

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

**«Інформаційна система автоматизації роботи
приватної поліклініки»**

Виконала студентка групи ІІЗ-22-3
КОХАН Тетяна Олегівна

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ФАНТ Микола Олександрович

Житомир – 2026

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: КОХАН Тетяна Олегівна

Керівник роботи: ЧИЖМОТРЯ Олена Геннадіївна

Тема роботи: «Інформаційна система автоматизації роботи приватної поліклініки»,

затверджена Наказом закладу вищої освіти від «27» лютого 2026 р., № 92/с

Вихідні дані для роботи: процеси автоматизації роботи приватної поліклініки в частині бронювання прийомів та обліку відвідувань; веборієнтовані технології розробки інформаційних систем.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	01.03.2026	01.03.2026
2	Чижмотря О.Г.	01.03.2026	01.03.2026
3	Чижмотря О.Г.	01.03.2026	01.03.2026

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з інформаційної системи автоматизації роботи приватної поліклініки та пояснювальної записки. Пояснювальна записка до випускної роботи складається зі вступу, 3-х розділів, висновків, списку використаних джерел із 30 найменувань, додатків, 26 таблиць та 36 ілюстрацій по тексту. Повний обсяг роботи становить 92 сторінки, з них 77 сторінок основного тексту.

Метою роботи є розробка та впровадження веборієнтованої інформаційної системи автоматизації ключових процесів приватної поліклініки, зокрема онлайн-бронювання прийомів, ведення електронного обліку відвідувань та керування інформаційними ресурсами й користувачами системи.

У роботі сформульовано вимоги до функціональних характеристик системи, організації вхідних і вихідних даних, часових характеристик і надійності функціонування. Розроблено варіанти використання системи, які передбачають розмежування прав доступу для різних категорій користувачів (адміністратор, лікар, пацієнт), об'єктно-орієнтовану структуру та логічну модель бази даних із визначенням основних сутностей і зв'язків між ними.

Результатом роботи є розроблена інформаційна система з інтуїтивно зрозумілим інтерфейсом, який автоматизує рутинні операції реєстратури, зменшує ймовірність помилок при записі на прийом та покращує загальну ефективність управління медичним закладом. Практична цінність розробки підтверджується можливістю її впровадження у діяльність реальних приватних поліклінік для підвищення якості обслуговування.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ, ОНЛАЙН-ЗАПИС, ПРИВАТНА ПОЛІКЛІНІКА, ІНТЕРФЕЙС.

					ІПЗ.КР.Б-121-26-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кохан Т.О.			Інформаційна система автоматизації роботи приватної поліклініки	Літ.	Арк.	Аркуші
Керівник		Чижевотря О.Г.					3	92
						Житомирська політехніка, група ІПЗ-22-3		
Рецензент		Фант М.О.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of an information system for automating the operations of a private polyclinic and an explanatory note. The explanatory note to the graduation thesis consists of an introduction, 3 chapters, conclusions, a list of 30 references, appendices, 26 tables, and 36 illustrations throughout the text. The total volume of the thesis is 92 pages, including 77 pages of the main text.

The purpose of the work is the development and implementation of a Weboriented information system for automating the key processes of a private polyclinic, namely online appointment booking, maintaining electronic records of visits, and managing the system's information resources and users.

The work formulates requirements for the functional characteristics of the system, the organization of input and output data, timing characteristics, and operational reliability. Use cases have been developed that provide for differentiated access rights for various user categories (administrator, doctor, patient), an object-oriented structure, and a logical database model defining the primary entities and relationships between them.

The result of the work is a developed information system with an intuitive interface that automates routine registry operations, reduces the likelihood of errors in appointment scheduling, and improves the overall management efficiency of the medical facility. The practical value of the development is confirmed by the possibility of its implementation in the activities of real private clinics to improve the quality of service.

KEYWORDS: INFORMATION SYSTEM, AUTOMATION, ONLINE APPOINTMENT BOOKING, PRIVATE POLYCLINIC, INTERFACE.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ПРИВАТНОЇ ПОЛІКЛІНІКИ ..	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури інформаційної системи.....	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	21
Висновки до першого розділу.....	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ РОБОТИ ПРИВАТНОЇ ПОЛІКЛІНІКИ	27
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	27
2.2. Розробка бази даних системи.....	38
2.3. Проєктування та реалізація алгоритмів роботи системи	46
2.4. Реалізація функціоналу системи.....	50
Висновки до другого розділу	55
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ АВТОМАТИЗАЦІЇ РОБОТИ ПРИВАТНОЇ ПОЛІКЛІНІКИ ..	57
3.1. Порядок встановлення та налаштування параметрів системи.	57
3.2. Структура інтерфейсу	60
3.3. Тестування роботи.....	74
Висновки до третього розділу.....	78
ВИСНОВКИ.....	80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	82
ДОДАТКИ.....	85

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД — база даних.

ІС — інформаційна система.

ПЗ – програмне забезпечення.

СКБД — система керування базами даних.

CSS (Cascading Style Sheets) — мова стилів для оформлення вебсторінок.

HTML (HyperText Markup Language) — мова розмітки для створення вебсторінок.

JavaScript — динамічна мова програмування, що виконується у браузері на стороні клієнта.

MySQL — система керування реляційними базами даних з відкритим вихідним кодом.

MVC (Model-View-Controller) — архітектурний шаблон розробки програмного забезпечення, що розділяє програму на три взаємопов'язані компоненти: модель (дані/логіка), вигляд (інтерфейс) та контролер (обробка запитів).

PHP — мова програмування з відкритим вихідним кодом, що виконується на стороні сервера та генерує динамічний вебвміст.

UML (Unified Modeling Language) — уніфікована мова моделювання.

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

ВСТУП

Активна цифровізація управлінських процесів зумовлює стрімке впровадження інформаційних технологій у важливі сфери, серед яких охорона здоров'я посідає одне з ключових місць. Сучасні медичні заклади, особливо приватного сектору, працюють в умовах високої конкуренції, що вимагає постійного покращення внутрішніх процесів, підвищення якості обслуговування та ефективного використання кадрових ресурсів. Однією з актуальних проблем є організація ефективного запису пацієнтів на прийом до лікаря та ведення обліку відвідувань у медичному закладі.

Традиційні методи адміністрування, що базуються на паперових носіях або телефонній комунікації, сьогодні є малоефективними. Вони призводять до значних витрат часу персоналу, помилок та накладок у розкладі лікарів, що негативно впливає на рівень задоволеності пацієнтів. Тому розробка та впровадження інформаційних систем автоматизації стає необхідною умовою для створення зручного та оперативного механізму взаємодії між закладом і пацієнтом. Такі системи забезпечують зручний онлайн-запис до лікаря, знижують навантаження на реєстратуру. Сучасні веборієнтовані інформаційні системи дозволяють користувачам отримувати доступ до послуг незалежно від місця перебування та підвищують оперативність обробки інформації.

