

– Лікарі: ведення розкладу прийомів, доступ до історії відвідувань.

– Адміністратори: управління календарем клініки, формування звітів, контроль розкладу лікарів.

#### 2.3 Загальна структура і склад системи

– Клієнтська частина (Front-end) – веб-інтерфейс для пацієнтів, лікарів та адміністраторів.

– Серверна частина (Back-end) – модуль обробки запитів, прогнозування та управління даними.

– База даних – зберігання даних про пацієнтів, відвідування, розклад.

– Модуль прогнозування – використання алгоритмів машинного навчання для визначення ймовірності неявки.

### 2.4 Загальні обмеження

– Системні вимоги до апаратного забезпечення для коректної роботи.

– Необхідність постійного доступу до Інтернету.

– Можливі обмеження на точність прогнозування, що залежать від обсягу історичних даних.

## 3. ФУНКЦІЇ СИСТЕМИ

### 3.1 Реєстрація та авторизація користувачів

**Опис функції**

Функція дозволяє створювати облікові записи для персоналу.

**Вхідна і вихідна інформація**

Вхідна інформація – ім’я, email, телефон, пароль.

Вихідна інформація – підтвердження реєстрації, доступ до кабінету.

### 3.2 Управління записами пацієнтів

**Опис функції**

Функція дозволяє адміністраторам керувати розкладом.

**Вхідна і вихідна інформація**

Вхідна інформація – дата, час, лікар.

Вихідна інформація – підтвердження запису, оновлення календаря.

### 3.3 Прогнозування відвідувань

**Опис функції**

Функція дозволяє прогнозувати ймовірність неявки.

**Вхідна і вихідна інформація**

Вхідна інформація – дані про попередні відвідування, поведінка користувача (зміни, скасування).

Вихідна інформація – індикатор ризику неявки (низький, середній, високий).

### 3.4 Аналітика та звітність

**Опис функції**

Функція дозволяє формувати статистику відвідувань або неявок.

**Вхідна і вихідна інформація**

Вхідна інформація – дані про прийоми, відвідуваність.

Вихідна інформація – графіки або звіти у форматі Excel.

### 3.5 Система сповіщень

**Опис функції**

Функція дозволяє нагадувати пацієнтам про візити.

**Вхідна і вихідна інформація**

Вхідна інформація – дані про дату та час візиту.

Вихідна інформація –email повідомлення користувачу.

## 4. ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Джерела і зміст вхідної інформації

У цьому вебзастосунку вхідні дані отримуються від адміністратора та лікарів клініки, які вносять інформацію про розклад прийомів, записи пацієнтів, історію відвідувань, а також результати прогнозів системи. Додатково система враховує історичні дані про неявки пацієнтів.

### 4.2 Нормативно-довідникова інформація (класифікація, довідники тощо)

– Довідники лікарів та їх спеціалізацій;

– Класифікатори медичних послуг клініки;

– Базові дані пацієнтів (ПІБ, контакти, історія відвідувань).

### 4.3 Вимоги до способів організації, збереження та ведення інформації

Обмін даними відбувається через обробку запитів до серверної частини системи з використанням реляційної бази даних PostgreSQL.

## 5. ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для розробки програмного забезпечення немає значних технічних обмежень. Система повинна коректно працювати на стандартних ПК, ноутбуках, планшетах та смартфонах з доступом до Інтернету.

## 6. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 6.1 Архітектура програмної системи

Система складається з:

– клієнтської частини (веб-інтерфейс);

– серверної частини (обробка даних);

– бази даних (збереження інформації про користувачів, записи, історію відвідування, статистику прогнозів).

### 6.2 Системне програмне забезпечення

Використання операційної системи Windows 7,10,11 чи Linux.

### 6.3 Програмне забезпечення ведення інформаційної бази

Використано БД PostgreSQL.

### 6.4 Мові і технологія розробки ПЗ

Мова програмування: Typescript.

Фреймворк: Next.js.

## 7. ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

### 7.1 Інтерфейс користувача

Інтерфейс має бути інтуїтивно зрозумілим, адаптивним, із підтримкою мовної локалізації (українська, англійська).

### 7.2 Апаратний інтерфейс

Програмний застосунок має бути оптимізованим для роботи як на стаціонарних ПК, так і на планшетних або мобільних пристроях.

### 7.3 Програмний інтерфейс

Доступ до API повинен бути забезпечений через RESTful сервіси.

### 7.4 Комунікаційний протокол

Система повинна використовувати HTTP/HTTPS для взаємодії між компонентами.

## 8. ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 8.1 Доступність

Перегляд розкладу та прогнозів відвідувань доступний лікарям і адміністраторам після авторизації.

### 8.2 Супроводжуваність

Забезпечення оновлення функціоналу (модулів прогнозування, календаря), усунення технічних помилок.

### 8.3 Переносимість

Застосунок доступний для різних пристроїв та операційних систем без втрати функціоналу.

### 8.4 Продуктивність

Продуктивність системи залежить від якості Інтернет-з’єднання та обсягу оброблюваних даних.

### 8.5 Надійність

Програмне забезпечення повинно мати механізми для обробки помилок і відновлення після збоїв.

### 8.6 Безпека

Лише адміністратор має доступ до редагування. Шифрування персональних даних та захист відповідно до GDPR.

## 2.2 Опис набору даних

Для складання датасету була використана певна кількість інформації про відвідування стоматологічної клініки, включаючи демографічні характеристики пацієнтів, історію їхніх візитів, типи процедур та організаційні чинники.

Цільова змінна. Цільовою змінною є показник відвідуваності пацієнта (no_show), що визначає факт явки на прийом або його неявки.

Опис ознак. Датасет містить 11 характеристик пацієнтів, лікарів та умов запису, а також 3000 записів із відповідними даними.

Стовпчики:

– patient_gender – стать пацієнта (чоловіча/жіноча/невідома);

– patient_age – вік пацієнта у роках;

– attendance_rate_hist – історична відвідуваність пацієнта (частка відвіданих прийомів від усіх записів);

– procedure_type – тип стоматологічної процедури (огляд, гігієна, терапія, хірургія, імплантація, ортодонтія, дитяча);

– procedure_complexity – складність процедури (1 – низька, 2 – середня, 3 – висока);

– day_of_week – день тижня, на який призначено прийом;

– time_of_day_block – час доби (ранок, день, вечір);

– is_weekend – ознака вихідного дня (так/ні);

– reminder_sent – факт надсилання нагадування пацієнту (так/ні);

– doctor_no_show_rate_90d – частка неявок у прийомах цього лікаря за останні 90 днів;

– no_show – цільова змінна (0 – пацієнт відвідав або скасував завчасно; 1 – пацієнт не з’явився або скасував пізно).

## 2.3 Первинний аналіз даних

Продемонструємо первинний аналіз даних (рис. 2.1). На початковому етапі було виконано завантаження датасету та перегляд перших рядків таблиці з метою ознайомлення зі структурою даних [15]. Датасет містить інформацію про пацієнтів стоматологічної клініки, їхні демографічні характеристики, типи процедур, часові параметри прийому та факт відвідування чи неявки. Такий підхід дозволяє отримати загальне уявлення про зміст набору даних, виявити основні змінні та оцінити їхню коректність. Зокрема, ми бачимо, що серед полів присутні як числові показники (вік, історична відвідуваність, частка неявок у лікаря), так і категоріальні (стать, тип процедури, день тижня). Це дає змогу надалі застосувати як статистичні методи аналізу, так і алгоритми машинного навчання для прогнозування відвідуваності.

```
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline

from matplotlib import pyplot as plt
plt.rcParams["figure.figsize"] = (15, 15)

import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing, metrics

data_patients = pd.read_csv("./patient_attendance_dataset.csv", sep=",")
data_patients.head()
```

| | patient_gender | patient_age | attendance_rate_hist | procedure_type | procedure_complexity | day_of_week | time_of_day_block | is_weekend | reminder_sent | doctor_no_show_rate_90d | no_show |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | male | 70 | 0.613 | surgery | 3 | Thu | evening | 0 | 0 | 0.088913 | 0 |
| 1 | female | 83 | 0.883 | hygiene | 1 | Fri | evening | 0 | 1 | 0.165557 | 0 |
| 2 | female | 21 | 0.753 | checkup | 1 | Sat | evening | 1 | 1 | 0.142497 | 1 |
| 3 | female | 84 | 1.000 | hygiene | 1 | Tue | evening | 0 | 1 | 0.109103 | 0 |
| 4 | female | 63 | 0.760 | surgery | 3 | Thu | evening | 0 | 1 | 0.114958 | 0 |

Рисунок 2.1 – Первинні дані з датасету

Імпортований датафрейм без змін даних містить 3000 записів та 11 ознак. Отримаємо детальну інформацію про ознаки датафрейму (рис. 2.2).

```
data_patients.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   patient_gender          3000 non-null   object
 1   patient_age             3000 non-null   int64
 2   attendance_rate_hist    3000 non-null   float64
 3   procedure_type          3000 non-null   object
 4   procedure_complexity    3000 non-null   int64
 5   day_of_week             3000 non-null   object
 6   time_of_day_block       3000 non-null   object
 7   is_weekend              3000 non-null   int64
 8   reminder_sent           3000 non-null   int64
 9   doctor_no_show_rate_90d 3000 non-null   float64
 10  no_show                 3000 non-null   int64
dtypes: float64(2), int64(5), object(4)
memory usage: 257.9+ KB
```

Рисунок 2.2 – Детальна інформація про ознаки датафрейму

В датафреймі дані представлені типом «float», «object» та «int64». Пропусків даних немає, оскільки кількість non-null елементів дорівнює кількості всіх елементів даних. Переглянемо поточну статистику даних за допомогою методу describe (рис. 2.3).

```
data_patients.describe()
```

| | patient_age | attendance_rate_hist | procedure_complexity | is_weekend | reminder_sent | doctor_no_show_rate_90d | no_show |
|---|---|---|---|---|---|---|---|
| count | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 |
| mean | 45.572667 | 0.816495 | 1.839333 | 0.283000 | 0.712667 | 0.156658 | 0.250000 |
| std | 22.223351 | 0.108042 | 0.836512 | 0.450532 | 0.452594 | 0.064041 | 0.433085 |
| min | 6.000000 | 0.498000 | 1.000000 | 0.000000 | 0.000000 | 0.054003 | 0.000000 |
| 25% | 27.000000 | 0.741000 | 1.000000 | 0.000000 | 0.000000 | 0.109103 | 0.000000 |
| 50% | 46.000000 | 0.823000 | 2.000000 | 0.000000 | 1.000000 | 0.148052 | 0.000000 |
| 75% | 65.000000 | 0.889000 | 3.000000 | 1.000000 | 1.000000 | 0.169080 | 0.250000 |
| max | 86.000000 | 1.000000 | 3.000000 | 1.000000 | 1.000000 | 0.287980 | 1.000000 |

Рисунок 2.3 – Детальна інформація про ознаки датафрейму

Оскільки у датасеті присутні числові змінні, метод describe() дозволив отримати основні статистичні характеристики цих ознак. З аналізу видно, що середній вік пацієнтів становить близько 45 років, при цьому мінімальний – 6 років, а максимальний – 86 років, що свідчить про широкий віковий діапазон вибірки. Показник історичної відвідуваності (attendance_rate_hist) у середньому дорівнює 0.81, тобто більшість пацієнтів зазвичай відвідують свої прийоми, однак існує певна частка тих, хто схильний до пропусків.

Складність процедур варіюється від 1 до 3**,** при цьому медіанне значення дорівнює 2, що вказує на домінування процедур середнього рівня складності. Частка записів на вихідні дні становить близько 28%, тоді як у 71% випадків пацієнтам надсилаються нагадування.

Середній показник doctor_no_show_rate_90d дорівнює 0.15, що може відображати вплив певних лікарів на рівень дисципліни пацієнтів. Цільова змінна no_show має середнє значення 0.25, що відповідає приблизно 25% неявок від загальної кількості записів.

Оскільки в датасеті є дані з типом «int64», то застосування методу describe без додаткових параметрів надає статистику лише для числових значень, які містяться у датасеті.

Розглянемо окремо характеристику статі пацієнтів, а також, наприклад, статистику за віковими групами (рис. 2.4 – рис. 2.5). Для цього доцільно використати методи групування та візуалізації даних.

```
data_patients["patient_gender"].value_counts()
```

```
patient_gender
female     1689
male       1258
unknown      53
Name: count, dtype: int64
```

Рисунок 2.4 – Характеристика статі пацієнтів

```
bins = [0, 18, 35, 60, 100]
labels = ["<18", "18-35", "36-60", "60+"]
data_patients["age_group"] = pd.cut(data_patients["patient_age"], bins=bins, labels=labels)

data_patients["age_group"].value_counts()

data_patients.groupby("age_group")["no_show"].mean()
```

```
age_group
<18      0.260465
18-35    0.253463
36-60    0.246604
60+      0.245791
Name: no_show, dtype: float64
```

Рисунок 2.5 – Статистика за віковими групами

Аналіз розподілу за статтю показав, що у вибірці переважають пацієнти жіночої статі (1689 записів), тоді як чоловіків значно менше (1258), а кількість записів із невизначеною статтю є незначною (53). Це може свідчити про те, що жінки частіше звертаються за стоматологічними послугами.

Аналіз за віковими групами показав, що рівень неявок у всіх категоріях є досить близьким і коливається в межах від ~24% до ~26%. Найвищий показник спостерігається серед пацієнтів віком до 18 років (~26%), тоді як у групах 18–35 років та 36–60 років рівень неявок дещо нижчий (~25%). Найменше значення зафіксовано серед пацієнтів віком 60+ (~24,6%). Загалом, можна зробити висновок, що в даній вибірці віковий фактор не має суттєвого впливу на ймовірність неявки, адже різниця між групами є незначною.

## 2.4 Первинний візуальний аналіз даних

За допомогою функції створення pairplot в бібліотеці Seaborn створимо графік взаємних відношень числових ознак датасету (рис. 2.6) [16].

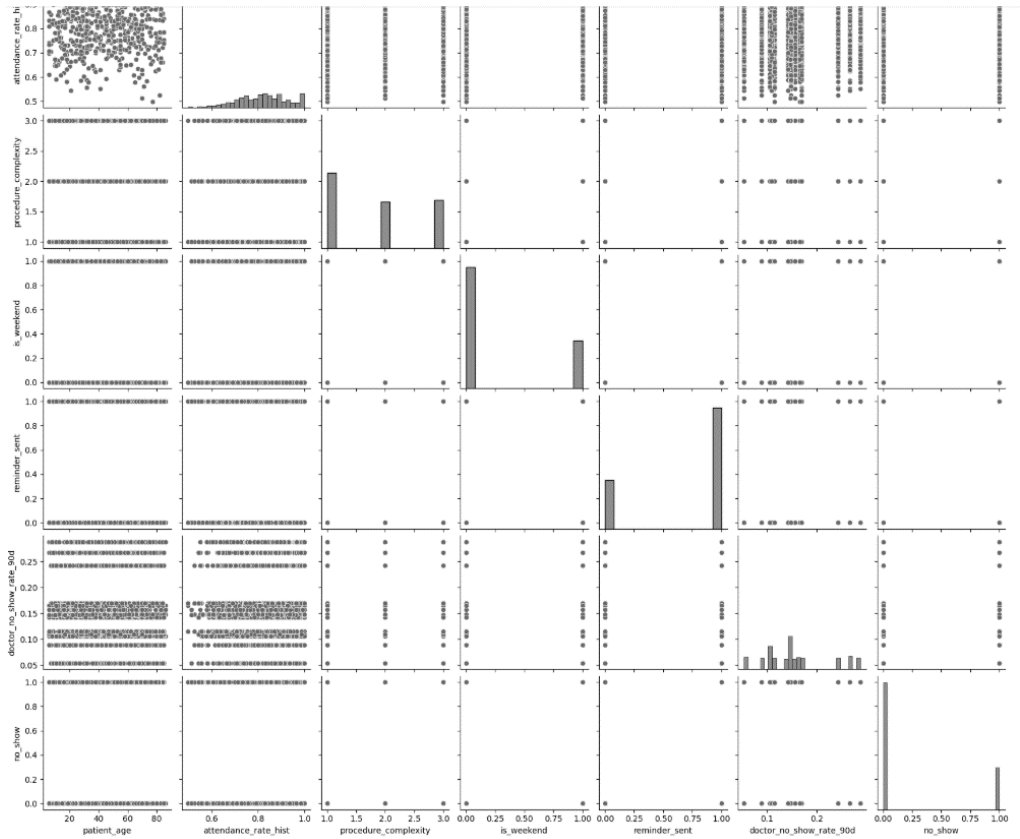



Рисунок 2.6 – Графік взаємних положень ознак датасету

Згідно з побудованим графіком взаємних положень можна зробити висновок, що числові ознаки датасету мають різний ступінь варіативності та розподілу. Вік пацієнтів (patient_age) розподілений у широкому діапазоні – від дітей до літніх пацієнтів, тоді як такі показники, як is_weekend, reminder_sent та no_show, мають двійковий формат і відображаються у вигляді точкових скупчень. Ознака attendance_rate_hist демонструє високу концентрацію значень поблизу 0.8–0.9, що свідчить про загальну стабільність у дотриманні записів більшості пацієнтів. Особливу увагу заслуговує показник doctor_no_show_rate_90d, який варіюється у відносно вузькому діапазоні та може бути корисним для виявлення залежності між неявками та практикою конкретного лікаря. Ознака procedure_complexity представлена трьома рівнями (1–3), і на графіку добре видно дискретність цього параметра.