Метою роботи є проектування та розробка інформаційної системи автоматизації роботи приватної поліклініки, яка забезпечує зручний онлайн-запис пацієнтів на прийом до лікаря та ведення обліку відвідувань у медичному закладі. Розроблювана система отримала назву «VITO» (від лат. Vita — життя), що символізує її спрямованість на турботу про здоров'я та якість життя пацієнтів.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати процеси запису пацієнтів на прийом та обліку відвідувань у медичному закладі, а також можливості їх автоматизації;
- визначити архітектуру та структуру веборієнтованої системи;
- обґрунтувати та вибрати технології для розробки системи;

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- розробити структуру бази даних та інформаційне забезпечення системи;
- розробити інтерфейс користувача для різних категорій користувачів (адміністратор, лікар, пацієнт);
- реалізувати систему розмежування прав доступу для користувачів;
- протестувати розроблену систему та перевірити її працездатність.

Об'єкт дослідження — процеси організації запису пацієнтів на прийом до лікаря та обліку відвідувань у приватній поліклініці.

Предмет дослідження — веборієнтована інформаційна система автоматизації роботи приватної поліклініки.

У роботі використано методи системного аналізу для дослідження процесів роботи поліклініки, методи об'єктно-орієнтованого проєктування для побудови структури системи та бази даних, а також методи вебпрограмування і проєктування інтерфейсу користувача для реалізації програмного продукту.

Практичне значення одержаних результатів полягає у створенні готового до впровадження програмного продукту, який автоматизує ключові процеси роботи приватної поліклініки. Впровадження системи сприяє скороченню часу на обробку звернень пацієнтів, мінімізації людського фактору при організації прийомів та підвищенню прозорості адміністративних процесів.

Крім того, система забезпечує зручний доступ до інформації для всіх категорій користувачів, що в цілому позитивно впливає на ефективність роботи медичного закладу.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

**РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ
ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ
ПРИВАТНОЇ ПОЛІКЛІНІКИ**

1.1. Постановка задачі

За даними Міністерства охорони здоров'я України, у 2024 році кількість електронних медичних записів в системі eHealth досягла 3,7 млрд, що у понад п'ять разів перевищує показник 2022 року [1]. Це свідчить про стрімке зростання попиту на цифрові медичні сервіси в Україні. Проте, незважаючи на загальну тенденцію до цифровізації, значна частина приватних медичних закладів досі використовує застарілі методи організації роботи.

Існуючі підходи до організації запису пацієнтів на прийом у медичних закладах, що базуються на телефонному обслуговуванні та паперовій документації, не забезпечують належного рівня ефективності. Вони спричиняють помилки в розкладі, дублювання записів, ускладнюють контроль відвідуваності та створюють додаткове навантаження на адміністративний персонал.

Створення інформаційної системи ставиться за мету не просто замінити паперові носії на цифрові, а об'єднати розрізнені процеси — запис, облік та керування — в єдиний інтегрований механізм. Це необхідно для досягнення таких цілей:

1. Ручне введення даних неминуче супроводжується помилками — неправильно записане прізвище, пропущений запис або одразу два пацієнти, призначені на одну й ту саму годину до одного лікаря. Система усуває цю проблему шляхом автоматичної перевірки коректності введених даних та блокування будь-яких спроб подвійного запису, що суттєво підвищує надійність і точність ведення медичної документації.

2. Необхідність телефонувати до реєстратури або особисто приходити до клініки лише для того, щоб домовитися про прийом, є незручною як для пацієнтів, так і для персоналу. Система вирішує цю

проблему, надаючи пацієнтам цілодобовий доступ до актуального розкладу лікарів: вони можуть самостійно обрати зручний час та записатися на прийом.

3. Планування графіків роботи лікарів вручну — це трудомісткий процес, який легко призводить до накладок і плутанини, особливо коли лікарів у закладі багато. Завдяки системі адміністратор керує розкладами в єдиному інтерфейсі і витрачає на організаційну роботу значно менше часу, який натомість можна спрямувати на безпосереднє обслуговування пацієнтів.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

1. Необхідно детально дослідити та описати процеси роботи приватної поліклініки, зокрема визначити, як різні категорії користувачів взаємодіють із системою. Пацієнти взаємодіють із системою під час пошуку лікаря та запису на прийом, лікарі — під час перегляду свого розкладу та ведення обліку пацієнтів, а адміністратори — під час керування записами та налаштування системи.

2. База даних є фундаментом усієї системи, тому її структура має бути продуманою з самого початку. Необхідно визначити основні сутності — лікарі, пацієнти, графіки роботи, записи візитів, новини — і встановити між ними коректні зв'язки. Неправильно спроектована база даних може стати причиною втрати даних, повільної роботи або складнощів із подальшим розширенням системи, тому цьому етапу слід приділити особливу увагу.

3. Вибір технологій безпосередньо впливає на те, наскільки зручно буде розробляти систему, як стабільно вона працюватиме і чи можна буде її розширювати в майбутньому. Мова програмування та система керування базами даних мають бути обрані з урахуванням специфіки медичного закладу: система повинна однаково добре працювати на різних пристроях, бути нескладною у впровадженні та обслуговуванні, а також мати достатній запас масштабованості на випадок зростання навантаження.

4. Для кожної категорії користувачів розробляється окремий інтерфейс, адаптований до їхніх конкретних потреб і рівня доступу. Пацієнт

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

отримує простий і зрозумілий інтерфейс для запису на прийом, перегляду історії відвідувань та керування особистими даними. Лікар бачить актуальні записи і може зручно відстежувати інформацію про записаних до нього пацієнтів. Адміністратор має доступ до розширеного інструментарію для створення графіків, керування обліковими записами користувачів та налаштування системи відповідно до поточних потреб закладу.

Відповідно до рекомендацій Міністерства охорони здоров'я України, сучасні медичні інформаційні системи повинні не лише автоматизувати повсякденні робочі процеси, а й формувати єдиний інформаційний простір для обміну даними на всіх етапах надання медичної допомоги [2]. Виходячи з цього, розроблювана система орієнтована на автоматизацію ключових процесів приватної поліклініки та повинна забезпечувати виконання основних функцій.

Система передбачає реєстрацію користувачів з подальшим входом до особистого кабінету. Залежно від призначеної ролі — пацієнт, лікар або адміністратор — користувач отримує відповідний рівень доступу до функцій системи. Кожна категорія користувачів має можливість переглядати та редагувати власні персональні дані, а вся інформація зберігається у структурованому вигляді та є доступною лише авторизованим користувачам.

Адміністратор має можливість формувати графіки роботи лікарів, визначати доступні часові інтервали для запису, а також додавати та видаляти лікарів і пацієнтів із системи. На основі сформованого розкладу пацієнти можуть самостійно обирати зручний час і записуватися на прийом до потрібного лікаря. Лікар, у свою чергу, має можливість переглядати список запланованих прийомів та підтверджувати або скасовувати записи пацієнтів.

Окрім цього, система зберігає історію прийомів, яка є доступною як для лікаря, так і для пацієнта в їхніх особистих кабінетах. Це дозволяє обом сторонам у будь-який момент переглянути інформацію про попередні відвідування. Важливою складовою є також захист персональних даних

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів та забезпечення доступу до інформації лише авторизованим особам.

1.2. Аналіз аналогів програмного продукту

Для визначення вимог до розроблюваної системи проведено огляд існуючих програмних рішень у сфері автоматизації роботи медичних закладів. Сучасний ринок ІТ-продуктів пропонує широкий спектр систем для запису на прийом та обліку відвідувань, серед яких зустрічаються як універсальні платформи, так і спеціалізовані медичні рішення.

Одним із таких рішень є вебпортал онлайн-запису до лікарів — portal-doctor.eleks.com. Вебпортал онлайн-запису до лікарів portal-doctor.eleks.com (рис. 1.1) є офіційним сервісом для запису до лікарів у місті Житомир, реалізованим на базі платформи ELEKS.

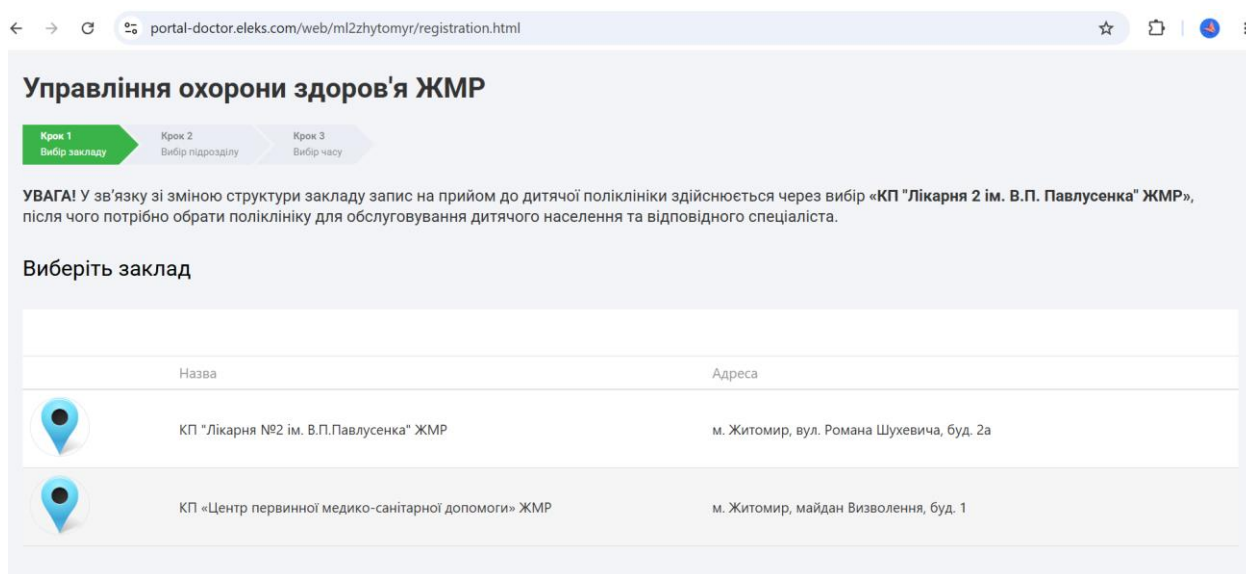


Рис. 1.1. Головна сторінка вебпорталу portal-doctor.eleks.com.

Сервіс дозволяє пацієнтам дистанційно обирати спеціальність лікаря, дату та час прийому, а також здійснювати запис на прийом.

Перевагою сервісу є простота використання та доступність з будь-якого пристрою з доступом до інтернету, що дозволяє зменшити навантаження на реєстратуру та скоротити час очікування пацієнтів. Проте даний сервіс орієнтований насамперед на зручність пацієнтів і надає лише можливість онлайн-запису на прийом (рис. 1.2). Він не забезпечує повноцінного обліку відвідувань та керування роботою медичного персоналу: відсутній особистий

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

кабінет пацієнта з історією візитів, а також обмежені можливості керування графіком лікарів. Таким чином, сервіс є зручним для пацієнтів, однак не задовольняє потреби медичного закладу у системному обліку та адмініструванні. Окрім обмеженого функціоналу, сервіс має спрощений та застарілий інтерфейс користувача, якому бракує сучасних елементів UI/UX-дизайну.

Рис. 1.2. Вигляд сторінки запису на прийом до певного лікаря на вебпорталі portal-doctor.eleks.com.

Наступним розглянутим аналогом є онлайн-платформа запису до лікарів — Oxford Medical Житомир. Сайт zhitomir.oxford-med.com.ua (рис. 1.3) є офіційним вебсервісом медичного центру Oxford Medical у місті Житомир. Сервіс дозволяє пацієнтам обирати лікаря за спеціальністю та залишати заявку на запис на прийом за допомогою відповідної форми.