Узагальнюючи, можна відзначити, що більшість змінних є або дискретними, або бінарними, тому візуалізація у вигляді матриці парних залежностей допомагає лише загально оцінити структуру даних.

## 2.5 Обробка даних та подальший аналіз

Аномальних значень у датасеті не виявлено. Це означає, що всі змінні знаходяться в межах допустимих діапазонів і не мають значних відхилень, які могли б суттєво вплинути на якість подальшого аналізу або навчання моделей. Проте, якщо спробувати побудувати графік залежності числових і нечислових змінних, то можна побачити, що функція pairplot відображає лише числові значення на осях координат і не будує графіки для текстових полів, таких як стать пацієнта чи тип процедури. Це пов’язано з тим, що бібліотека Seaborn працює із числовими даними, тоді як більшість класифікаторів та алгоритмів машинного навчання також не здатні обробляти текстові змінні без попереднього перетворення. Для розв’язання цієї проблеми було виконано кодування всіх категоріальних (нечислових) ознак у числові значення (рис. 2.7).

```
df_enc = data_patients.copy()

gender_map = {"male": 0, "female": 1, "unknown": 2}
tod_map = {"morning": 0, "day": 1, "evening": 2}
dow_order = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
dow_map = {d:i for i, d in enumerate(dow_order)}
proc_map = {p:i for i, p in enumerate(sorted(data_patients["procedure_type"].unique()))}

df_enc["patient_gender_num"]      = df_enc["patient_gender"].map(gender_map)
df_enc["time_of_day_block_num"]= df_enc["time_of_day_block"].map(tod_map)
df_enc["day_of_week_num"]      = df_enc["day_of_week"].map(dow_map)
df_enc["procedure_type_num"]   = df_enc["procedure_type"].map(proc_map)

display(df_enc.head(10)[[
    "patient_gender","patient_gender_num",
    "time_of_day_block","time_of_day_block_num",
    "day_of_week","day_of_week_num",
    "procedure_type","procedure_type_num"
]])
```

| | patient_gender | patient_gender_num | time_of_day_block | time_of_day_block_num | day_of_week | day_of_week_num | procedure_type | procedure_type_num |
|---|---|---|---|---|---|---|---|---|
| 0 | male | 0 | evening | 2 | Thu | 3 | surgery | 5 |
| 1 | female | 1 | evening | 2 | Fri | 4 | hygiene | 1 |
| 2 | female | 1 | evening | 2 | Sat | 5 | checkup | 0 |
| 3 | female | 1 | evening | 2 | Tue | 1 | hygiene | 1 |
| 4 | female | 1 | evening | 2 | Thu | 3 | surgery | 5 |
| 5 | unknown | 2 | evening | 2 | Thu | 3 | hygiene | 1 |
| 6 | male | 0 | evening | 2 | Wed | 2 | therapy | 6 |
| 7 | female | 1 | evening | 2 | Fri | 4 | implant | 2 |
| 8 | male | 0 | evening | 2 | Sat | 5 | therapy | 6 |
| 9 | female | 1 | evening | 2 | Sun | 6 | pediatric | 4 |

Рисунок 2.7 – Перетворення нечислових ознак у числові

Після виконаного перетворення з’явилася можливість детальніше аналізувати категоріальні змінні. Наприклад, було розглянуто статистику за статтю пацієнтів, де видно, що жінки становлять більшість вибірки, тоді як чоловіків менше, а кількість записів із невизначеною статтю є незначною (рис. 2.8). Такий розподіл може бути зумовлений особливостями вибірки або випадковими чинниками під час збору даних. Важливо зазначити, що наявність дисбалансу за статтю необхідно враховувати при побудові прогнозних моделей, адже він може впливати на точність результатів. Таким чином, перетворення даних дозволило отримати не лише кількісну статистику, а й забезпечило можливість більш об’єктивного аналізу змінних, що у подальшому підвищує якість прогнозування відвідуваності пацієнтів.

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2, figsize=(12, 4))

data_patients["patient_gender"].value_counts().reindex(["female","male","unknown"]).plot(
    kind="bar", ax=ax[0]
)
ax[0].set_title("Кількість записів за статтю")
ax[0].set_xlabel("Стать")
ax[0].set_ylabel("Кількість")

sns.barplot(
    x="patient_gender", y="no_show",
    data=data_patients, order=["female","male","unknown"],
    estimator="mean", ax=ax[1]
)
ax[1].set_title("Частка неявок за статтю")
ax[1].set_xlabel("Стать")
ax[1].set_ylabel("Середнє значення no_show")

plt.tight_layout()
plt.show()
```

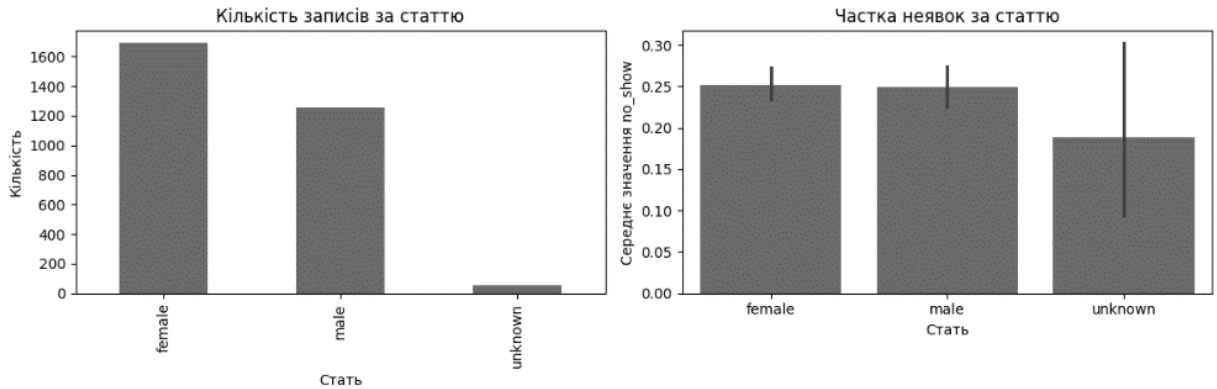



Рисунок 2.8 – Статистика даних за віковими групами

Аналіз розподілу пацієнтів за статтю показав, що найбільшу частку становлять жінки, тоді як чоловіків менше, а кількість записів із невизначеною статтю є незначною (рис. 2.9). Це свідчить про певний дисбаланс у вибірці, який важливо враховувати під час побудови моделей прогнозування. Додатково було проаналізовано рівень неявок у різних гендерних групах. Результати демонструють, що середні показники відсутності на прийомах є подібними серед пацієнтів чоловічої та жіночої статі, тоді як у групі з невизначеним значенням спостерігається вища варіативність через малу кількість записів. Загалом це свідчить про відносно рівномірний розподіл неявок за статтю, без суттєвих відмінностей між основними групами.

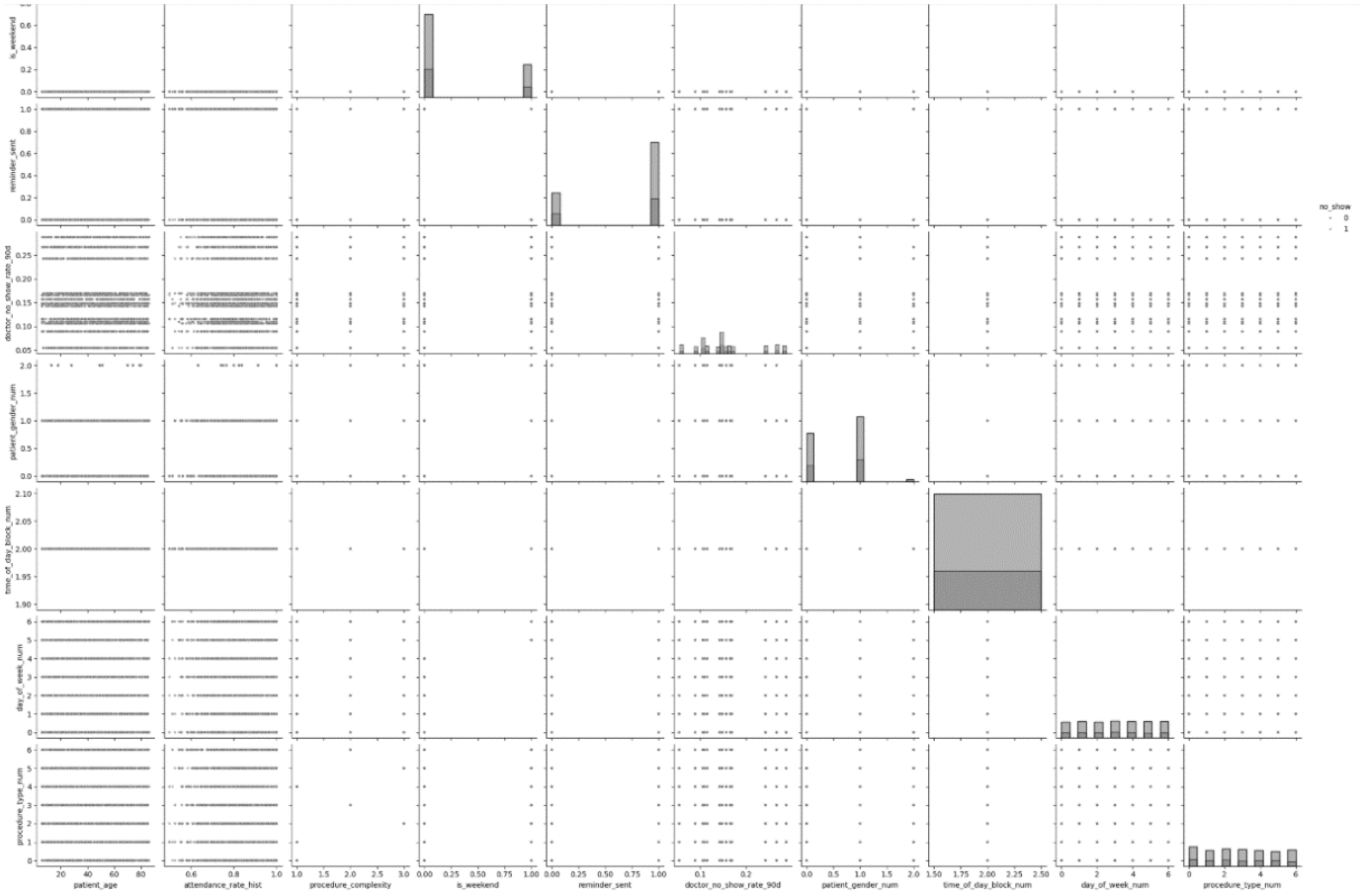



Рисунок 2.9 – Графік після перетворення даних (pairplot числових ознак)

Було проведено аналіз розподілу віку пацієнтів залежно від факту відвідування або неявки (рис. 2.10). Графік демонструє, що у вибірці присутні пацієнти різного віку – від дітей до літніх осіб. При цьому медіанні значення віку для обох груп (тих, хто відвідав/скасував завчасно, і тих, хто не з’явився/скасував пізно) є дуже близькими. Це свідчить про відсутність суттєвого впливу віку на ймовірність неявки, адже як молодші, так і старші пацієнти мають схожу поведінкову динаміку. Незначні відмінності у крайніх значеннях можуть пояснюватися індивідуальними випадками, але загальна тенденція показує, що фактор віку не є визначальним при прогнозуванні пропусків візитів.

```
fig, ax = plt.subplots(figsize=(7, 4))

sns.boxplot(x="no_show", y="patient_age", data=data_patients, ax=ax)
ax.set_xticklabels(["Відвідав/скасував завчасно (0)", "Не з’явився/скасував пізно (1)"])
ax.set_xlabel("no_show")
ax.set_ylabel("Вік пацієнта")
ax.set_title("Розподіл віку за фактом відвідування")

ax.axhline(data_patients["patient_age"].mean(), linestyle="--")

plt.tight_layout()
plt.show()
```

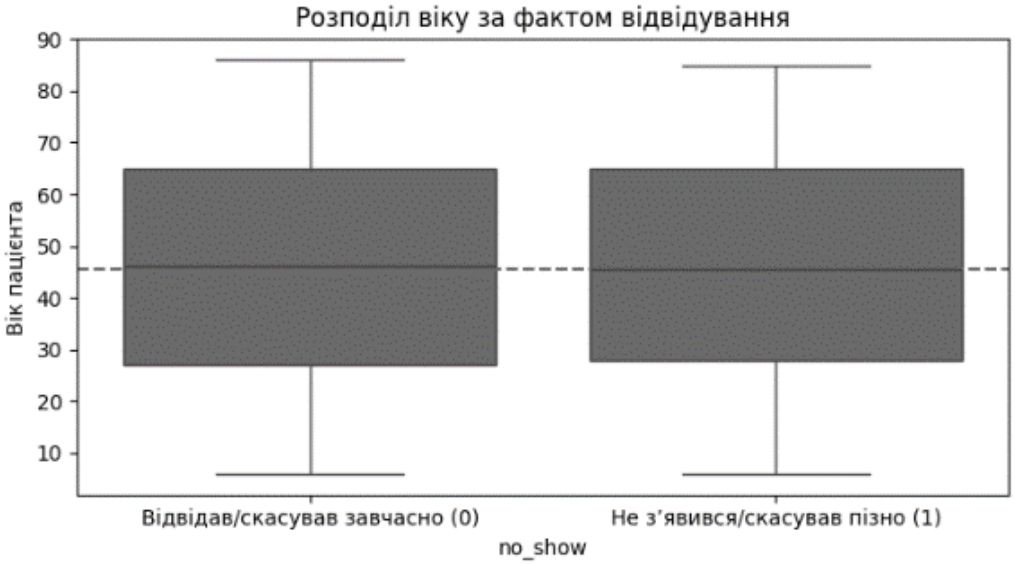



Рисунок 2.10 – Статистика відвідуваності за віком (boxplot + лінія середнього)

## 2.6 Створення моделі та передбачення

В якості змінної, яка підлягає передбаченню, було обрано no_show – факт неявки пацієнта. Для побудови моделі дані були поділені на навчальну та тестову вибірки у співвідношенні 80% до 20%.

Отримані результати моделювання на основі дерева рішень (DecisionTreeClassifier) показують наступне (рис. 2.11):

1) Загальна точність моделі становить ~75%, що означає правильну класифікацію 3 з 4 випадків.

2) Для класу 0 (пацієнти, які відвідали/скасували візит завчасно) модель показує високий рівень якості:

– precision = 0.753 – близько 75% прогнозів «відвідав» є правильними;

– recall = 0.996 – майже всі реальні випадки відвідувань виявлені коректно;

– f1-score = 0.857 – баланс точності та повноти дуже високий.

3) Для класу 1 (неявки) модель працює значно гірше:

– precision = 0.600 – лише 60% прогнозів «неявка» правильні;

– recall = 0.020 – лише 2% реальних неявок виявлені правильно;

– f1-score = 0.039 – майже відсутня практична цінність у виявленні цього класу.

У результаті дерево рішень фактично зосередилося на прогнозуванні більш поширеного класу – відвідувань, ігноруючи менш представлений клас неявок [17].

```
df = data_patients.copy()
X = df.drop(columns=["no_show"])
y = df["no_show"]

num_cols = X.select_dtypes(include=["int64","float64","bool"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object","category"]).columns.tolist()

# препроцесинг
numeric_tf = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="median"))
])
categorical_tf = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="most_frequent")),
    ("oh", OneHotEncoder(handle_unknown="ignore"))
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_tf, num_cols),
        ("cat", categorical_tf, cat_cols),
    ]
)

clf = Pipeline(steps=[
    ("prep", preprocess),
    ("model", DecisionTreeClassifier(max_depth=5, random_state=4353))
])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Точність:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=3))
```

```
Точність: 0.7516666666666667
              precision    recall  f1-score   support

           0      0.753     0.996     0.857       450
           1      0.600     0.020     0.039       150

    accuracy                          0.752       600
   macro avg      0.676     0.508     0.448       600
weighted avg      0.715     0.752     0.653       600
```

Рисунок 2.11 – Класифікація за допомогою дерева рішень (DecisionTreeClassifier)

Виконано класифікацію за допомогою методу k-найближчих сусідів (KNN). Даний метод показав загальну точність у 0.74 (рис. 2.12), що трохи нижче, ніж у дерева рішень (0.75).

Загалом, алгоритм K-Nearest Neighbors виявився неефективним для прогнозування випадків неявки, оскільки модель сильно «зміщена» у бік більшості (відвідувань). Це пов’язано з дисбалансом класів: пацієнтів, які з’являються, значно більше, ніж тих, хто не прийшов. У таких випадках базова точність виглядає високою, але модель не приносить користі для виявлення саме пропусків візитів, що є ключовою задачею [18].