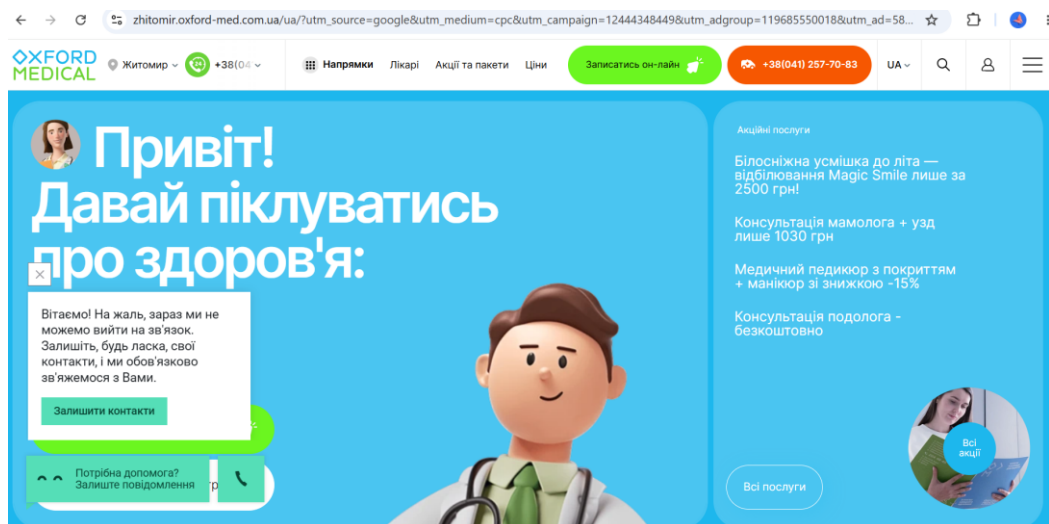


Рис. 1.3. Головна сторінка онлайн-платформи Oxford Medical Житомир

Перевагою платформи є простота використання та наявність контактних форм, що дозволяє пацієнтам швидко ініціювати запис. На відміну від попереднього аналогу, сайт Oxford Medical Житомир має сучасний та привабливий інтерфейс користувача. Візуальне оформлення виконане у єдиному корпоративному стилі з чіткою структурою навігації, що дозволяє користувачам легко орієнтуватися на сторінці. Сайт відповідає сучасним стандартам UI/UX-дизайну, має адаптивну верстку та коректно відображається як на комп'ютері, так і на мобільних пристроях.

Проте після подання заявки пацієнту телефонують оператори для підтвердження дати та часу прийому, тому система не є повністю автоматизованою. Це означає, що навантаження на адміністративний персонал залишається значним, а сам процес запису залежить від доступності операторів. Система Oxford Medical розроблена як корпоративне рішення для великої мережі медичних центрів, що охоплює десятки клінік по всій Україні. Це робить її надмірно масштабною та складною для впровадження у невеликій приватній поліклініці, оскільки більшість функцій орієнтована на потреби великої мережі, а не окремого закладу.

Крім того, адміністратор окремої клініки не має власних інструментів для керування розкладом лікарів — графіки та записи контролюються централізовано на рівні всієї мережі, що унеможливує гнучке налаштування системи під потреби конкретного закладу.

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

Третій аналог, який було обрано для аналізу — онлайн-платформа запису до лікарів — «Хілінг». Сайт медичного центру «Хілінг» (healing.zt.ua) є офіційним вебресурсом багатoproфільної приватної клініки у місті Житомир, що надає широкий спектр медичних послуг для дорослих і дітей (рис. 1.4).

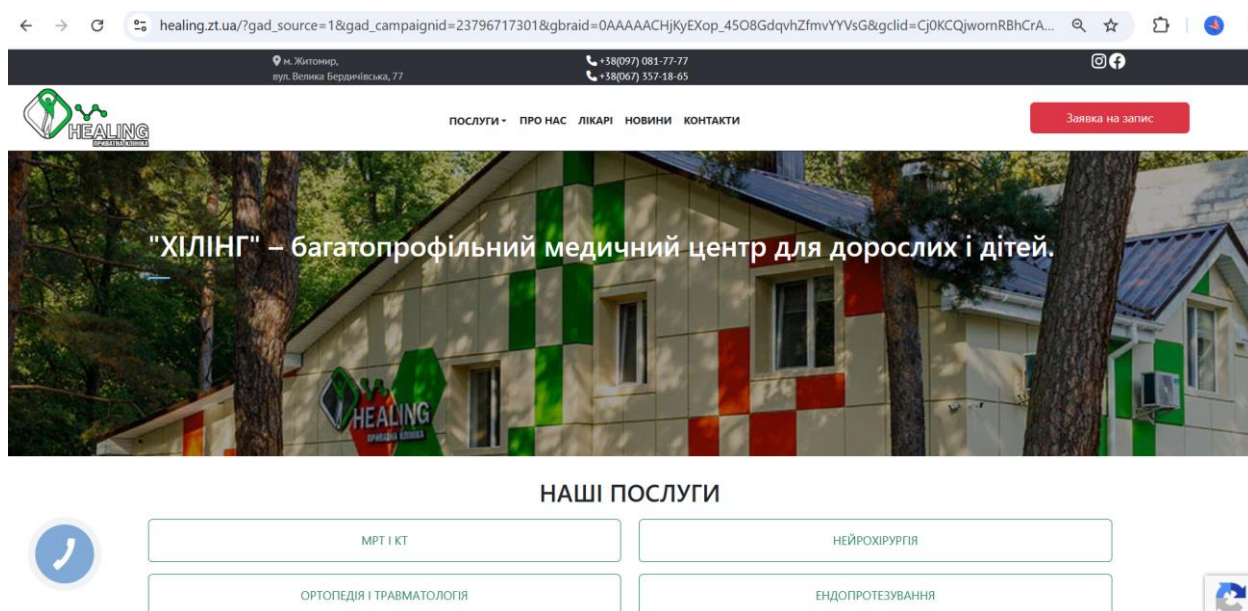


Рис. 1.4. Головна сторінка онлайн-платформи запису до лікарів «Healing»

Серед переваг платформи варто відзначити простий та інтуїтивно зрозумілий інтерфейс з адаптивною версткою, що забезпечує зручне використання як на комп'ютері, так і на мобільних пристроях. На сайті представлений перелік лікарів з фотографіями та профілями, що дозволяє пацієнту заздалегідь ознайомитися зі спеціалістами та обрати потрібного. Також на кожній сторінці сайту доступна форма для запису на прийом, що спрощує процес звернення до клініки.

На сайті представлений перелік лікарів з їхніми профілями, що дозволяє пацієнтам заздалегідь ознайомитися зі спеціалістами клініки. Форма запису на прийом є надто спрощеною — вона містить лише поля для введення імені та номера телефону, після чого персонал клініки зателефонує пацієнту для уточнення дати та часу візиту. Такий підхід дозволяє лише ініціювати запис дистанційно, однак не передбачає повністю автоматизованого процесу вибору конкретного лікаря, дати та часового слоту в реальному часі (рис. 1.5).

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Поля із зрочкою (*) обов'язкові для заповнення

Як вас звати? *

Ваш номер телефону *

☒ Я згоден на [обробку персональних даних](#)

Записатися

Запис на прийом

Ми передзвонимо вам найближчим часом і уточнимо деталі візиту.



Рис. 1.5. Форма запису на прийом на сайті Healing

Платформа має значні обмеження з точки зору керування записами та обліку відвідувань. Відсутній особистий кабінет пацієнта з історією попередніх візитів.

Проведений аналіз існуючих програмних рішень показав, що більшість розглянутих платформ орієнтована переважно на зручність пацієнтів і забезпечує лише базову функцію онлайн-запису на прийом. Жодна з розглянутих систем не надає повноцінного інструментарію для керування роботою медичного закладу зсередини — відсутні особисті кабінети для лікарів, інструменти керування розкладом для адміністратора та облік історії відвідувань. Це підтверджує доцільність розробки власної інформаційної системи, яка б поєднувала зручний інтерфейс для пацієнтів з повноцінними інструментами адміністрування та обліку для персоналу клініки.

Для наочного порівняння розглянутих аналогів із розроблюваною системою складено порівняльну таблицю основних функціональних можливостей (таблиця 1.1).

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1

Порівняльний аналіз існуючих рішень та власної розроблюваної системи

Параметр	portal- doctor.eleks.com	Oxford Medical	Healing	Власна система VITO
Автоматизований запис	+	-	-	+
Особистий кабінет пацієнта	-	+	-	+
Особистий кабінет лікаря	-	+	-	+
Облік відвідувань	-	+	-	+
Керування розкладом	-	-	-	+
Розмежування прав доступу	+	+	+	+
Підтвердження/ скасування прийому	-	+	-	+
Керування користувачами	-	+	-	+
Адаптивний інтерфейс	+	+	+	+
Сучасний дизайн	-	+	-	+

Проведене порівняння показало, що існуючі медичні інформаційні системи частково реалізують необхідний функціонал, проте жодна з них не забезпечує його повного охоплення. Зокрема, одні системи підтримують онлайн-запис, але не мають особистого кабінету, інші — забезпечують облік відвідувань, але не надають зручного інструменту для запису пацієнтів.

Таким чином, наявні рішення не до кінця забезпечують комплексну автоматизацію процесу запису і обслуговування пацієнтів.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

У зв'язку з цим виникає необхідність розробки власної інформаційної системи, яка поєднуватиме всі ключові функції, зокрема автоматизований запис, особистий кабінет пацієнта, облік відвідувань та сучасний зручний інтерфейс.

1.3. Вибір архітектури інформаційної системи

Вибір архітектури є одним із ключових етапів проєктування інформаційної системи, оскільки саме вона визначає загальну структуру системи, принципи взаємодії її компонентів, а також забезпечує стабільність, масштабованість і безпеку в процесі експлуатації. Правильно обрана архітектура спрощує подальшу розробку, тестування та супровід системи, а також дозволяє легко вносити зміни та розширювати функціонал без необхідності переписувати весь код.

Для веборієнтованих медичних систем ключовими архітектурними вимогами є відокремлення бізнес-логіки від рівня представлення даних, забезпечення можливості одночасної роботи багатьох користувачів, чіткий контроль доступу та розмежування прав між різними категоріями користувачів, а також захист персональної інформації пацієнтів і можливість подальшого розширення функціоналу системи. Кожна з цих вимог безпосередньо впливає на вибір архітектурного рішення: наприклад, захист персональних даних пацієнтів вимагає чіткого розмежування між рівнем доступу до даних і рівнем їх відображення, щоб унеможливити витік конфіденційної інформації через інтерфейс. Розмежування прав доступу, своєю чергою, означає, що система має на архітектурному рівні передбачати механізм перевірки ролі користувача перед виконанням будь-якої операції — незалежно від того, чи це пацієнт, лікар або адміністратор.

Перед вибором архітектурного рішення було розглянуто кілька поширених підходів до побудови вебзастосунків. Одношарова архітектура є простою у реалізації, однак погано підходить для систем із складною бізнес-логікою та великою кількістю користувачів, оскільки весь код зосереджений в одному місці без чіткого розділення відповідальності — це ускладнює

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

налагодження, тестування та будь-які подальші зміни у функціоналі. Багатошарова архітектура передбачає розділення системи на окремі рівні представлення, логіки та даних, що значно покращує структуру коду та спрощує його супровід. Мікросервісна архітектура забезпечує високу гнучкість та масштабованість, однак є надмірно складною для системи невеликого та середнього масштабу: вона вимагає окремої інфраструктури для кожного сервісу, складнішого налаштування міжсервісної взаємодії та суттєво більших витрат часу на розробку й підтримку. Порівняння розглянутих підходів наведено в таблиці 1.2.

Таблиця 1.2

Порівняння архітектурних підходів

Критерій	Одношарова	Багатошарова (MVC)	Мікросервісна
Складність розробки	Низька	Середня	Висока
Розділення відповідальності	Відсутнє	Чітке	Чітке
Масштабованість	Низька	Середня	Висока
Підходить для медичних систем	Ні	Так	Надлишково
Контроль доступу	Складний	Зручний	Зручний

З урахуванням вимог до розроблюваної системи та її масштабу було обрано архітектурний патерн MVC (Model-View-Controller) (рис. 1.6). Цей патерн є одним із найбільш поширених підходів у веброботці та дозволяє чітко розділити систему на три основні компоненти, кожен із яких відповідає за окремий аспект роботи програми [8].

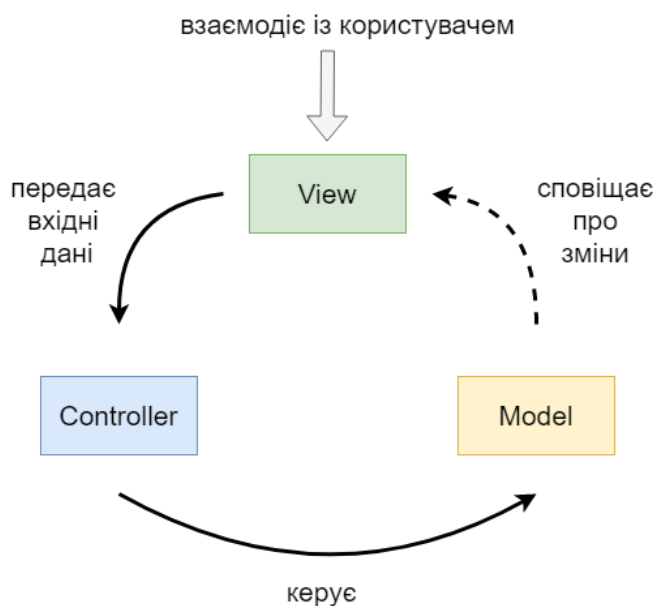


Рис. 1.6. Архітектурний патерн MVC

Перший компонент — Model (Модель) — відповідає за роботу з даними та бізнес-логіку системи. У контексті розроблюваної системи модель забезпечує взаємодію з базою даних MySQL, зокрема зберігання та обробку інформації про пацієнтів, лікарів, розклади та історію відвідувань. Для взаємодії з базою даних використовується розширення PDO (PHP Data Objects), яке забезпечує уніфікований інтерфейс доступу до даних та підтримує параметризовані запити. Параметризовані запити є важливим механізмом захисту: вони унеможливають SQL-ін'єкції — один із найпоширеніших типів атак на вебзастосунки, — оскільки дані користувача ніколи не підставляються безпосередньо в текст SQL-запиту, а передаються окремо та обробляються сервером баз даних як звичайні значення, а не як частина команди.