```
df = data_patients.copy()
X = df.drop(columns=["no_show"])
y = df["no_show"]

num_cols = X.select_dtypes(include=["int64","float64","bool"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object","category"]).columns.tolist()

numeric_tf = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])
categorical_tf = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="most_frequent")),
    ("oh", OneHotEncoder(handle_unknown="ignore"))
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_tf, num_cols),
        ("cat", categorical_tf, cat_cols),
    ]
)

knn_clf = Pipeline(steps=[
    ("prep", preprocess),
    ("model", KNeighborsClassifier(n_neighbors=12))
])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

knn_clf.fit(X_train, y_train)
knn_pred = knn_clf.predict(X_test)

print("Точність (KNN):", accuracy_score(y_test, knn_pred))
print(classification_report(y_test, knn_pred, digits=3))
```

```
Точність (KNN): 0.74
              precision    recall  f1-score   support

           0      0.750     0.980     0.850       450
           1      0.250     0.020     0.037       150

    accuracy                          0.740       600
   macro avg      0.500     0.500     0.443       600
weighted avg      0.625     0.740     0.647       600
```

Рисунок 2.12 – Класифікація за допомогою методу K-Nearest Neighbors

Застосовано класифікацію RandomForestClassifier (рис 2.13), що є ключовим методом та поєднує в собі велику кількість дерев рішень. Це дозволяє зменшити ризик перенавчання і підвищити стійкість до даних з випадковими відхиленнями. В результаті навчання модель продемонструвала базову точність на рівні ~73%**,** однак при стандартному порозі класифікації точність виявлення класу «неявка» залишалась низькою. Для покращення цього показника було здійснено оптимізацію порогу під метрику F1-score для класу no_show = 1. Це дало змогу суттєво

підвищити повноту (recall), що є ключовим для задачі виявлення можливих неявок пацієнтів, хоча й відбулося зниження точності (precision).

```
Точність (RF, p>=0.5): 0.733
ROC-AUC: 0.487
Матриця змішувань:
 [[435  15]
 [145   5]]
              precision    recall  f1-score   support

           0      0.750     0.967     0.845       450
           1      0.250     0.033     0.059       150

    accuracy                          0.733       600
   macro avg      0.500     0.500     0.452       600
weighted avg      0.625     0.733     0.648       600


=== Поріг оптимізовано під F1(клас 1) ===
Оптимальний поріг: 0.037
Accuracy: 0.25
F1 (клас 1): 0.4
Матриця змішувань:
 [[  0 450]
 [  0 150]]
              precision    recall  f1-score   support