Другий компонент — View (Вигляд) — відповідає за відображення інформації користувачу. Він формує інтерфейс, який бачить користувач у браузері, та не містить жодної бізнес-логіки — лише шаблони з мінімальними вставками PHP для виведення підготовлених даних [29]. У розроблюваній системі вигляд формує сторінки особистих кабінетів пацієнта, лікаря та адміністратора, а також форми запису на прийом. Такий підхід означає, що

зміна зовнішнього вигляду сторінки не вимагає жодного втручання в логіку роботи системи — достатньо змінити лише відповідний шаблон.

Третій компонент — Controller (Контролер) — виступає посередником між моделлю та виглядом. Він отримує запити від користувача, передає їх до моделі для обробки та повертає відповідний результат у вигляді для відображення. Крім того, саме на рівні контролера відбувається перевірка прав доступу: перед виконанням будь-якої дії контролер перевіряє, чи авторизований користувач і яку роль він має в системі. Наприклад, коли пацієнт надсилає запит на запис до лікаря, контролер спочатку перевіряє наявність активної сесії та роль користувача, після чого звертається до моделі для перевірки доступності слоту та повертає результат у відповідний шаблон вигляду.

Патерн MVC було реалізовано власноруч. Такий підхід має низку переваг: він дає повний контроль над структурою та поведінкою системи, виключає залежність від зовнішніх бібліотек, які можуть застаріти або мати власні вразливості, а також зменшує навантаження на сервер завдяки відсутності зайвого коду. Структура проєкту організована таким чином, що кожен компонент MVC розміщується у відповідній директорії: папка controllers/ містить класи контролерів для кожного розділу системи, папка models/ — класи для роботи з базою даних, папка views/ — HTML-шаблони з мінімальними вставками PHP. Вхідною точкою системи є єдиний файл index.php, через який проходять усі запити — це дозволяє централізовано керувати маршрутизацією та застосовувати перевірки безпеки до кожного запиту ще до того, як він досягне контролера. Чітке розділення компонентів дозволяє розробляти та змінювати кожну частину системи незалежно, що значно спрощує супровід коду та пошук помилок.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Важливим етапом розробки є вибір інструментальних засобів та формування вимог до апаратного забезпечення, що забезпечуватимуть

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

стабільну та ефективну роботу системи. Вибір технологій здійснювався з урахуванням таких критеріїв: відповідність обраній архітектурі MVC, простота інтеграції компонентів між собою, доступність та відкритий вихідний код, широка документація, а також популярність технологій у сфері веброзробки.

Основною мовою програмування серверної частини обрано PHP — інтерпретовану серверну мову програмування з відкритим вихідним кодом, призначену для створення динамічних вебзастосунків [21, 22]. PHP добре інтегрується з HTML та виконується на стороні сервера, що дозволяє обробляти запити користувачів, реалізовувати бізнес-логіку запису на прийом, керування користувацькими акаунтами та забезпечувати ефективну взаємодію з базою даних [23, 24]. Для управління зовнішніми бібліотеками та залежностями проєкту використано менеджер пакетів Composer [10].

Для створення клієнтської частини системи використовуються технології HTML, CSS та JavaScript. HTML формує структуру сторінок та визначає розміщення елементів інтерфейсу [18]. CSS відповідає за візуальне оформлення та адаптивність інтерфейсу для різних пристроїв — комп'ютерів, планшетів та смартфонів [17]. JavaScript забезпечує інтерактивність інтерфейсу та дозволяє динамічно оновлювати окремі елементи сторінки без її повного перезавантаження, що покращує зручність використання системи [19].

Для прискорення розробки адаптивного інтерфейсу використовується Bootstrap — популярний CSS-фреймворк з відкритим вихідним кодом. Bootstrap надає готові стилі, компоненти та сіткову систему, що дозволяє швидко створювати сучасний та зручний інтерфейс, який коректно відображається на пристроях з різною роздільною здатністю екрану [9]. Використання Bootstrap значно скорочує час розробки інтерфейсу та забезпечує його єдиний візуальний стиль.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Для зберігання та обробки даних перед вибором конкретного рішення було проведено порівняльний аналіз трьох найпоширеніших реляційних систем керування базами даних: MySQL, PostgreSQL та Microsoft SQL Server.

MySQL є однією з найпоширеніших реляційних СКБД із відкритим вихідним кодом, яка розповсюджується безкоштовно за ліцензією GPL. Однією з її головних технічних переваг є висока швидкість виконання запитів на читання даних, що є особливо важливим для медичної інформаційної системи, адже більшість операцій у ній пов'язана саме з отриманням інформації — перегляд розкладу лікарів, історії записів пацієнтів, доступних часових слотів тощо. Крім того, переважна більшість вебхостингів підтримує MySQL за замовчуванням, а її інтеграція з PHP є однією з найбільш відпрацьованих у веброзробці.

PostgreSQL є потужною об'єктно-реляційною СКБД із відкритим кодом, яка підтримує складніші типи даних, розширені можливості індексування та більш суворе дотримання стандартів SQL. Однак для задач автоматизації роботи приватної поліклініки більшість цих можливостей є надлишковими, оскільки структура даних у розроблюваній системі є достатньо стандартною і не потребує специфічних типів даних чи складних аналітичних запитів. Додатковим недоліком є складніше адміністрування та значно менша підтримка з боку вебхостингів порівняно з MySQL.

Microsoft SQL Server, своєю чергою, є комерційною системою, орієнтованою переважно на великі корпоративні проекти. Її головним недоліком є платна ліцензія, тоді як безкоштовна версія Express має суттєві обмеження за обсягом бази даних та функціональністю. Крім того, SQL Server найефективніше працює в екосистемі Microsoft у зв'язці з Windows-сервером та технологіями .NET, що робить його інтеграцію з PHP менш зручною та обґрунтованою.

З урахуванням результатів порівняльного аналізу для зберігання та обробки даних у системі обрано MySQL. Вона забезпечує надійне зберігання структурованої інформації про пацієнтів, лікарів, записи на прийом, розклад

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

роботи, історію візитів та новини, при цьому залишаючись безкоштовною, простою у розгортанні та природно сумісною з обраним технологічним стеком.