           0      0.000     0.000     0.000       450
...
    accuracy                          0.253       600
   macro avg      0.625     0.502     0.205       600
weighted avg      0.813     0.253     0.107       600
```

Рисунок 2.13 – Класифікація за допомогою RandomForestClassifier

Таким чином, модель RandomForest дозволяє краще враховувати дисбаланс класів і може бути використана для прогнозування відвідуваності, особливо якщо головна мета – мінімізувати кількість пропущених візитів [19]. Повний код реалізації наведений у додатку Б.

Підсумовуючи, можна зазначити, що RandomForestClassifier продемонстрував вищу ефективність порівняно з DecisionTreeClassifier та K-Nearest Neighbors. Основна перевага полягає в тому, що алгоритм об’єднує велику кількість дерев рішень, що суттєво знижує ризик перенавчання та підвищує стійкість до впливу окремих особливостей вибірки [20]. Завдяки такому підходу метод RandomForestClassifier забезпечує більш надійні та точні прогнози у задачі передбачення відвідуваності пацієнтів.

## Висновки до розділу 2

У другому розділі проведено комплексний аналіз предметної області та сформовано вимоги до розроблюваного програмного забезпечення для стоматологічної клініки з функцією прогнозування відвідуваності пацієнтів. Було визначено призначення, межі та сферу застосування майбутнього вебзастосунку та окреслено функціональні можливості системи.

Особлива увага приділялася опису набору даних, що використовується для навчання моделей машинного навчання. Датасет містить 3000 записів та 11 ознак, які характеризують пацієнтів, лікарів і умови візитів, серед яких: вік, стать, історична відвідуваність, тип та складність процедур, час і день прийому, надсилання нагадувань та показник неявки. Первинний аналіз даних показав, що вибірка є відносно збалансованою за ключовими параметрами, а частка неявок становить близько 25%. Аномальних значень виявлено не було, що підтверджує коректність і придатність датасету для побудови прогнозних моделей.

Було виконано статистичний та візуальний аналіз даних, зокрема розглянуто залежності між віком пацієнтів, їх відвідуваністю та іншими чинниками. Аналіз показав, що значних відмінностей між віковими групами немає.

На наступному етапі було проведено моделювання за допомогою різних алгоритмів машинного навчання. Найбільш ефективним виявився алгоритм RandomForestClassifier, який як ансамблевий метод поєднує велику кількість дерев рішень. Це дозволило підвищити стійкість моделі до варіативності даних.

Таким чином, у ході проведеного аналізу було підтверджено доцільність використання ансамблевих методів, зокрема RandomForest, для задачі прогнозування відвідуваності пацієнтів. Він продемонстрував кращі результати порівняно з DecisionTreeClassifier та K-Nearest Neighbors і є найбільш перспективним для інтеграції у розроблюваний вебзастосунок [21].

# 3 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

## 3.1 Вибір технологій та мов програмування

Під час розробки вебзастосунку було обрано такі технології:

1) Мова програмування та фреймворк: Next.js (React, TypeScript).
2) Адміністративна панель: NextAdmin.
3) СКБД: PostgreSQL [22].
4) ORM: Prisma ORM [23].

Обраний стек технологій забезпечує ефективну, передбачувану та масштабовану розробку вебзастосунку стоматологічної клініки з функцією прогнозування відвідуваності пацієнтів. Його архітектура спрямована на швидку обробку даних, стабільну роботу сервісу та зручне адміністрування.

Мова програмування Typescript додає типізацію, автодоповнення та перевірку типів під час компіляції, що особливо важливо для великих систем з великою кількістю сутностей, таких як: пацієнти, лікарі, процедури чи розклади прийомів. Завдяки TypeScript зменшується кількість помилок на етапі розробки, підвищується надійність коду та полегшується підтримка проєкту.

Основними перевагами використання Typescript є:

1) підтримка сучасного синтаксису ECMAScript;
2) покращена інтеграція з IDE (Visual Studio Code);
3) сумісність з JavaScript-бібліотеками.

Next.js – це сучасний React-фреймворк, який поєднує клієнтську та серверну логіку в одному середовищі [24]. Фреймворк підтримує SSR (Server-Side Rendering), SSG (Static Site Generation) та ISR (Incremental Static Regeneration) – технології, які забезпечують:

1) миттєве завантаження сторінок;
2) оптимізацію для пошукових систем (SEO);
3) динамічне оновлення даних без повного перезавантаження сторінки.

Переваги використання Next.js у вебзастосунку:

1) поєднання фронтенду та бекенду в одному середовищі;
2) ефективна робота з динамічними даними (розклади, записи);
3) висока продуктивність і безпека;
4) гнучка інтеграція з системами аналітики та машинного навчання.

PostgreSQL – це об’єктно-реляційна система керування базами даних (СКБД) з відкритим вихідним кодом, яка поєднує надійність класичних реляційних моделей із гнучкістю сучасних підходів до роботи з даними. Ця система підтримує повний набір можливостей для створення, зберігання, обробки та захисту структурованих даних. Система забезпечує повну транзакційність і суворе дотримання ACID-принципів (Atomicity, Consistency, Isolation, Durability), що гарантує цілісність даних навіть у разі збоїв. PostgreSQL підтримує складні SQL-запити, тригери, представлення, збережені процедури та функції, що робить її зручною для реалізації складних логічних операцій на рівні бази даних. Серед ключових технічних можливостей варто відзначити підтримку типів JSON та JSONB, які дають змогу ефективно працювати з напівструктурованими даними. Крім того, PostgreSQL має високопродуктивну систему індексації (B-tree, Hash, GIN, GiST, BRIN), що забезпечує швидкий пошук і фільтрацію великих обсягів інформації. Завдяки масштабованості, реплікації та розвиненій системі безпеки (SSL, контроль доступу, ролі та привілеї), PostgreSQL вважається однією з найнадійніших і найгнучкіших баз даних у світі.

Переваги PostgreSQL для медичних систем:

1) висока надійність і цілісність даних;
2) підтримка розширень (uuid-ossp, pg_cron, postgis) для генерації унікальних ідентифікаторів, автоматичних задач та просторових даних;
3) потужні механізми безпеки (аутентифікація, шифрування, контроль доступу);

4) можливість виконання аналітичних запитів для формування звітів про відвідування, навантаження лікарів та ефективність прогнозів.

Prisma ORM – це інструмент, який спрощує роботу з базою даних, дозволяючи розробнику взаємодіяти з таблицями через об’єктну модель замість написання SQL-коду. Це робить розробку більш швидкою, безпечною та передбачуваною. Prisma автоматично генерує типобезпечний клієнт, який синхронізується зі схемою бази даних, забезпечуючи коректність і надійність у виконанні запитів до даних.

У застосунку Prisma використовується для:

1) створення та оновлення таблиць пацієнтів, лікарів, візитів;
2) запису результатів прогнозу неявки;
3) виконання складних запитів до історичних даних;
4) міграцій бази при зміні структури таблиць.

Основні переваги Prisma ORM:

1) автоматичне створення типів на основі схеми БД;
2) підтримка реляцій між таблицями (one-to-many, many-to-many);
3) висока продуктивність запитів;
4) зручний CLI для керування міграціями.

NextAdmin – це сучасне рішення для створення гнучких і функціональних адміністративних панелей, побудоване на основі фреймворку Next.js. Цей інструмент поєднує у собі зручність користувацького інтерфейсу, високу швидкодію та глибоку інтеграцію з серверною частиною застосунку. NextAdmin забезпечує ефективне керування внутрішніми процесами та даними, дозволяючи адміністраторам і працівникам отримувати швидкий доступ до ключових сутностей системи, відстежувати показники ефективності та взаємодіяти з інформацією у реальному часі.

Основні можливості NextAdmin включають:

1) розширену фільтрацію та сортування даних, що забезпечує швидкий пошук потрібної інформації;

2) побудову інтерактивних дашбордів із ключовими показниками, такими як відсоток неявок, середня тривалість прийому, частота повторних візитів;

3) інтеграцію з Prisma ORM, завдяки якій панель отримує прямий доступ до бази даних без необхідності створення додаткових API або проміжних сервісів.

Завдяки своїй архітектурі NextAdmin дає змогу швидко створювати кастомні модулі, розширювати функціональність системи та забезпечувати зручну роботу персоналу без потреби у глибоких технічних знаннях. Це робить її універсальним інструментом для ефективного адміністрування будь-яких вебзастосунків.

Обґрунтування технологій наведено у табл. 3.1.

Таблиця 3.1 – Обґрунтування вибору технологій

| **№** | **Критерій** | **Обґрунтування** |
|---|---|---|
| 1 | Продуктивність і швидкість | Next.js з SSR забезпечує швидке завантаження сторінок та SEO-оптимізацію. |
| 2 | Надійність даних | PostgreSQL гарантує цілісність транзакцій і підтримує великі обсяги записів. |
| 3 | Безпечність | TypeScript і Prisma зменшують ризик логічних помилок. |
| 4 | Масштабованість | Архітектура Next.js дозволяє легко розширювати систему – додавати нові модулі або інтегрувати ШІ-моделі. |
| 5 | Швидкість розробки | NextAdmin скорочує час створення внутрішніх панелей управління. |

Комбінація технологій Next.js, NextAdmin, Prisma ORM та PostgreSQL створює надійну, безпечну та гнучку архітектуру для вебзастосунку стоматологічної клініки. Таким чином, обраний стек технологій відповідає як технічним, так і функціональним вимогам проєкту, забезпечуючи основу для подальшого розвитку системи у сфері цифрової медицини.

## 3.2 Створення сценаріїв

Use Case сценарій – це спосіб опису функціональності системи через моделювання окремих сценаріїв використання або взаємодії користувачів із системою. Це один із ключових етапів аналізу вимог, який допомагає зрозуміти, як система має поводитися у різних ситуаціях та для різних типів користувачів (адміністратор, лікар, пацієнт). Під час створення вебзастосунку стоматологічної клініки з функцією прогнозування відвідування пацієнтів, сценарії допомогли деталізувати функціональні вимоги, логіку роботи модулів, а також послідовність дій користувачів у типових ситуаціях.

***Короткий сценарій***

Пацієнт телефонує до клініки. Адміністратор відкриває застосунок, знаходить або створює картку пацієнта, обирає лікаря та доступний час запису. Адміністратор узгоджує варіант із пацієнтом і фіксує запис. Система надсилає SMS або email-повідомлення з деталями прийому.

***Середній сценарій***

*Головний сценарій (успішний):*

Пацієнт телефонує до стоматологічної клініки, щоб записатися на прийом. Адміністратор відкриває вебзастосунок клініки та переходить до модуля «Запис пацієнтів». Він знаходить у базі даних наявного пацієнта або створює нову картку, вносячи основні дані (ПІБ, телефон, дата народження). Після цього адміністратор обирає лікаря, послугу та бажану дату і час прийому. Система автоматично

перевіряє, чи є обраний час вільним у розкладі лікаря. Адміністратор підтверджує запис, і система зберігає його у базі даних. Пацієнт отримує SMS або email-повідомлення з деталями прийому (дата, час, лікар, адреса клініки). Інформація про запис автоматично з’являється у розкладі лікаря та у звітах адміністратора.

*Альтернативний сценарій (висока ймовірність неявки, виявлена під час щоденного оновлення):*

Після завершення робочого дня система автоматично запускає модуль машинного навчання, який аналізує усі створені записи та оновлює прогноз імовірності неявки пацієнтів.

Система формує рекомендації для кожного такого запису, наприклад:

1) надіслати додаткові нагадування (за 24 години і за 3 години до прийому);

2) зв’язатися з пацієнтом для підтвердження запису;

Адміністратор узгоджує варіант із пацієнтом і вносить відповідні зміни у запис. Після цього система надсилає SMS або email-повідомлення із уточненими деталями прийому.

***Повний сценарій***

Таблиця 3.2 – Повний сценарій

| Поле | Опис |
|---|---|
| Use Case Name | Запис пацієнта телефоном з подальшим прогнозуванням неявки. |
| Scope | Вебзастосунок стоматологічної клініки з автоматичним ML-модулем прогнозування. |

Продовження таблиці 3.2

| | |
|---|---|
| Primary Actor | Адміністратор. |
| Secondary Actors | Лікар (отримує розклад), SMS або email-повідомлення (розсилки нагадувань), ML-сервіс (щоденне оновлення прогнозів). |
| Stakeholders & Interests | Пацієнт: зручний запис і нагадування. Адміністратор: швидке внесення даних пацієнта. Лікар: актуальний розклад. Керівництво: доступ до щоденної аналітики ризику неявок і динаміки відвідуваності. |
| Trigger | Вхідний дзвінок від пацієнта з проханням записатися на прийом. |
| Preconditions | Адміністратор авторизований; розклад синхронізовано; у системі наявні дані про пацієнтів, лікарів та послуги; активний модуль ML-аналітики (налаштований щоденний запуск). |
| Minimal Guarantees | Дії адміністратора логуються; система надає зрозумілі повідомлення про помилки. |
| Success Guarantees (Postconditions) | Запис створено та збережено у базі; пацієнту відправлено SMS або email-повідомлення; розклад лікаря оновлено. |

Продовження таблиці 3.2

| | |
|---|---|
| Main Success Scenario | 1. Пацієнт телефонує до клініки.<br>2. Адміністратор відкриває модуль “Запис пацієнтів”.<br>3. Знаходить або створює картку пацієнта.<br>4. Обирає лікаря, послугу, дату й час.<br>5. Система перевіряє доступність наявного часу.<br>6. Адміністратор підтверджує запис.<br>7. Система зберігає запис і надсилає пацієнту email-підтвердження.<br>8. Наприкінці дня ML-модуль автоматично обробляє всі нові записи, оновлюючи прогноз неявки пацієнтів. |
| Extensions | 1. Відсутні контакти пацієнта: адміністратор додає телефон або пошту для відправки повідомлень.<br>2. Невдала розсилка: автоматичний повтор відправлення email або SMS підтвердження. |

Кінець таблиці 3.2

| | |
|---|---|
| Special Requirements | 1. Обробка запиту менше 5 секунд.<br>2. Надійне логування.<br>3. Автоматичний CRON-запуск ML-модуля щодня о визначений час. |
| Technology & Data Variations List | PostgreSQL + Prisma ORM (пацієнти, записи, розклад, історія неявок); ML (щоденний запуск через CRON); черга повідомлень для розсилок; аналітичний дашборд у NextAdmin. |
| Assumptions | ML-модуль має доступ до актуальних записів і історичних даних; адміністратор оновлює статуси “відвідав/не з’явився” для коректного навчання моделі. |
| Open Issues | Визначення періодичності оновлення (1 раз/доба чи частіше); поріг ризику для автоматичних дій; формат виведення статистики у дашборді. |

Перевагою табличних та текстових сценаріїв є те що їх легше читати та розуміти замовнику та легше змінювати ніж діаграми.

## 3.3 Створення use cases діаграм (варіантів використання)

Діаграма варіантів використання – це графічне представлення можливої взаємодії користувачів із системою (рис. 3.1). Вона демонструє різні способи використання системи та типи користувачів, які з нею працюють, і зазвичай

доповнюється діаграмами інших типів [25]. Такий інструмент дозволяє візуалізувати основні сценарії використання, визначити ролі учасників і межі їхньої взаємодії із системою.

Основними елементами діаграми є актори (позначаються у вигляді фігур людей або зовнішніх систем) і варіанти використання (зображуються еліпсами), що відображають конкретні дії або функції, доступні користувачам.

Діаграма варіантів використання допомагає уточнити вимоги до системи та визначити, як вона застосовуватиметься на практиці. Вона особливо корисна на етапі аналізу вимог, адже дозволяє сформувати уявлення про функціональність системи з точки зору користувача. Такий підхід сприяє кращому розумінню між розробниками, замовниками та іншими зацікавленими сторонами щодо того, як система повинна працювати у реальних умовах і які задачі має вирішувати.

Крім того, діаграма варіантів використання часто виступає основою для подальшого проєктування архітектури системи. Використання таких діаграм сприяє точнішому визначенню функціональних вимог, підвищує якість розробки та забезпечує створення програмного забезпечення, яке максимально відповідає потребам користувачів.

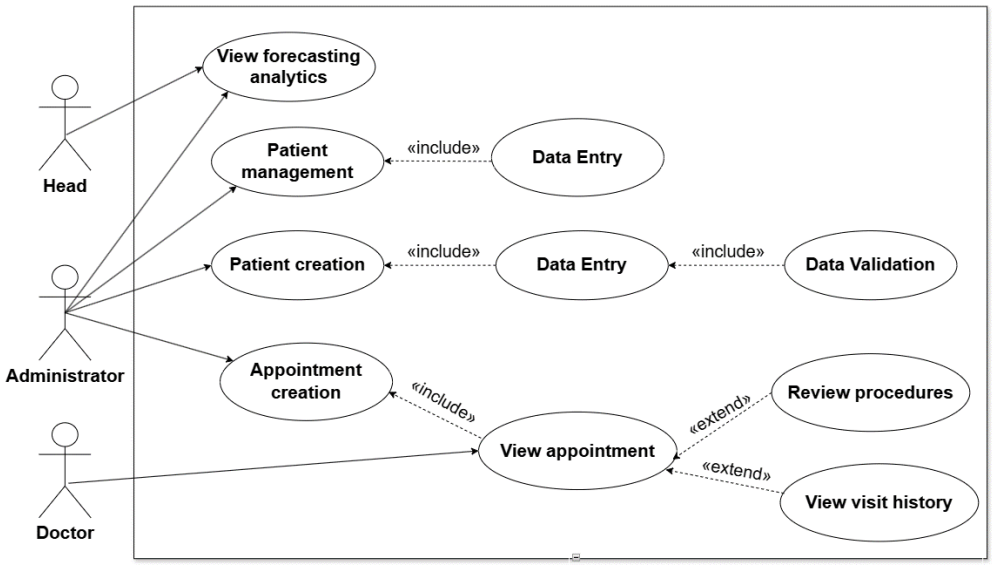



Рисунок 3.1 – Діаграма варіантів використання

На рис. 2.1 представлено діаграму варіантів використання, яка демонструє основні функціональні можливості системи та взаємодію користувачів із нею. Система призначена для ефективного керування процесом роботи стоматологічної клініки, включаючи створення, редагування та перегляд записів пацієнтів, а також прогнозування ймовірності неявки на прийом.

Адміністратор має найширші можливості у системі, а саме може створювати та редагувати інформацію про пацієнтів, лікарів, клініки, а також керувати записами на прийом. Крім того, адміністратор відповідає за валідацію даних, тобто перевірку правильності введених відомостей (формат дати, контактних даних, наявність лікаря у розкладі тощо). Також має доступ до функції перегляду прогнозів неявок, що дозволяє оцінити ймовірність відвідування пацієнта та оптимізувати графік прийомів.

Лікар у системі може переглядати власні записи на прийом, ознайомлюватися з історією відвідувань пацієнтів та отримувати доступ до процедур, призначених конкретному пацієнту. Це дозволяє лікарю ефективно планувати робочий час і готуватися до прийому.

Керівник клініки має можливість переглядати аналітику прогнозування відвідуваності пацієнтів, що дає змогу оцінювати загальні показники роботи клініки, визначати рівень неявок, навантаження на лікарів і ефективність розкладу.

На рис. 2.2 зображено діаграму станів, що описує життєвий цикл об’єкта «Запис на прийом» у системі [26]. Вона відображає послідовність переходів між станами під час створення, редагування, перевірки та завершення запису пацієнта до лікаря. Після створення нового запису він переходить у стан «Appointment has been created». Далі система виконує валідацію даних (перевірку правильності заповнення полів, дати, лікаря, доступності часу). Якщо виявлено помилки, запис повертається у стан редагування. Якщо ж дані підтверджено, запис переходить до стану «Appointment in Database», де він зберігається в базі даних. На етапі обробки

адміністратор може редагувати або видаляти запис. У разі редагування об’єкт переходить у стан «Appointment is being edited», після чого знову проходить перевірку. У випадку скасування запис переходить у стан «Appointment has been deleted», після чого процес завершується.

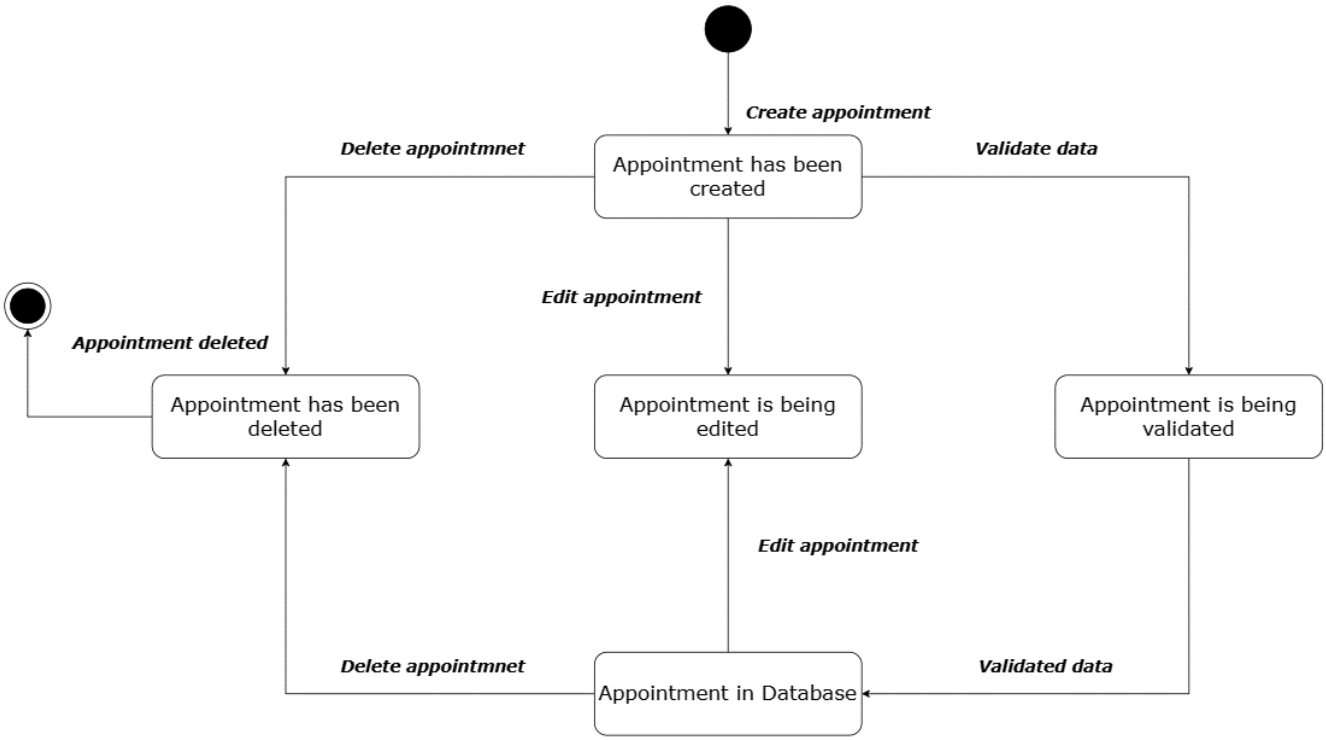



Рисунок 3.2 – Діаграма станів запису пацієнтів

Таким чином, діаграма станів відображає логічний ланцюг життєвого циклу запису пацієнта, від моменту створення до видалення, забезпечуючи контроль даних, їхню валідацію та узгодженість у базі.

## 3.4 Проєктування та опис бази даних

Під час розробки вебзастосунку для стоматологічної клініки з функцією прогнозування відвідування пацієнтів було створено реляційну базу даних, що забезпечує зберігання, обробку та доступ до інформації про користувачів, лікарів, пацієнтів, процедури, а також дані для прогнозування неявок. Для роботи з базою даних використано PostgreSQL.

PostgreSQL – це надійна та масштабована система керування базами даних з відкритим кодом. Взаємодія із СКБД здійснюється за допомогою Prisma ORM, яка забезпечує високорівневий доступ до даних через об’єктно-реляційне відображення. Це дозволяє уникати написання сирих SQL-запитів, спрощує міграції та підтримку структури бази.

База даних містить 14 таблиць, кожна з яких відповідає окремій сутності предметної області (рис. 3.3):

1) User – зберігає дані всіх користувачів системи (адміністраторів, лікарів). Поля: id, name, email, image, coverImage, password, role, createdAt, address, isActive, phone, updatedAt.

2) Administrator – містить інформацію про адміністраторів, які мають повний доступ до системи. Поля: id, name, email, password, role, createdAt.

3) Doctor – зберігає дані про лікарів клініки. Поля: id, clinicId, fullName, speciality, experienceYears, active, createdAt.

4) Patient – таблиця з інформацією про пацієнтів. Поля: id, clinicId, firstName, lastName, birthDate, phone, email, address, gdprConsent, createdAt, gender.

5) Clinic – зберігає інформацію про клініки, що входять до системи. Поля: id, name, address, timeZone, createdAt.

6) Room – таблиця для зберігання даних про кабінети в клініці. Поля: id, clinicId, name, notes, createdAt.

7) ProcedureType – містить перелік стоматологічних процедур (пломбування, чистка, імплантація тощо). Поля: id, code, name, defaultDurationMin, complexityLevel, createdAt.

8) Appointment – основна таблиця записів пацієнтів на прийом. Поля: id, clinicId, patientId, doctorId, roomId, scheduledStart, scheduledEnd, status, createdByUserId, notes, source, createdAt.

9) AppointmentProcedure – проміжна таблиця для зв’язку між Appointment і ProcedureType. Поля: appointmentId, procedureTypeId, plannedDurationMin, pricePlanned, createdAt.

10) PatientPreferences – зберігає індивідуальні налаштування або вподобання пацієнтів. Поля: id, patientId, preferredChannel, preferredTimeBlock, createdAt.

11) Confirmation – таблиця для фіксації підтвердження візитів (SMS, email). Поля: id, appointmentId, patientId, confirmed, channel, confirmedAt, sources, createdAt.

12) Communication – містить історію сповіщень між лікарем, пацієнтом та системою. Поля: id, appointmentId, patientId, channel, type, sentAt, status, metadata, createdAt.

13) Session – таблиця для зберігання активних сесій користувачів. Поля: id, sessionToken, userId, expires.

14) _prisma_migrations – службова таблиця Prisma ORM, що зберігає історію змін у структурі бази.

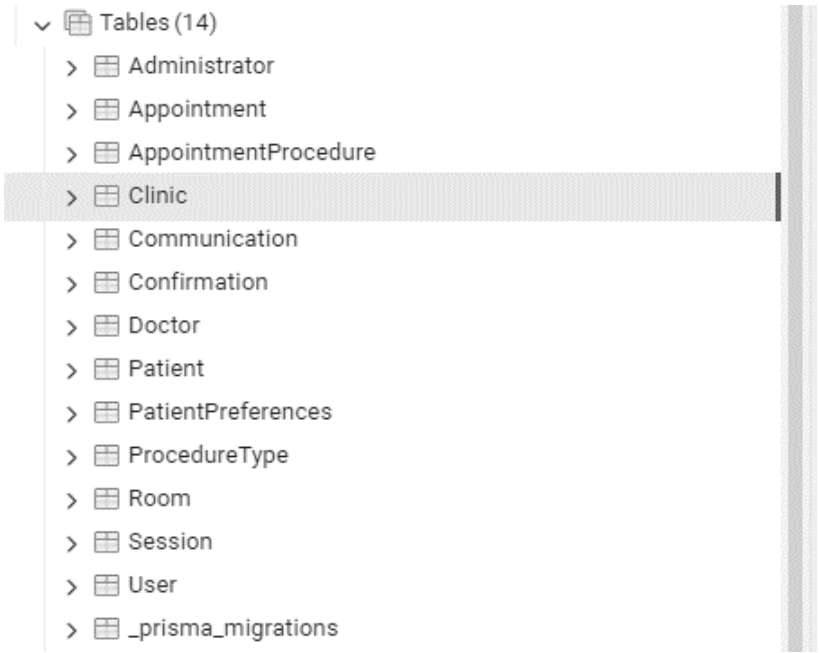



Рисунок 3.3 – Таблиці бази даних

Така структура забезпечує логічну цілісність даних, зручність доступу до інформації та можливість подальшого розширення системи без порушення існуючих зв’язків між сутностями.

## 3.5 Створення діаграми класів

Діаграма класів – це один із основних типів діаграм UML, який використовується для моделювання структури системи шляхом відображення її класів, атрибутів, методів та зв’язків між ними. На рисунку 3.4 зображено діаграму класів, побудовану на основі структури бази даних веб-застосунку «Стоматологічна клініка з функцією прогнозування відвідування пацієнтів». Діаграма створена засобами pgAdmin 4 і відображає таблиці бази даних, які відповідають класам системи, а також зв’язки між ними через зовнішні ключі. Така візуалізація є аналогом UML-діаграми класів, що показує основні сутності системи, їх атрибути та взаємозв’язки.

На діаграмі класів вебзастосунку» продемонстровано основні сутності, що формують структуру бази даних. Кожен клас відповідає таблиці в базі даних, а лінії між ними відображають зв’язки «один-до-багатьох» або «один-до-одного» на основі зовнішніх ключів.

Завдяки діаграмі класів можна простежити логіку зберігання, обробки та передачі інформації між різними компонентами: лікарями, адміністраторами, записами на прийом, процедурами. Діаграма класів також є важливим інструментом під час розробки, тестування та супроводу системи, оскільки допомагає забезпечити узгодженість між логічною моделлю даних і фізичною структурою бази. Вона слугує основою для реалізації ORM-моделей у Prisma, що спрощує взаємодію з PostgreSQL та гарантує цілісність і узгодженість інформації в межах усього вебзастосунку.

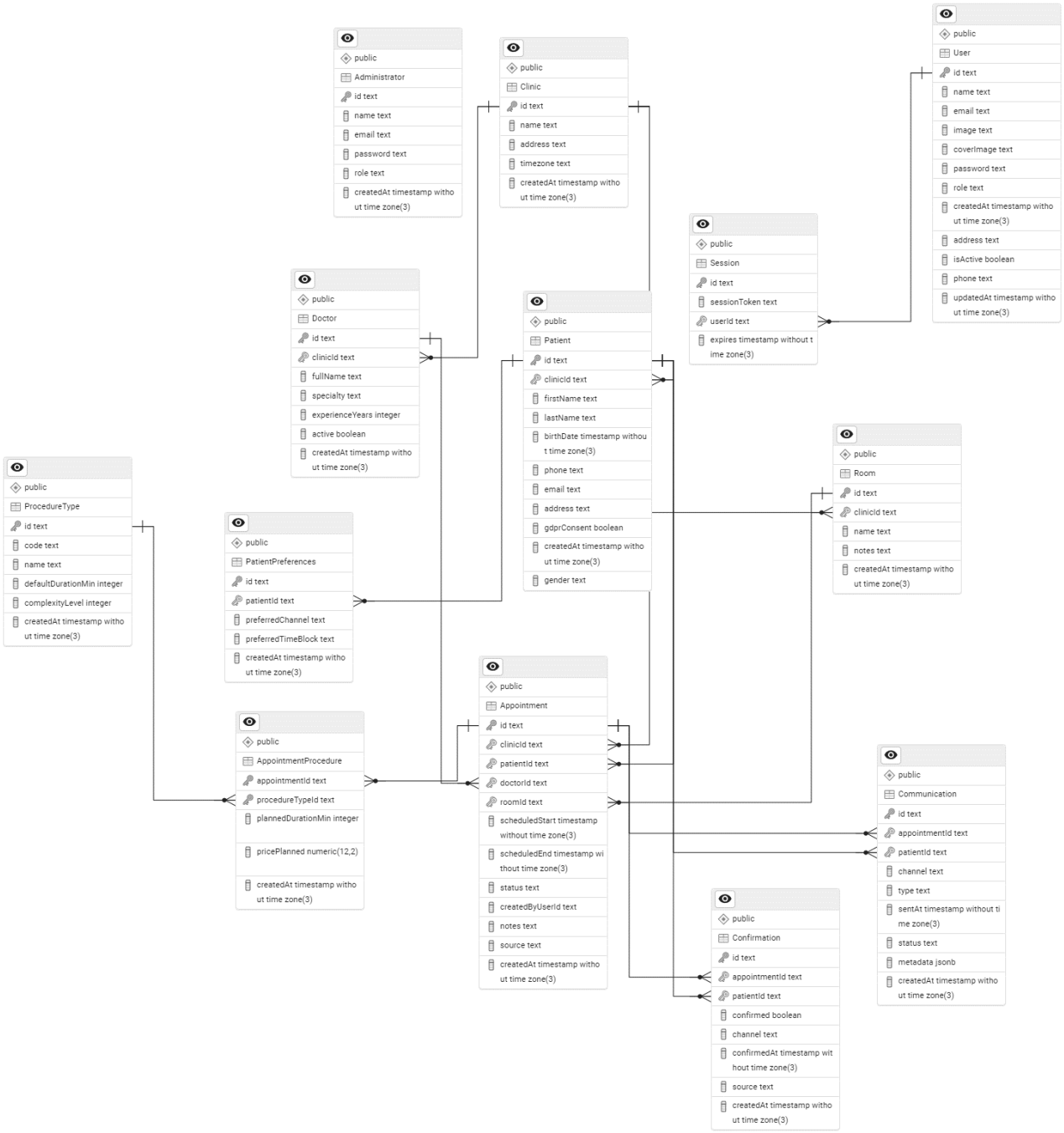



Рисунок 3.4 – Діаграма класів

Таким чином, діаграма класів відображає логічну структуру всієї системи управління стоматологічною клінікою.

## Висновки до розділу 3

У третьому розділі виконано проєктування та моделювання вебзастосунку стоматологічної клініки з функцією прогнозування відвідування пацієнтів. На основі аналізу функціональних вимог обґрунтовано вибір технологій, мов програмування та інструментів розробки, які забезпечують ефективну, надійну та масштабовану роботу системи.

Створено сценарії взаємодії користувачів із системою, які відображають типові процеси запису пацієнтів, обробки даних та прогнозування неявок. На їхній основі розроблено діаграму варіантів використання, що ілюструє основні ролі користувачів (адміністратор, лікар, керівник клініки) та їхні функції у межах системи.

Для моделювання життєвого циклу об’єкта “Запис на прийом” створено діаграму станів, яка відображає послідовність переходів між станами під час створення, редагування, підтвердження або скасування записів пацієнтів.

Розроблено реляційну базу даних з 14 таблиць, що охоплюють усі сутності предметної області. Така структура забезпечує логічну цілісність даних, підтримку реляційних зв’язків та гнучкість у масштабуванні системи.

Побудовано діаграму класів, яка відображає структуру бази даних і зв’язки між сутностями у вигляді UML-моделі. Вона демонструє логічну архітектуру системи та слугує основою для створення ORM-моделей у Prisma.

Таким чином, у результаті проєктування було сформовано повну архітектуру вебзастосунку, що забезпечує надійну взаємодію між компонентами, узгодженість даних і можливість подальшого розвитку системи, зокрема інтеграції модулів аналітики та машинного навчання для прогнозування поведінки пацієнтів.

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

## 4.1 Реалізація клієнтської частини

Стоматологічний вебзастосунок побудований на сучасному технологічному стеці з використанням Next.js 15 як основного фреймворку для розробки. Проєкт використовує TypeScript для забезпечення типізації та покращення якості коду, що є критично важливим для медичних застосунків, де точність даних має вирішальне значення.

Для стилізації використовується Tailwind CSS з кастомною конфігурацією, що включає:

1) розширену палітру кольорів з підтримкою темної теми;
2) кастомні шрифти;
3) анімації та переходи;
4) систему тіней та ефектів для покращення UX.

Застосунок використовує App Router Next.js 15 з групуванням маршрутів (рис 4.1).