Для реалізації функції формування електронних талонів використовується бібліотека Dompdf — безкоштовна PHP-бібліотека з відкритим вихідним кодом, яка дозволяє генерувати PDF-файли на основі HTML-розмітки [11]. У розроблюваній системі вона використовується для автоматичного створення електронного талона після успішного запису пацієнта на прийом, який містить інформацію про дату, час та лікаря.

Для локального розгортання та тестування системи використовується середовище XAMPP — безкоштовний пакет програмного забезпечення, який включає вебсервер Apache, систему керування базами даних MySQL та інтерпретатор PHP [30]. XAMPP дозволяє швидко розгорнути повноцінне середовище для розробки та тестування системи на локальному комп'ютері без необхідності налаштовувати кожен компонент окремо. Це значно спрощує процес розробки та дозволяє перевіряти роботу системи до її розгортання на реальному сервері.

Як середовище розробки обрано Visual Studio Code — сучасний кросплатформений редактор коду з відкритим вихідним кодом. VS Code підтримує PHP, HTML, CSS та JavaScript і забезпечує зручне редагування коду завдяки підсвітці синтаксису, інтелектуальному автодоповненню коду та інструментам для рефакторингу. Редактор має вбудовану інтеграцію з системою контролю версій Git [15]. Це дозволяє відстежувати зміни у проєкті та повертатися до попередніх версій коду за необхідності. Додатковою перевагою є велика бібліотека розширень, яка дозволяє налаштувати середовище розробки відповідно до конкретних потреб проєкту [28].

Стабільна робота інформаційної системи потребує дотримання певних вимог до апаратного та програмного забезпечення. Для розробки та локального тестування системи необхідна операційна система Windows 10 або новіша, не менше 8 ГБ оперативної пам'яті для забезпечення стабільної роботи

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

всіх компонентів середовища розробки, встановлений пакет XAMPP як локальний вебсервер, Visual Studio Code як середовище розробки, а також сучасний веббраузер для тестування клієнтської частини системи.

Таким чином, для розробки інформаційної системи автоматизації роботи приватної поліклініки було обрано набір технологій, які добре інтегруються між собою та відповідають усім вимогам до системи.

Висновки до першого розділу

У першому розділі було розглянуто та обґрунтовано основні аспекти розробки веборієнтованої інформаційної системи автоматизації роботи приватної поліклініки.

У підрозділі 1.1 сформульовано постановку задачі, визначено основні проблеми існуючих підходів до організації запису пацієнтів на прийом, які базуються на телефонному обслуговуванні та паперовій документації. Обґрунтовано необхідність розробки інформаційної системи, яка об'єднає процеси запису, обліку та керування в єдиний автоматизований механізм. Визначено основні цілі системи, сформульовано задачі розробки та описано функціональні можливості, які повинна забезпечувати система для кожної категорії користувачів.

У підрозділі 1.2 проведено аналіз існуючих програмних аналогів — вебпорталу онлайн-запису portal-doctor.eleks.com, платформи Oxford Medical Житомир та медичного центру «Healing». У ході аналізу виявлено, що найбільш функціональною з розглянутих систем є платформа Oxford Medical, яка порівняно з іншими аналогами має ширший набір можливостей. Проте навіть вона не забезпечує повного набору необхідних функцій — зокрема, відсутні інструменти керування розкладом лікарів для адміністратора окремої клініки, а процес запису не є повністю автоматизованим. Інші розглянуті системи орієнтовані переважно на зручність пацієнтів і не надають інструментів для повноцінного адміністрування та обліку відвідувань. Результати порівняльного аналізу підтвердили доцільність розробки власної системи та дозволили сформулювати конкретні вимоги до її функціональності.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

У підрозділі 1.3 обґрунтовано вибір архітектурного патерну MVC як основи для побудови системи. Розглянуто альтернативні підходи до організації архітектури вебсистем та визначено переваги MVC для розроблюваної системи, зокрема чітке розділення компонентів, спрощення розробки та тестування, а також можливість легкого розширення функціоналу в майбутньому.

У підрозділі 1.4 обґрунтовано вибір інструментальних засобів розробки. Як основну серверну мову програмування обрано PHP, для роботи з базою даних — MySQL, для розробки клієнтської частини — HTML, CSS, JavaScript та Bootstrap, для формування електронних талонів— бібліотеку Dompdf, а як середовище розробки — Visual Studio Code із локальним сервером XAMPP. Сформовано вимоги до апаратного забезпечення, необхідного для стабільної роботи системи.

Таким чином, результати першого розділу формують теоретичне та технологічне підґрунтя для подальшого проектування та розробки інформаційної системи автоматизації роботи приватної поліклініки.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ РОБОТИ ПРИВАТНОЇ ПОЛІКЛІНІКИ

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

Основою для проєктування будь-якої інформаційної системи є чітке визначення вимог до її функціональності. Для системи автоматизації роботи приватної поліклініки такі вимоги формуються виходячи з потреб трьох категорій користувачів — адміністратора, лікаря та пацієнта — кожен з яких взаємодіє із системою по-різному. Для наочного відображення взаємодії користувачів із системою розроблено діаграму варіантів використання (рис. 2.1) [12].