```
src/app/
├── (site)/           # Основні сторінки застосунку
│   ├── (home)/       # Головна сторінка
│   ├── dashboard/    # Головна панель
│   ├── patients/     # Управління пацієнтами
│   ├── appointments/ # Управління записами
│   ├── doctors/      # Управління лікарями
│   ├── clinics/      # Управління клініками
│   ├── procedures/   # Управління процедурами
│   ├── rooms/        # Управління кабінетами
│   ├── forecasting/  # Прогнозування відвідуваності
│   └── users/        # Управління користувачами
├── api/              # API маршрути
└── auth/             # Авторизація
```

Рисунок 4.1 – Маршрутизація App Router

Однією з ключових нових можливостей є повна інтернаціоналізація застосунку. Система локалізації побудована на власному LanguageContext, який забезпечує динамічне завантаження перекладів з fallback механізмами. Підтримуються англійська (основна) та українська мови. Система автоматично завантажує відповідні переклади при зміні мови та зберігає вибір користувача в localStorage. Структура перекладів організована за функціональними групами: dashboard (головна панель), forecasting (прогнозування), navigation (навігація) та common (загальні елементи). Кожен компонент використовує функцію перекладу для динамічного відображення контенту відповідно до обраної мови.

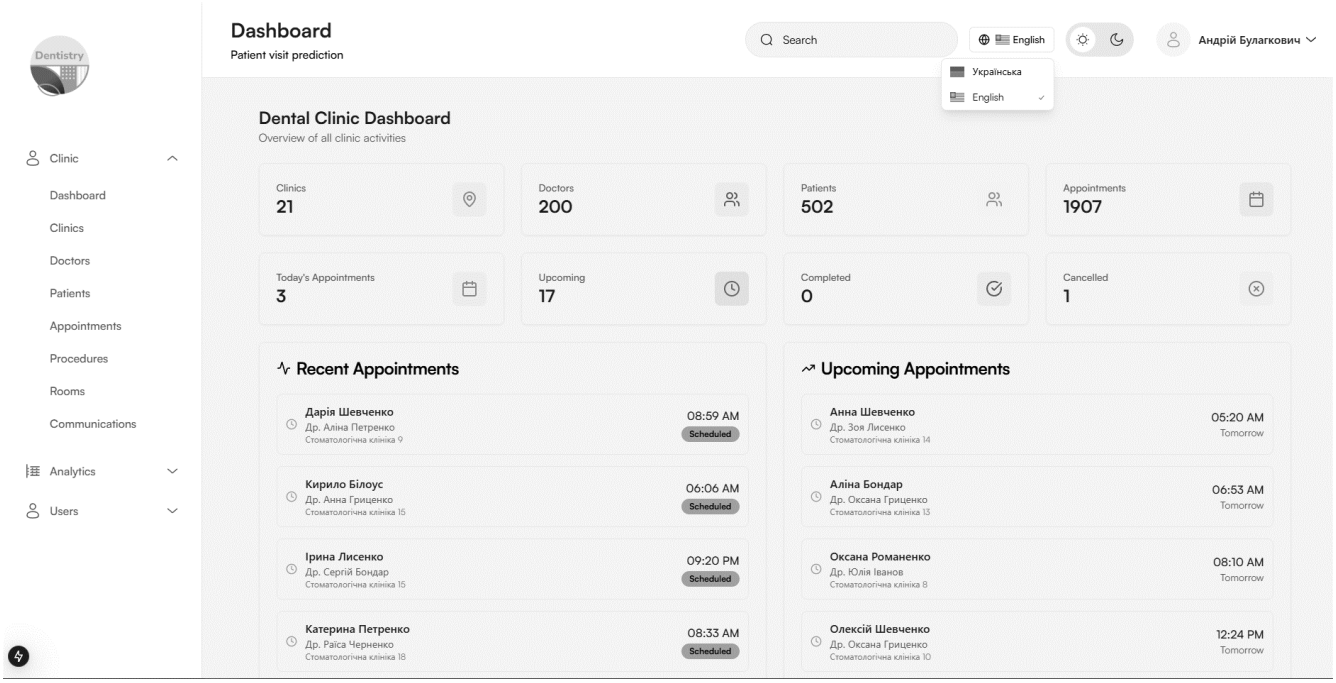



Рисунок 4.2 – Підтримка зміни мови інтерфейсу

Застосунок побудований на модульній компонентній архітектурі з чіткою ієрархією компонентів. UI компоненти розташовані в окремій директорії та включають універсальний FormModal для створення форм у модальних вікнах, базові елементи інтерфейсу (Button, Input, Card), розширені компоненти форми (Badge, Select, Textarea), компоненти модальних вікон (Modal, Dialog) та

спеціалізовані компоненти для прогнозування. Layout компоненти забезпечують структуру застосунку та включають Header з навігацією, пошуком, перемикачем теми та користувацькими функціями, Sidebar з ієрархічною структурою меню та мобільною адаптацією, а також SidebarProvider для управління станом бічної панелі з автоматичним збереженням в localStorage.

Dashboard є центральною сторінкою застосунку та надає комплексний огляд всієї діяльності клініки з повною локалізацією. Сторінка включає статистичні картки з загальною статистикою (кількість клінік, лікарів, пацієнтів, записів) та щоденною статистикою (записи на сьогодні, майбутні записи, завершені, скасовані). Всі елементи підтримують переклади та адаптуються до обраної мови.

Сторінка управління пацієнтами (рис 4.3) надає повноцінну систему управління з розширеними можливостями. Форма додавання пацієнта використовує покращений компонент FormModal з підтримкою обов’язкових полів (клініка, ім'я, прізвище, стать), додаткових полів (дата народження, телефон, email, адреса), GDPR згоди та покращеної валідації з відображенням помилок у реальному часі (рис 4.4).

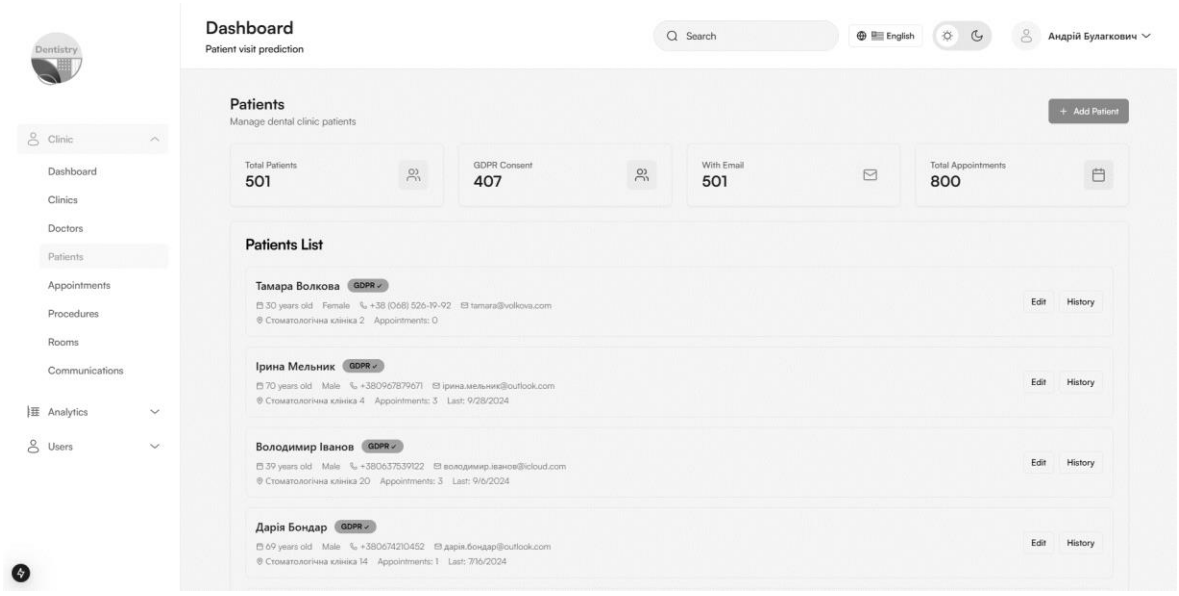



Рисунок 4.3– Сторінка управління пацієнтами

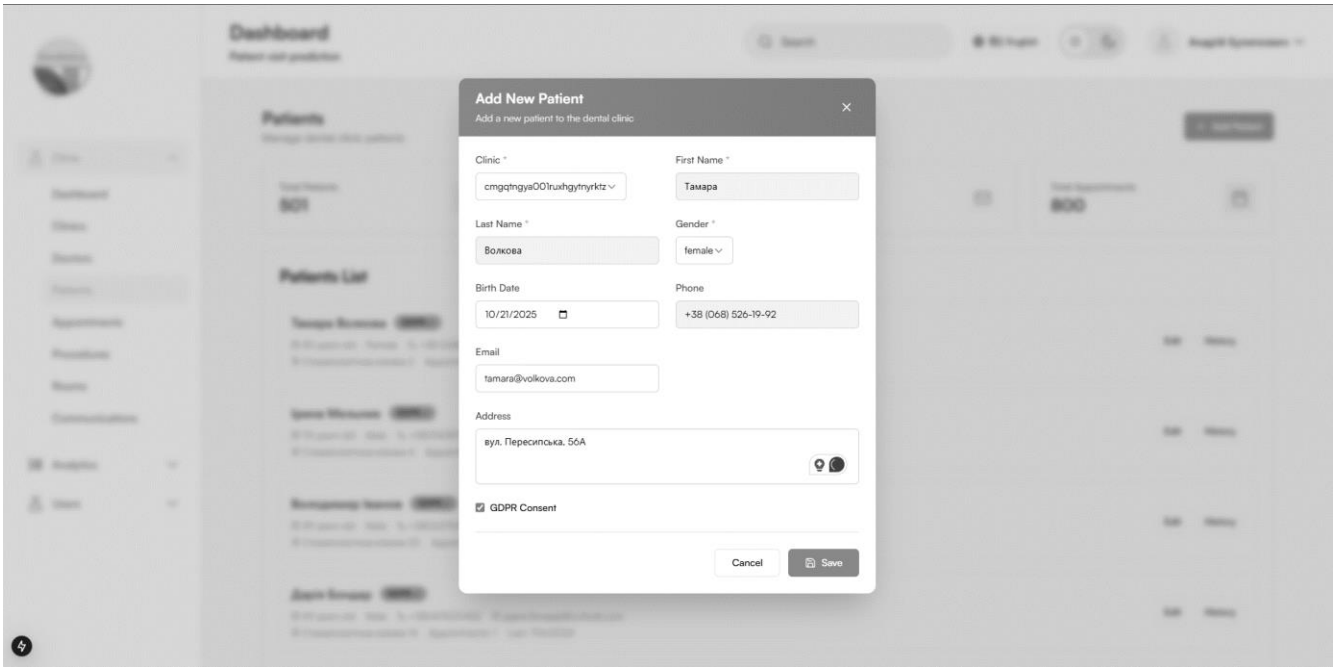



Рисунок 4.4 – Додавання нового пацієнта

Сторінка управління записами забезпечує комплексну систему планування та управління записами з автоматичними розрахунками тривалості та часу закінчення на основі обраних процедур (рис 4.5).

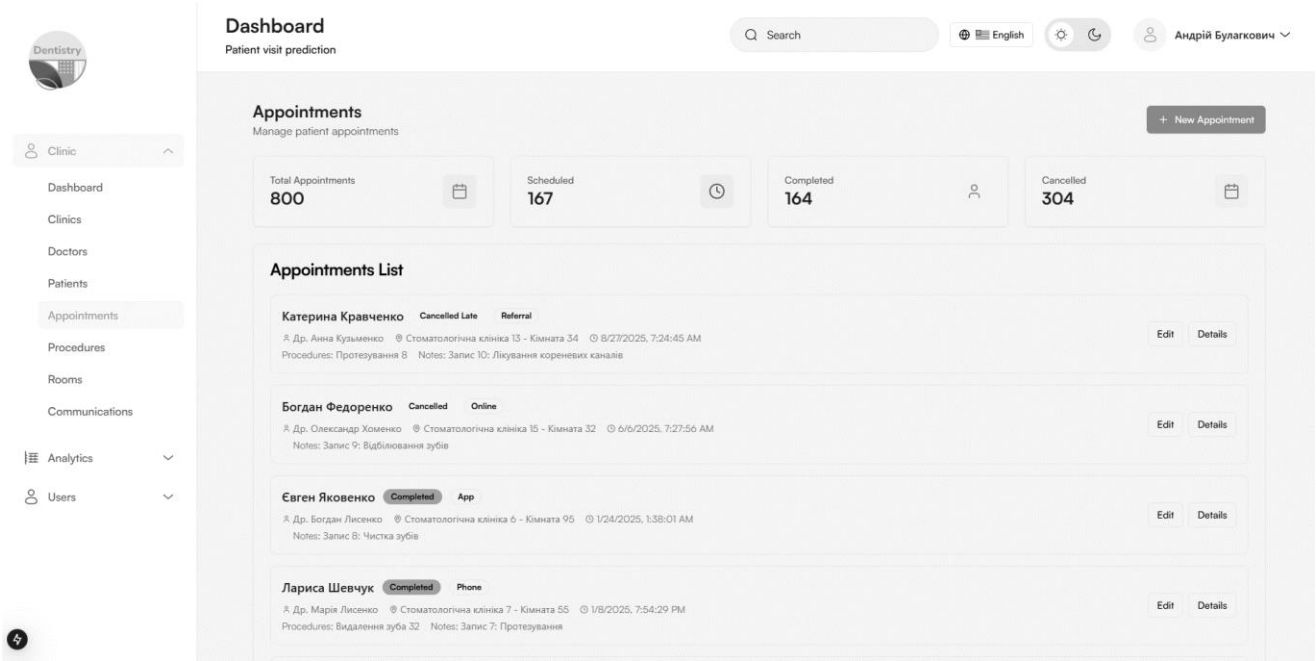



Рисунок 4.5 – Управління записами

Сторінка управління процедурами забезпечує комплексну систему планування та адміністрування процедур з автоматичними розрахунками тривалості та часу завершення на основі обраних параметрів (рис 4.6).

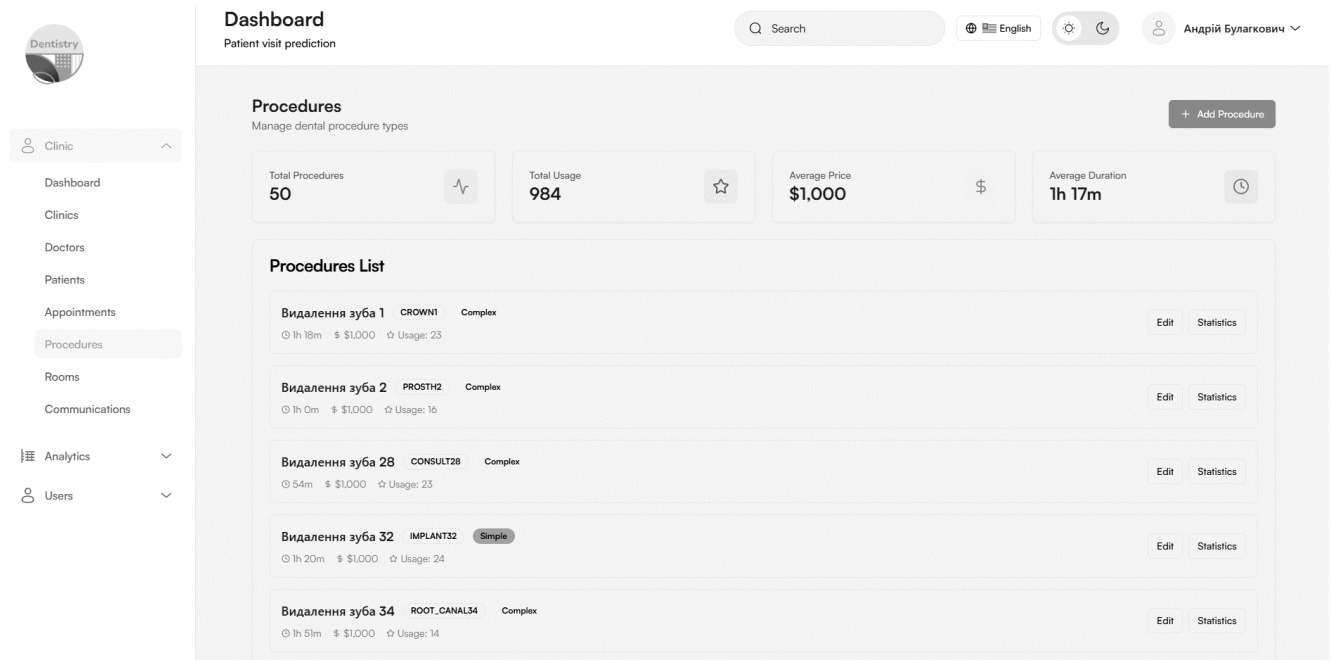



Рисунок 4.6 – Управління процедурами

Сторінка управління комунікаціями забезпечує зручну та ефективну систему взаємодії з пацієнтами, яка дозволяє надсилати повідомлення через SMS або здійснювати дзвінки за вказаним номером телефону (рис. 4.7). Завдяки цій функціональності адміністратор може швидко повідомляти пацієнтів про підтвердження запису, нагадування про прийом, зміни у розкладі чи необхідність повторного візиту (рис. 4.8). Система також сприяє підвищенню рівня сервісу та зменшенню кількості пропущених прийомів, оскільки пацієнти отримують актуальну інформацію вчасно й у зручному форматі.

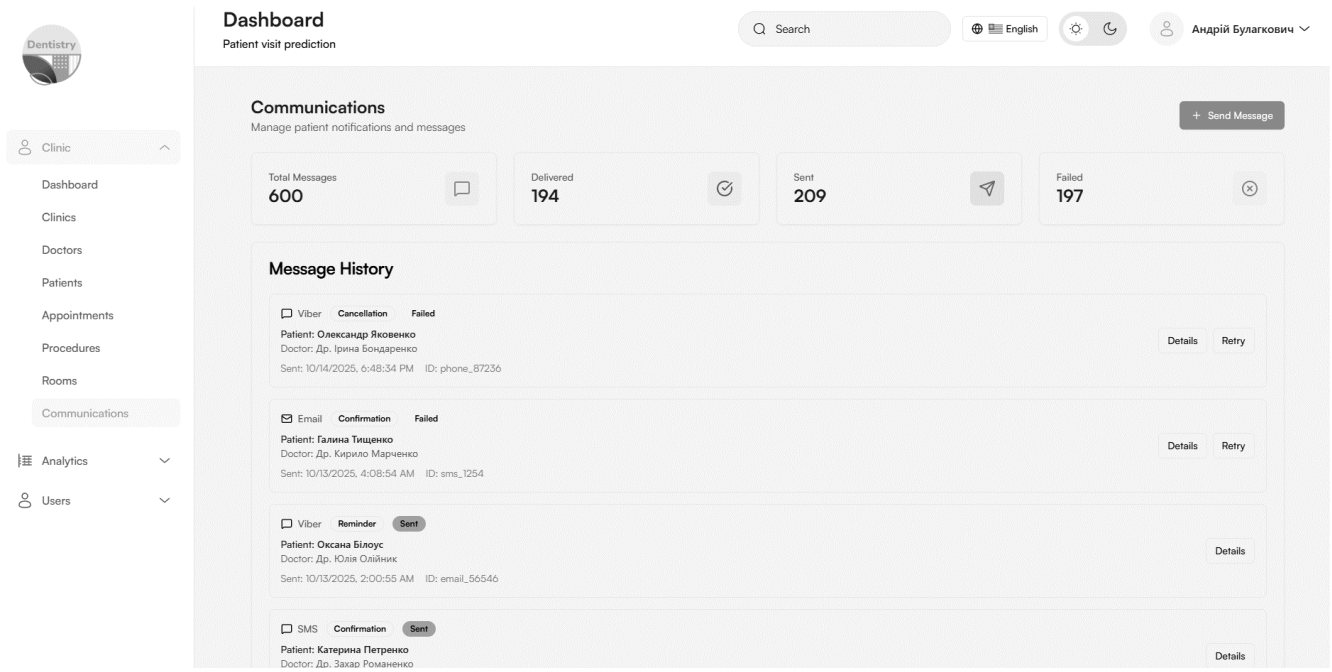



Рисунок 4.7 – Управління комунікаціями

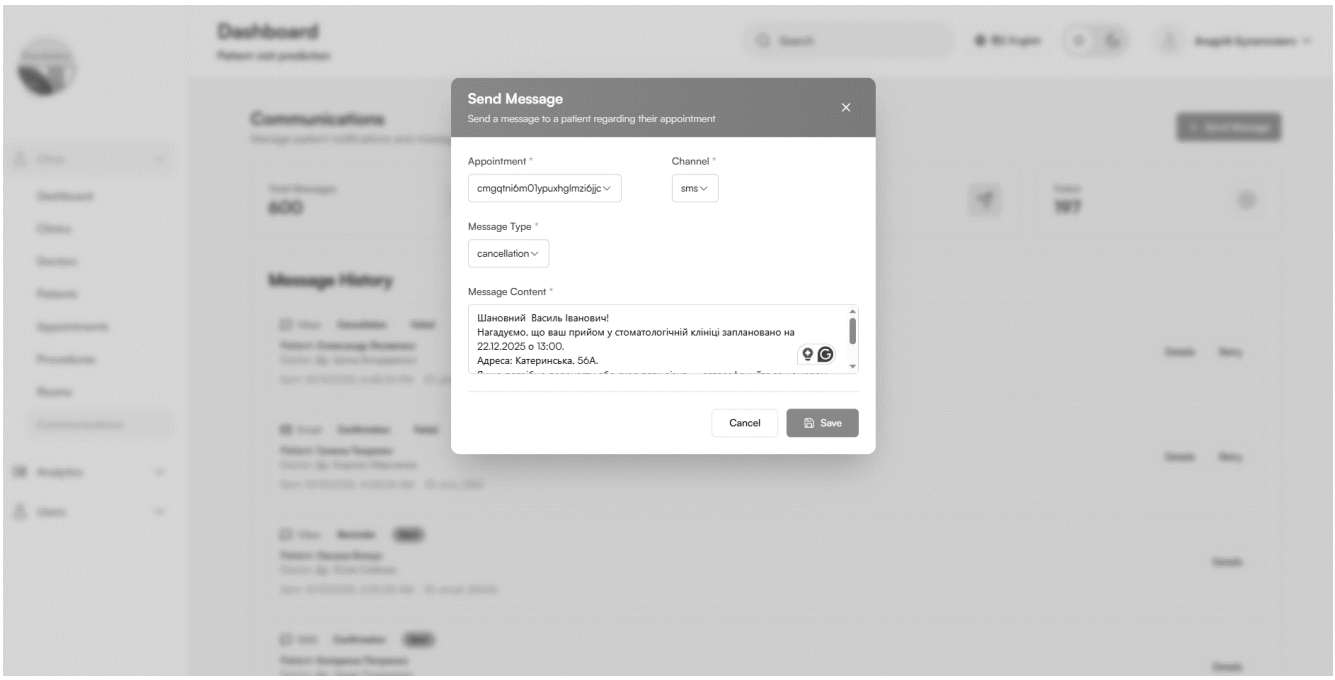



Рисунок 4.8 – Надсилання повідомлення пацієнту

У результаті було продемонстровано головні сторінки вебзастосунку, кожна з яких виконує важливу функцію в процесі управління роботою стоматологічної клініки.

## 4.2 Реалізація адміністративної панелі

Адміністративна частина застосунку побудована на NextAuth.js для управління сесіями та авторизації. Система ролей включає ADMIN (повний доступ до системи включаючи прогнозування), DOCTOR (доступ до пацієнтів та записів) та HEAD (обмежений доступ). Middleware застосунку забезпечує захищені маршрути з ролевою перевіркою та автоматичними перенаправленнями для неавторизованих користувачів. Next.js middleware реалізує комплексну систему контролю доступу. Система перевіряє авторизацію користувача та його роль перед доступом до захищених маршрутів. Неавторизовані користувачі автоматично перенаправляються на сторінку входу, а авторизовані користувачі з неправильними ролями перенаправляються на відповідні сторінки.

Адміністративна панель надає повноцінну систему управління користувачами з CRUD операціями. Сторінка списку користувачів включає пошук по імені, відображення ролей, статус активності з візуальними індикаторами та дії для редагування, видалення та зміни статусу користувачів. Система перевіряє унікальність поштових адрес та автоматично шифрує паролі з використанням bcrypt. Форма редагування користувача дозволяє оновлення всіх полів, включаючи зміну паролю. Система включає клієнтську валідацію з відображенням помилок у реальному часі та перевірку унікальності пошти серед інших користувачів.

API маршрути для управління користувачами реалізують повний набір CRUD операцій. GET маршрут повертає список всіх користувачів з фільтрацією по пошуковому запиту, POST маршрут створює нових користувачів з валідацією та хешуванням паролів, PUT маршрут оновлює існуючих користувачів з перевіркою унікальності email, DELETE маршрут видаляє користувачів з підтвердженням, а PATCH маршрут перемикає статус активності користувачів. Всі API маршрути включають обробку помилок, валідацію даних та безпечне повернення користувачів без паролів. Система автоматично оновлює поле updatedAt при змінах

та забезпечує цілісність даних через Prisma ORM. Всі операції з даними пацієнтів здійснюються з дотриманням принципів GDPR [27].

### 4.3 Інтеграція модуля прогнозування у вебзастосунку

Інтеграція функціоналу прогнозування відвідувань пацієнтів у вебзастосунок розглядається як складний інженерно-науковий процес, що об'єднує збір і підготовку даних, побудову і валідацію моделі машинного навчання [28], інтеграцію моделі з бекендом і фронтендом, а також організацію безпечного та масштабованого розгортання.

Принцип побудови прогнозного функціоналу базується на модульності: логіка підготовки даних та навчання моделі розміщена в окремому сервісі, який взаємодіє з основним бекендом через визначені API. Такий підхід дозволяє змінювати або замінювати модель без необхідності масштабних змін у фронтенді, а також забезпечує можливість виконувати операції навчання на відокремлених обчислювальних вузлах або в контейнерах. Джерелами даних виступають оперативні таблиці змісту записів на прийом у електронній системі клініки: для кожного запису доступні часові позначки початку та кінця прийому, ідентифікатори пацієнта, лікаря та клініки, статус запису та інші допоміжні поля, які можуть впливати на поведінку пацієнта. На етапі інтеграції відбувається уніфікація форматів, перевірка валідності часових позначок та відбір релевантних полів для подальшої обробки.

Особлива увага приділяється побудові ознак, оскільки саме набір і якість ознак визначають корінну здатність моделі вловлювати патерни поведінки. Часова інформація перетворюється на числові і бінарні ознаки, що дозволяє моделі враховувати добові й тижневі закономірності: витягуються годинна складова, день тижня, день місяця, місяць, а також формуються індикатори ранкового, післяобіднього та вечірнього інтервалів. Додатково обчислюється тривалість

прийому у хвилинах, що може корелювати із типом послуги і, опосередковано, з поведінкою пацієнта. Найважливішим компонентом інформаційного набору є історичні агрегати для сутностей «пацієнт», «лікар» і «клініка»: частка відвідувань у загальному числі записів, а також загальна кількість попередніх візитів. Ці агрегати дозволяють врахувати індивідуальні схильності та контекст, у якому здійснюється запис. Комбінація часових, поведінкових і контекстних ознак формує багатовимірний простір, у якому вибрана модель шукає статистичні закономірності.

Модель повинна розпізнавати кілька типів подій: чи пацієнт прийшов на прийом, не з’явився, скасував запис заздалегідь або скасував запізно. Для роботи з такими складними даними необхідне глибоке розуміння принципів машинного навчання [29]. Такий підхід дозволяє стоматологічній клініці гнучкіше реагувати на різні ситуації, адже для кожного типу події можна застосувати свої правила дій – наприклад, по-різному нагадувати пацієнтам про прийом або встановлювати різні політики скасування. В якості базового алгоритму обрано ансамблевий метод Random forest, що поєднує стійкість до викидів, здатність працювати з мішаними типами ознак і можливість інтерпретації через показники важливості ознак [30]. Навчальний процес передбачає поділ даних на тренувальну і тестову підвибірки із збереженням пропорцій класів, застосування ваг класів для компенсації дисбалансу та обчислення метрик якості – accuracy, precision, recall і F1 для кожного класу.

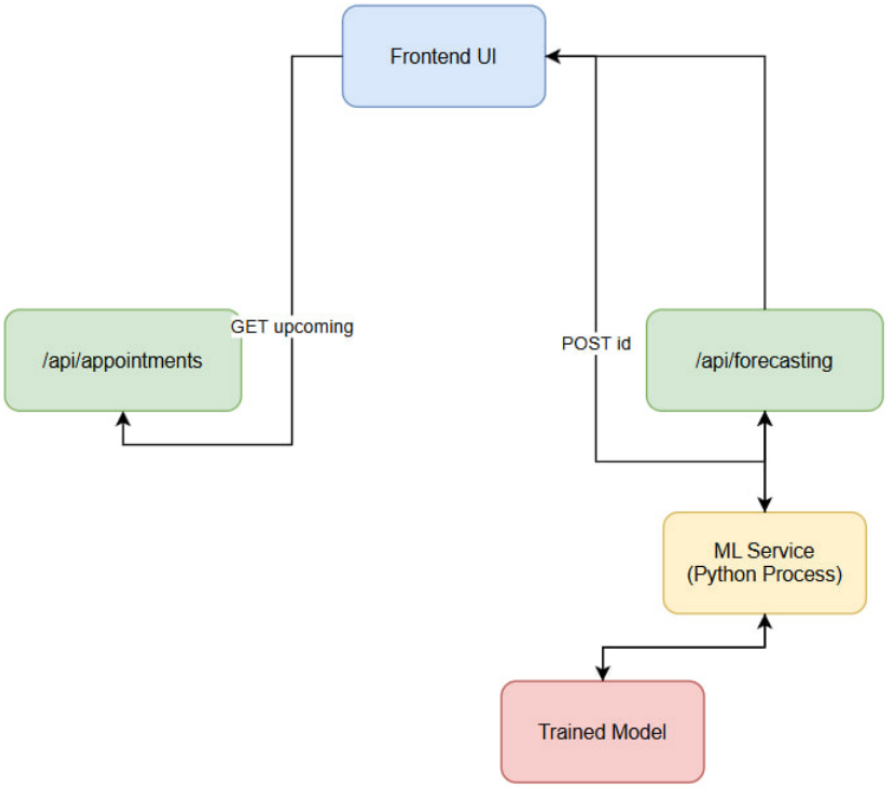



Рисунок 4.9 – Структура інтеграції прогнозного сервісу у вебзастосунок

Нижче наведено приклади програмних фрагментів, що відображають ключові етапи роботи обчислювального сервісу. Перший із них реалізує процес формування ознак та їх вилучення із початкових даних про записи пацієнтів.

Приклад функції підготовки ознак:

```
def prepare_features(appointments_data):
# Перетворення отриманих даних у DataFrame
df = pd.DataFrame(appointments_data)

# Конвертація часових полів у формат datetime
df['scheduledStart'] = pd.to_datetime(df['scheduledStart'])
df['scheduledEnd'] = pd.to_datetime(df['scheduledEnd'])

# Витягнення часових характеристик із дати початку прийому
df['hour'] = df['scheduledStart'].dt.hour
df['day_of_week'] = df['scheduledStart'].dt.dayofweek
```

```
df['day_of_month'] = df['scheduledStart'].dt.day
df['month'] = df['scheduledStart'].dt.month

# Формування бінарних ознак для типових часових інтервалів
df['is_weekend'] = df['day_of_week'].isin([5, 6]).astype(int)
df['is_morning'] = df['hour'].between(8, 12).astype(int)
df['is_afternoon'] = df['hour'].between(13, 17).astype(int)
df['is_evening'] = df['hour'].between(18, 20).astype(int)

# Обчислення тривалості прийому у хвилинах
df['duration_minutes'] = (df['scheduledEnd'] - df['scheduledStart']).dt.total_seconds() / 60

# Агреговані статистики для пацієнтів
patient_stats = df.groupby('patientId').agg({
'status': lambda x: (x == 'attended').sum() / len(x),  # частка відвіданих прийомів
'scheduledStart': 'count'                              # загальна кількість записів
}).rename(columns={
'status': 'patient_attendance_rate',
'scheduledStart': 'patient_total_appointments'
})
df = df.merge(patient_stats, left_on='patientId', right_index=True, how='left')

doctor_stats = df.groupby('doctorId').agg({
'status': lambda x: (x == 'attended').sum() / len(x),
'scheduledStart': 'count'
}).rename(columns={
'status': 'doctor_attendance_rate',
'scheduledStart': 'doctor_total_appointments'
})
df = df.merge(doctor_stats, left_on='doctorId', right_index=True, how='left')

clinic_stats = df.groupby('clinicId').agg({
```

```
'status': lambda x: (x == 'attended').sum() / len(x),
'scheduledStart': 'count'
}).rename(columns={
'status': 'clinic_attendance_rate',
'scheduledStart': 'clinic_total_appointments'
})
df = df.merge(clinic_stats, left_on='clinicId', right_index=True, how='left')

# Кодування категоріальних змінних
le_status = LabelEncoder()
le_source = LabelEncoder()
df['status_encoded'] = le_status.fit_transform(df['status'])
df['source_encoded'] = le_source.fit_transform(df['source'].fillna('unknown'))

# Формування підсумкового набору ознак
feature_columns = [
'hour', 'day_of_week', 'day_of_month', 'month',
'is_weekend', 'is_morning', 'is_afternoon', 'is_evening',
'duration_minutes',
'patient_attendance_rate', 'patient_total_appointments',
'doctor_attendance_rate', 'doctor_total_appointments',
'clinic_attendance_rate', 'clinic_total_appointments',
'source_encoded'
]
return df[feature_columns], df['status_encoded'], le_status, le_source
```

Для реалізації попередньої обробки даних використано методи машинного навчання з застосуванням мови програмування Python.

Приклад функції навчання моделі:

```
def train_model(appointments_data):
    try:
```

```
# Підготовка ознак та цільової змінної
# Функція prepare_features перетворює вхідні дані на матрицю ознак (X) і цільову змінну (y)
X, y, le_status, le_source = prepare_features(appointments_data)

# Поділ даних на тренувальну (80%) і тестову (20%) вибірки
# stratify=y – зберігає пропорції класів у вибірках
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Створення моделі випадкового лісу (Random Forest)
# n_estimators=100 – кількість дерев у моделі
# max_depth=10 – максимальна глибина кожного дерева (контроль складності)
# min_samples_split=5 – мінімальна кількість прикладів для поділу вузла
# min_samples_leaf=2 – мінімальна кількість прикладів у листку
# class_weight='balanced' – автоматично компенсує дисбаланс класів
model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42,
    class_weight='balanced'
)

# Навчання моделі на тренувальних даних
model.fit(X_train, y_train)

# Прогнозування на тестових даних
y_pred = model.predict(X_test)
```

```
        # Обчислення точності класифікації
        accuracy = accuracy_score(y_test, y_pred)

        # Формування звіту з основними метриками (precision, recall, f1-score) для кожного
класу
        report = classification_report(y_test, y_pred, output_dict=True)

        # Отримання важливості кожної ознаки у прийнятті рішень моделлю
        feature_importance = dict(zip(X.columns, model.feature_importances_))

        # Формування словника з усією корисною інформацією про модель
        model_data = {
            'model': model,                    # Навчена модель
            'label_encoder': le_status,
            'source_encoder': le_source,
            'feature_columns': X.columns.tolist(),   # Список ознак
            'accuracy': accuracy,                    # Точність моделі
            'classification_report': report,
            'feature_importance': feature_importance # Важливість ознак
        }
        # Повертаємо словник із результатами
        return model_data

    except Exception as e:
        # Якщо виникла помилка – повертаємо її опис
        return {'error': str(e)}
```

Приклад функції передбачення для одного запису:

```
def predict_attendance(model_data, appointment_features):
    try:
        # Отримуємо навчену модель та енкодер для статусів із словника model_data
```

```
        model = model_data['model']
        le_status = model_data['label_encoder']

        # Перетворюємо ознаки одного запису (словника) у DataFrame для подачі в модель
        # Модель очікує табличні дані навіть для одного прикладу
        X_pred = pd.DataFrame([appointment_features])

        # Робимо передбачення класу (наприклад: "відвідання", "неявка", "скасування вчасно", "скасування пізно")
        prediction = model.predict(X_pred)[0]

        # Отримуємо ймовірності належності до кожного класу
        # Наприклад: {'Відвідання': 0.7, 'Неявка': 0.2, 'Скасування пізно': 0.1}
        probabilities = model.predict_proba(X_pred)[0]

        # Перетворюємо числову мітку назад у текстову назву класу (розшифровуємо через енкодер)
        predicted_status = le_status.inverse_transform([prediction])[0]

        # Отримуємо список усіх можливих класів (порядок відповідає позиціям у масиві ймовірностей)
        status_classes = le_status.classes_

        # Створюємо словник з назв класів та їх ймовірностями
        probabilities_dict = dict(zip(status_classes, probabilities))

        # Повертаємо результат у зручному форматі:
        # - передбачений статус (рядок)
        # - ймовірності для всіх класів
        # - впевненість моделі (найвища ймовірність серед усіх класів)
        return {
            'predicted_status': predicted_status,
```

```
            'probabilities': probabilities_dict,
            'confidence': max(probabilities)
        }

    except Exception as e:
        # У випадку помилки (наприклад, відсутня ознака або неправильний тип даних)
        # повертаємо текст помилки у словнику
        return {'error': str(e)}
```

На сторінці Forecasting вебзастосунку передбачена кнопка Train Model. Після її натискання відбувається відправлення GET-запиту до відповідного API-ендпоінту, який обробляє запит на сервері. У результаті цього запиту сервер отримує всі дані про appointments із бази даних. Далі запускається Python-процес, який відповідає за навчання моделі прогнозування. У цьому процесі дані обробляються за допомогою бібліотек pandas та scikit-learn. На першому етапі відбувається підготовка ознак (features) для моделі: дані очищуються, перетворюються у потрібний формат і можуть створюватися додаткові змінні, що впливають на точність прогнозу.

Після підготовки даних відбувається навчання моделі машинного навчання на отриманих даних. Після завершення навчання модель використовується для розрахунку ключових метрик, які оцінюють її точність та ефективність у прогнозуванні майбутніх відвідувань пацієнтів.

Після завершення всіх цих кроків результати повертаються на сторінку вебзастосунку, де користувач може ознайомитися з прогнозами та оцінками моделі. Щоб забезпечити швидку роботу та уникнути повторного навчання при кожному зверненні, модель кешується на 24 години. Це дозволяє користувачам отримувати швидкі прогнози без додаткових обчислень.

Після закінчення цього періоду в 24 години для оновлення прогнозів необхідно повторно запустити процес навчання моделі.

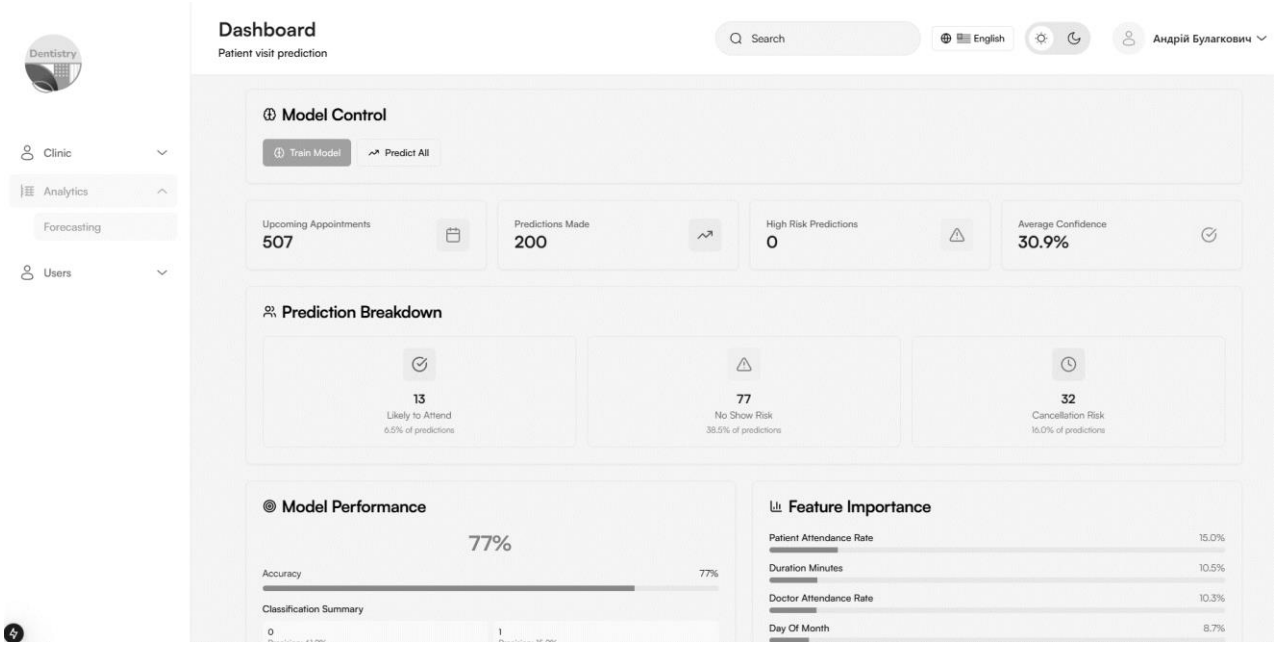



Рисунок 4.10 – Інтерфейс сторінки Forecasting

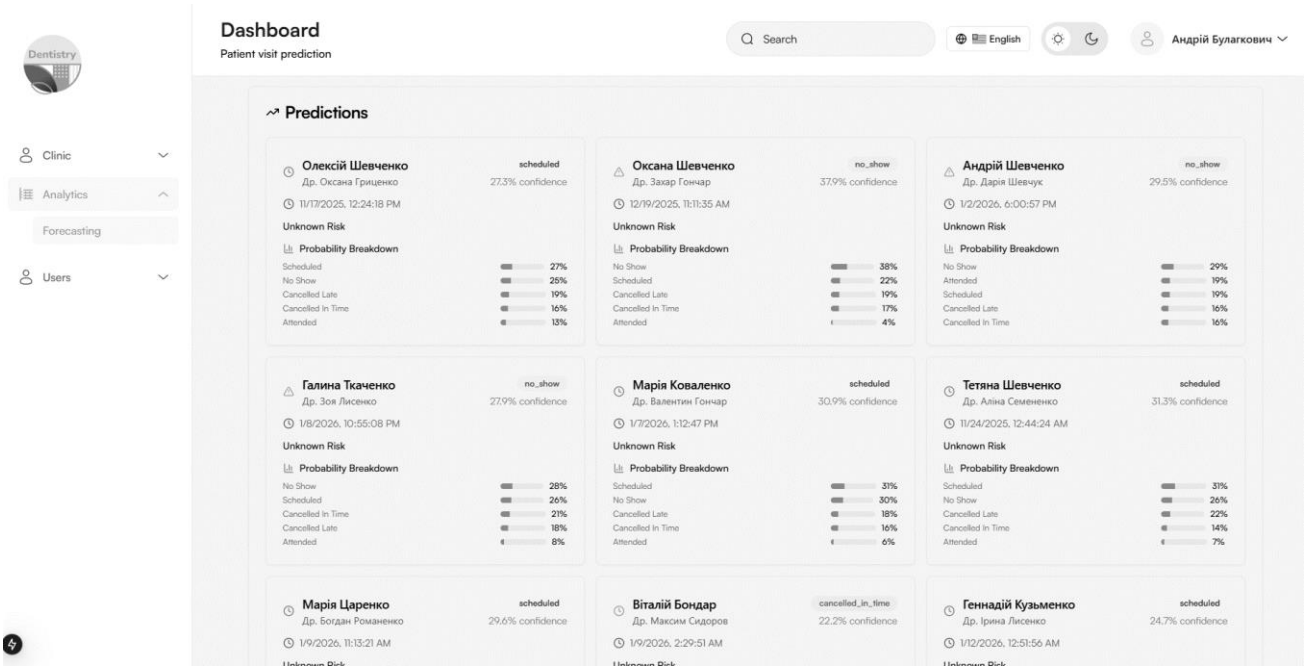



Рисунок 4.11 – Результат прогнозування відвідування пацієнтів

Таким чином було розроблено сторінку з прогнозуванням відвідувань пацієнтів у стоматологічній клініці, яка забезпечує інтеграцію машинного навчання з вебзастосунком, обробку даних, формування ознак і відображення прогнозів.

## 4.4 Тестування та оптимізація вебзастосунку

Тестування та оптимізація вебзастосунку стоматологічної клініки були спрямовані на підвищення продуктивності системи, зменшення часу відгуку API, а також на забезпечення ефективного використання ресурсів сервера та бази даних. Основна увага приділялася роботі з великими обсягами даних (клініки, лікарі, пацієнти, записи на прийом), де не оптимізовані запити призводили до затримок та надмірного навантаження на систему.

Метою тестування продуктивності (Performance testing) було:

1) перевірити час відгуку ключових API endpoints;
2) оцінити ефективність застосованих оптимізацій;
3) забезпечити масштабованість системи та комфортний користувацький досвід.

Тестування виконувалося за допомогою спеціального скрипта performance-test.js, який автоматично відправляє серію GET-запитів до API endpoints та вимірює час їх обробки (рис. 4.12). Кожен тест включав 10 ітерацій запитів, а також вимірював середній час відгуку, мінімальний та максимальний час, а також обсяг переданих даних.

```
const BASE_URL = process.env.BASE_URL || 'http://localhost:3000';

const TESTS = [
  { name: 'Clinics Endpoint', url: '/api/clinics', method: 'GET', iterations: 10,
      expectedMaxTime: 1000 },
  { name: 'Doctors Endpoint', url: '/api/doctors?page=1&limit=50', method: 'GET', iterations:
      10, expectedMaxTime: 500 },
  { name: 'Patients Endpoint', url: '/api/patients?page=1&limit=50', method: 'GET',
      iterations: 10, expectedMaxTime: 500 },
  { name: 'Appointments Endpoint', url: '/api/appointments?page=1&limit=50', method: 'GET',
      iterations: 10, expectedMaxTime: 600 },
];
```

Рисунок 4.12 – Вимірювання часу обробки запитів

Для вимірювання продуктивності на стороні сервера використовувалося додавання заголовка X-Response-Time у відповіді API (рис. 4.13).

```
export async function GET() {
  const startTime = performance.now();

  // Виконання запиту до бази даних
  const data = await prisma.clinic.findMany({ select: { id: true, name: true } });

  const endTime = performance.now();
  const duration = endTime - startTime;

  const response = NextResponse.json(data);
  response.headers.set('X-Response-Time', `${duration.toFixed(2)}ms`);

  return response;
}
```

Рисунок 4.13 – Вимірювання продуктивності на стороні сервера

Це дозволяло відстежувати час виконання запиту безпосередньо на сервері та отримувати статистику для подальшого аналізу.

Використаємо спеціальне поле _count в Prisma ORM з використанням повного підрахунку пов’язаних записів, що підвищує продуктивність (рис. 4.14).

```
const clinics = await prisma.clinic.findMany({
  select: {
    id: true,
    name: true,
    _count: {
      select: {
        doctors: true,
        patients: true,
        rooms: true,
        appointments: true,
      },
    },
  },
});
```

Рисунок 4.14 – Використання спеціального поля _count

Такий підхід дозволяє обмежитися лише підрахунком лікарів, пацієнтів та кабінетів, без повного завантаження їхніх записів, що значно зменшує обсяг переданої інформації.

Також додано пагінацію для завантажування тільки необхідні сторінки даних, що зменшує навантаження на базу даних та швидко повертає результат користувачеві (рис 4.15).

```
const page = parseInt(searchParams.get("page") || "1");
const limit = parseInt(searchParams.get("limit") || "50");
const skip = (page - 1) * limit;

const [patients, total] = await Promise.all([
  prisma.patient.findMany({
    where: whereClause,
    select: { firstName: true, lastName: true },
    skip,
    take: limit,
    orderBy: { createdAt: "desc" },
  }),
  prisma.patient.count({ where: whereClause }),
]);
```

Рисунок 4.15 – Додавання пагінації для завантаження необхідних сторінок

Після оптимізації результати значно покращилися. Результати наведено у табл. 4.1.

Таблиця 4.1 – Вплив оптимізації на продуктивність вебзастосунку

| Endpoint | До оптимізації | Після оптимізації | Покращення |
|---|---|---|---|
| Clinics | 2500 ms | 150 ms | ~94 % |
| Doctors | 1200 ms | 180 ms | ~85 % |
| Patients | 1800 ms | 220 ms | ~88 % |
| Appointments | 2500 ms | 280 ms | ~89 % |

Обсяг переданих даних зменшився у 10 разів, а навантаження на базу даних скоротилося приблизно на 70%. Система стала масштабованою та готовою до обробки великої кількості запитів.

**Висновки до розділу 4**

У четвертому розділі розглянуто програмну реалізацію вебзастосунку стоматологічної клініки, включно з клієнтською та адміністративною частинами, інтеграцією прогнозного функціоналу та оптимізацією продуктивності.

Клієнтська частина побудована на Next.js 15 із TypeScript та Tailwind, забезпечуючи адаптивний інтерфейс, модульність та інтернаціоналізацію.

Адміністративна панель реалізована з доступом через NextAuth.js і middleware Next.js, забезпечуючи безпеку та ефективне управління даними через Prisma ORM.

Інтеграція прогнозування відвідувань реалізована через модульну архітектуру із сервісами підготовки ознак, навчання моделі та прогнозування. Використання Random Forest дозволяє передбачати кілька типів подій, а сторінка Forecasting відображає результати з кешуванням моделей для швидкого доступу.

Оптимізація вебзастосунку зменшила час відгуку API та навантаження на базу даних, підвищивши продуктивність ключових endpoints до 94 %.

Таким чином, розроблений вебзастосунок поєднує сучасні технології, безпечну адміністративну панель, прогнозування відвідувань та високу продуктивність, забезпечуючи ефективне управління стоматологічною клінікою.

# ВИСНОВКИ

У ході виконання кваліфікаційної магістерської роботи було розроблено вебзастосунок для стоматологічної клініки з функцією прогнозування відвідування пацієнтів. Реалізоване рішення поєднує сучасні технології веброзробки, методи машинного навчання та принципи безпечної роботи з даними, що дозволило створити систему для автоматизації процесів і підвищення ефективності стоматологічної клініки.

У першому розділі обґрунтовано актуальність теми, визначено мету, завдання, об’єкт і предмет дослідження. Проблема неявки пацієнтів на прийом є суттєвою для медичних закладів, оскільки спричиняє втрату часу лікарів і зниження продуктивності. Впровадження рішення з прогнозуванням дозволяє зменшити кількість пропусків, покращити розклад і комунікацію з пацієнтами. Розглянуто архітектурні підходи та питання безпеки: використання хешування паролів, контроль доступу до адміністративних функцій і дотримання вимог законодавства щодо захисту персональних даних. Обраний технологічний забезпечує стабільність, масштабованість і зручність використання.

У другому розділі проведено аналіз предметної області й створено вимоги до системи. Визначено функціональні можливості вебзастосунку та склад даних для навчання прогнозної моделі. Датасет обсягом 3000 записів і 11 ознак характеризує поведінку пацієнтів, типи процедур, часові параметри та наявність нагадувань. Початковий аналіз даних засвідчив, що вибірка є збалансованою та не містить аномальних значень. Порівняння моделей машинного навчання показало, що найкращі результати продемонстрував алгоритм RandomForestClassifier, який забезпечив високу точність прогнозування завдяки поєднанню багатьох дерев рішень. Саме його інтегровано в систему як основний механізм прогнозування.

У третьому розділі виконано проєктування архітектури вебзастосунку. Створено діаграми варіантів використання, станів і класів, що описують взаємодію

користувачів, життєвий цикл записів і структуру бази даних. Було розроблено реляційну базу даних, що містить чотирнадцять таблиць, які охоплюють усі сутності предметної області: пацієнтів, лікарів, процедури, записи та прогнози. Така структура забезпечує логічну цілісність і можливість масштабування. Архітектура системи побудована за принципом клієнт-сервер з чітким розмежуванням відповідальності між компонентами, що створює надійну основу для інтеграції аналітичних і прогнозних модулів.

У четвертому розділі розглянуто програмну реалізацію вебзастосунку. Клієнтська частина створена з використанням Next.js 15, TypeScript, Tailwind CSS, що забезпечує адаптивний інтерфейс і модульність. Адміністративна панель реалізована через NextAuth.js і middleware Next.js для безпечної автентифікації та зручного управління даними через Prisma ORM. Інтеграція прогнозного функціоналу реалізована за модульною структурою: підготовка ознак, навчання моделі та прогнозування. Використання Random Forest дозволило передбачати ймовірність неявки пацієнтів, а сторінка Forecasting відображає результати з кешуванням моделей. Оптимізація продуктивності зменшила час відгуку API до 94 %, підтвердивши ефективність технічних рішень.

Підсумовуючи результати, можна стверджувати, що поставлена мета створення вебзастосунку стоматологічної клініки з функцією прогнозування відвідувань досягнута повністю. Реалізовано повний цикл розробки: від аналізу предметної області та проєктування, до створення клієнтської частини, навчання моделі й інтеграції її у вебзастосунок. Створений застосунок поєднує управління розкладом, базою пацієнтів, аналітикою та прогнозуванням відвідувань, забезпечуючи ефективну підтримку роботи стоматологічної клініки.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Задача прогнозування. URL: https://studfile.net/preview/7818681/ (дата звернення: 12.09.2025).

2. Види і особливості стоматологічних послуг. URL: https://danti.kiev.ua/uk/statti/vidi-i-osoblivosti-stomatologichnikh-poslug (дата звернення: 15.09.2025).

3. Правова інформація про надання медичних стоматологічних послуг. URL: https://stomatologia.zp.ua/uk/pravovaya-informaciya (дата звернення: 17.09.2025).

4. Основи законодавства України про охорону здоров’я : Закон України від 19.11.1992 № 2801-XII. URL: https://zakon.rada.gov.ua/laws/show/2801-12 (дата звернення: 20.09.2025).

5. ISO/IEC 27001:2013. Information technology – Security techniques – Information security management systems – Requirements. Geneva : ISO, 2013. 46 p.

6. Електронна система охорони здоров’я. URL: https://moz.gov.ua/uk/elektronna-sistema-ohoroni-zdorovya (дата звернення: 03.10.2025).

7. Опанасюк Ю. З. Світ сучасної стоматології. Київ: ТОВ Видавничо-інформаційний центр, 2005. 507с.

8. Світові тренди цифровізації сфери охорони здоров’я та принципи реалізації. URL: http://customs-admin.umsf.in.ua/archive/2023/1/8.pdf (дата звернення: 10.10.2025).

9. World Health Organization. Digital health. Geneva : WHO, 2023. URL: https://www.who.int/health-topics/digital-health (дата звернення: 17.10.2025).

10. A New Era of Dental Care: Harnessing Artificial Intelligence for Better Diagnosis and Treatment. URL: https://pubmed.ncbi.nlm.nih.gov/38143639/ (дата звернення: 20.10.2025).

11. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd ed. Sebastopol : O'Reilly Media, 2019. 856 p.

12. Research and application of artificial intelligence in dentistry from lower-middle income countries – a scoping review. URL: https://bmcoralhealth.biomedcentral.com/articles/10.1186/s12903-024-03970-y (дата звернення: 26.10.2025).

13. Стоматологічна інформаційна система Dentalogic. 2025. URL: https://dentalogic.com.ua/ (дата звернення: 27.10.2025).

14. Класифікація законодавства у сфері охорони здоров'я. URL: https://www.juris.vernadskyjournals.in.ua/journals/2019/6_2019/4.pdf (дата звернення: 28.10.2025).

15. Юзевич В. М., Мисюк Р. В., Шувар Р. Я. Основи аналітики даних у Python. Львів: ЛНУ ім. І. Франка, 2024. 70с.

16. Візуалізація даних. URL: https://datawiz.io/uk/blog/data-visualization-examples-of-retail-applications (дата звернення: 01.11.2025).

17. Decision Tree Classification in Python Tutorial. URL: https://www.datacamp.com/tutorial/decision-tree-classification-python (дата звернення: 03.11.2025).

18. K-Nearest Neighbors (KNN) Classification with scikit-learn. URL: https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn (дата звернення: 05.11.2025).

19. Random Forest Classification in Python With Scikit-Learn. URL: https://www.datacamp.com/tutorial/random-forests-classifier-python (дата звернення: 07.11.2025).

20. Davari A. Machine Learning: Understand Neural Networks And What Powers Them Up. 2022. 88 p.

21. Eknath, Prof. Upasani Dhananjay, 1st, Shete, Prof. Virendra Virbhadra, 1st. Machine Learning with Python: Machine Learning with Python. INSC International Publisher (IIP), 2021.

22. PostgreSQL Tutorial. URL: https://www.w3schools.com/postgresql/ (дата звернення: 09.11.2025).

23. The Prisma ORM: A Brief Overview and Introduction. URL: https://dev.to/sandrockjustin/the-prisma-orm-a-brief-overview-and-introduction-353m (дата звернення: 09.11.2025).

24. Next.js Tutorial. URL: https://www.geeksforgeeks.org/nextjs/nextjs-tutorial/ (дата звернення: 10.11.2025).

25. UML Use Case Diagram Tutorial | Lucidchart. URL: https://www.lucidchart.com/pages/tutorial/uml-use-case-diagram (дата звернення: 12.11.2025).

26. Rumpe B. Sequence Diagrams. Modeling with UML. Cham, 2016. P. 191–208. URL: https://link.springer.com/chapter/10.1007/978-3-319-33933-7_6 (дата звернення: 14.11.2025).

27. General Data Protection Regulation (GDPR). Regulation (EU) 2016/679. EUR-Lex. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (дата звернення: 15.11.2025).

28. Raschka S., Mirjalili V. Python Machine Learning. 3rd ed. Birmingham : Packt Publishing, 2019. 770 p.

29. Chatterjee I. Machine Learning and Its Application: A Quick Guide for Beginners. Bentham Science Publishers, 2022. 476 p.

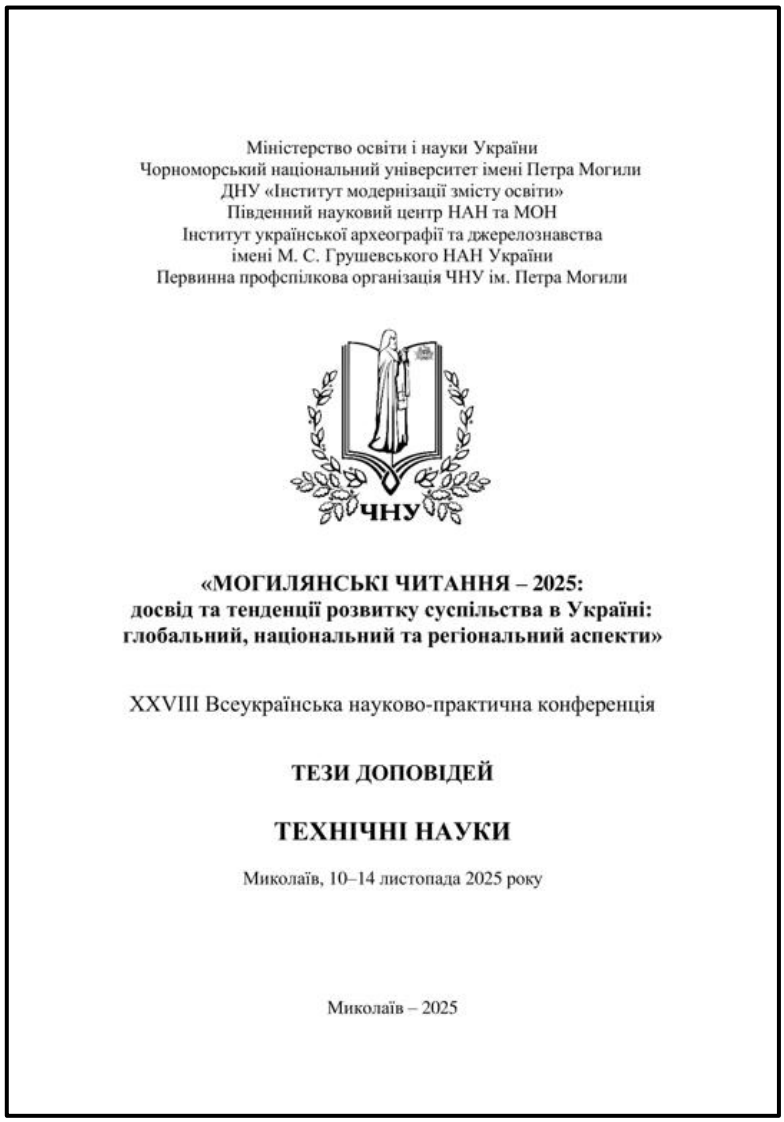
30. Breiman L. Random Forest. Machine Learning. 2001. Vol. 45, № 1. P. 5–32. DOI: 10.1023/A:1010933404324.

# ДОДАТОК А

## Апробація кваліфікаційної магістерської роботи

Результати досліджень були представлені на конференції:

– Могилянські читання – 2025, : Досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти : XXVIII Всеукр. наук.-практ. конф. : тези доповідей : Комп'ютерні науки. Тенхічні науки, Миколаїв, 10-14 листоп. 2025 р. / ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2025.



Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2025:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVIII Всеукраїнська науково-практична конференція

**ТЕЗИ ДОПОВІДЕЙ**

**ТЕХНІЧНІ НАУКИ**

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

Рисунок А.1 – Апробація

**УДК 004.42**

***Тебенко А. В.***
*здобувач другого (магістерського) рівня вищої освіти*
*освітньої програми «Інженерія програмного забезпечення»,*
***Давиденко Є. О.***
*канд. техн. наук, доцент, завідувач кафедри інженерії програмного забезпечення*
*ЧНУ імені Петра Могили, м. Миколаїв, Україна*

**РОЗРОБКА ЗАСТОСУНКУ СТОМАТОЛОГІЧНОЇ КЛІНІКИ З ФУНКЦІЄЮ ПРОГНОЗУВАННЯ ВІДВІДУВАННЯ ПАЦІЄНТІВ**

Рисунок А.2 – Тези

# ДОДАТОК Б

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score, classification_report, confusion_matrix,
    roc_auc_score, precision_recall_curve, f1_score
)
from sklearn.metrics import make_scorer

df = data_patients.copy()
X = df.drop(columns=["no_show"])
y = df["no_show"]

num_cols = X.select_dtypes(include=["int64","float64","bool"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object","category"]).columns.tolist()

numeric_tf = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="median")),
])
categorical_tf = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="most_frequent")),
    ("oh", OneHotEncoder(handle_unknown="ignore"))
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_tf, num_cols),
        ("cat", categorical_tf, cat_cols),
    ]
)

rf = RandomForestClassifier(
    n_estimators=300,
    max_depth=None,
    n_jobs=-1,
    random_state=42,
    class_weight="balanced"
)
```

```
pipe = Pipeline(steps=[
   ("prep", preprocess),
   ("model", rf)
])

X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=42, stratify=y
)

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
y_prob = pipe.predict_proba(X_test)[:, 1]

print("Точність (RF, p>=0.5):", round(accuracy_score(y_test, y_pred), 3))
print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 3))
print("Матриця змішувань:\n", confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=3))

prec, rec, thr = precision_recall_curve(y_test, y_prob)
f1s = 2 * (prec * rec) / (prec + rec + 1e-12)
best_idx = np.nanargmax(f1s)
best_thr = thr[best_idx] if best_idx < len(thr) else 0.5

y_pred_opt = (y_prob >= best_thr).astype(int)
print("\n=== Поріг оптимізовано під F1(клас 1) ===")
print("Оптимальний поріг:", round(float(best_thr), 3))
print("Accuracy:", round(accuracy_score(y_test, y_pred_opt), 3))
print("F1 (клас 1):", round(f1_score(y_test, y_pred_opt), 3))
print("Матриця змішувань:\n", confusion_matrix(y_test, y_pred_opt))
print(classification_report(y_test, y_pred_opt, digits=3))

param_grid = {
   "model__n_estimators": [200, 300, 500],
   "model__max_depth": [None, 6, 10, 20],
   "model__min_samples_leaf": [1, 2, 5],
   "model__max_features": ["sqrt", "log2", 0.5],
}
grid = GridSearchCV(
   pipe,
   param_grid,
   cv=5,


n_jobs=-1,
   scoring=make_scorer(f1_score, pos_label=1),
```

```
    verbose=0
)
grid.fit(X_train, y_train)

print("\n=== GridSearchCV (F1 для класу 1) ===")
print("Найкращі параметри:", grid.best_params_)
print("Найкращий CV F1(1):", round(grid.best_score_, 3))

best = grid.best_estimator_
y_prob_best = best.predict_proba(X_test)[:, 1]
prec_b, rec_b, thr_b = precision_recall_curve(y_test, y_prob_best)
f1s_b = 2 * (prec_b * rec_b) / (prec_b + rec_b + 1e-12)
best_idx_b = np.nanargmax(f1s_b)
best_thr_b = thr_b[best_idx_b] if best_idx_b < len(thr_b) else 0.5
y_pred_best = (y_prob_best >= best_thr_b).astype(int)

print("\nПоріг (найкраща модель):", round(float(best_thr_b), 3))
print("Accuracy:", round(accuracy_score(y_test, y_pred_best), 3))
print("F1 (клас 1):", round(f1_score(y_test, y_pred_best), 3))
print("Матриця змішувань:\n", confusion_matrix(y_test, y_pred_best))
print(classification_report(y_test, y_pred_best, digits=3))
```