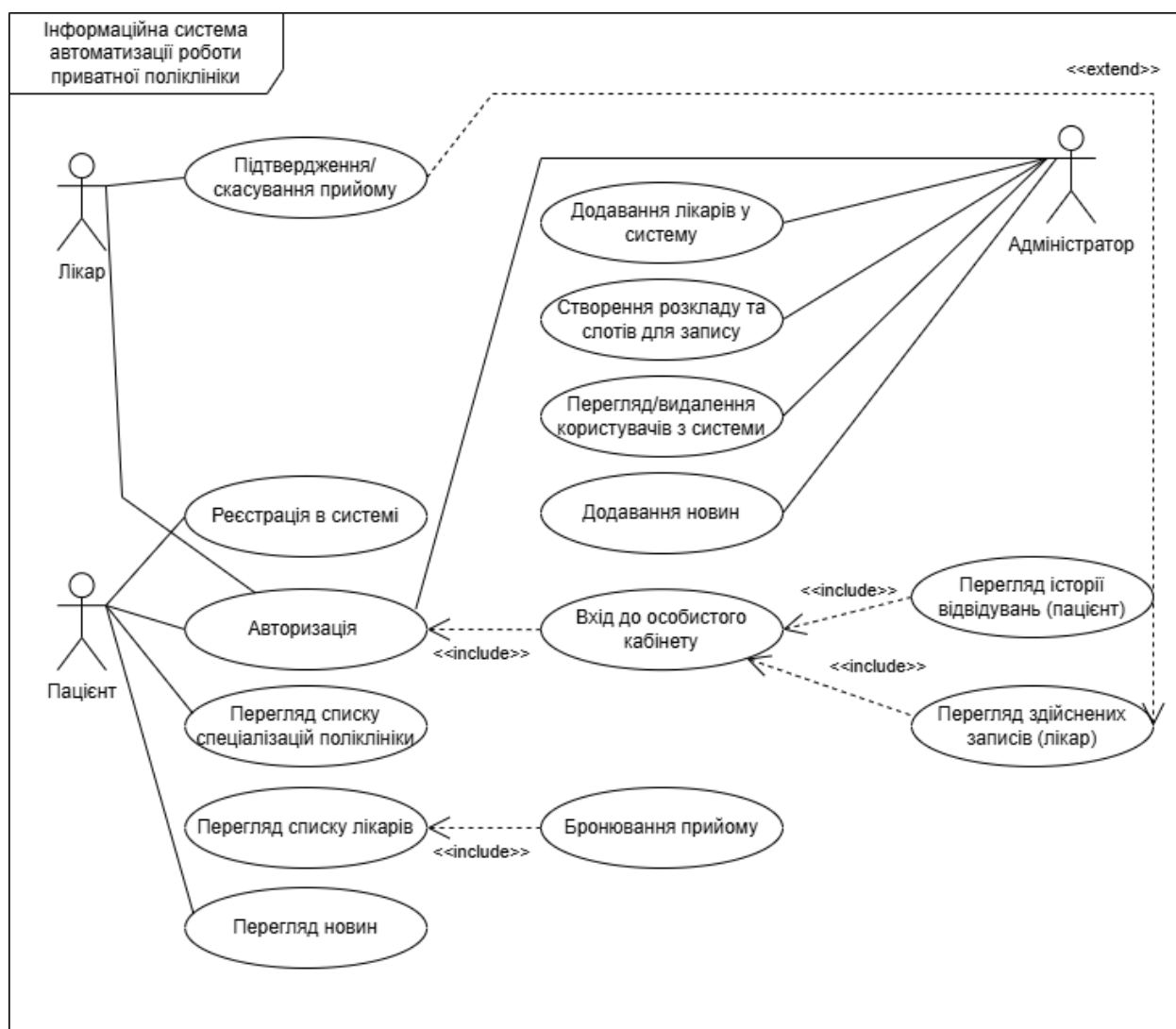


Рис. 2.1. Діаграма варіантів використання

Першим кроком взаємодії нового користувача з системою є процес реєстрації. Цей прецедент описує сценарій, за яким пацієнт створює власний обліковий запис, надаючи необхідні особисті дані для подальшого доступу до функціоналу системи (таблиця 2.1).

Таблиця 2.1.

Опис прецеденту «Реєстрація в системі»

Назва варіанту використання	Реєстрація в системі
Основні актори	Пацієнт
Короткий опис	Новий користувач може зареєструватися в системі, заповнивши форму з особистими даними, після чого отримує доступ до особистого кабінету як пацієнт.

Після реєстрації або при повторному відвідуванні системи користувач проходить процедуру авторизації. Цей прецедент описує загальний механізм входу в систему для всіх категорій користувачів (таблиця 2.2).

Таблиця 2.2.

Опис прецеденту «Авторизація»

Назва варіанту використання	Авторизація
Основні актори	Пацієнт, лікар, адміністратор
Короткий опис	Користувач входить до системи, вводячи логін та пароль, після чого отримує доступ до функціоналу відповідно до своєї ролі.

Після успішної авторизації користувач потрапляє до свого особистого кабінету, вміст якого формується відповідно до його ролі в системі. Цей прецедент відображає логіку розмежування доступу до функціональних розділів для пацієнта, лікаря та адміністратора (таблиця 2.3).

Таблиця 2.3.

Опис прецеденту «Вхід до особистого кабінету»

Назва варіанту використання	Вхід до особистого кабінету
Основні актори	Пацієнт, лікар, адміністратор
Короткий опис	Авторизований користувач отримує доступ до особистого кабінету, вміст якого залежить від його ролі в системі.

Перед тим як обрати лікаря, пацієнту необхідно визначитися з напрямком медичної допомоги. Цей прецедент описує функцію перегляду доступних медичних спеціалізацій, що пропонуються поліклінікою (таблиця 2.4).

Таблиця 2.4.

Опис прецеденту «Перегляд списку спеціалізацій поліклініки»

Назва варіанту використання	Перегляд списку спеціалізацій поліклініки
Основні актори	Пацієнт
Короткий опис	Пацієнт може переглянути перелік доступних спеціалізацій лікарів, що працюють у поліклініці, для вибору потрібного напрямку медичної допомоги.

Після вибору потрібної спеціалізації пацієнт переходить до перегляду конкретних лікарів цього напрямку. Цей прецедент описує сценарій формування та відображення списку лікарів обраної спеціалізації (таблиця 2.5).

Таблиця 2.5.

Опис прецеденту «Перегляд списку лікарів»

Назва варіанту використання	Перегляд списку лікарів
Основні актори	Пацієнт
Короткий опис	Пацієнт може переглянути список лікарів поліклініки конкретної спеціалізації.

Центральним функціональним сценарієм системи є процес запису пацієнта на прийом до лікаря. Цей прецедент описує вибір вільного часового слоту та автоматичне формування електронного талона після підтвердження запису (таблиця 2.6).

Таблиця 2.6.

Опис прецеденту «Бронювання прийому»

Назва варіанту використання	Бронювання прийому
Основні актори	Пацієнт
Короткий опис	Пацієнт може обрати зручний часовий слот та записатися на прийом до потрібного лікаря. Після успішного бронювання система автоматично формує електронний талон.

Окрім основного функціоналу запису, система надає пацієнтам доступ до інформаційного контенту закладу. Цей прецедент описує перегляд новин та оголошень, які публікує адміністрація поліклініки (таблиця 2.7).

Таблиця 2.7.

Опис прецеденту «Перегляд новин»

Назва варіанту використання	Перегляд новин
Основні актори	Пацієнт

Короткий опис	Пацієнт може переглядати актуальні новини та оголошення поліклініки, розміщені адміністратором.
---------------	---

Для зручності пацієнта система зберігає інформацію про всі його попередні звернення до лікарів. Цей прецедент описує функцію перегляду історії відвідувань у особистому кабінеті (таблиця 2.8).

Таблиця 2.8.

Опис прецеденту «Перегляд історії відвідувань»

Назва варіанту використання	Перегляд історії відвідувань
Основні актори	Пацієнт
Короткий опис	Авторизований пацієнт може переглянути історію своїх попередніх відвідувань у поліклініці, включаючи дату, час та лікаря.

З боку лікаря важливою функцією є можливість керування власним розкладом прийомів. Цей прецедент описує сценарій, за яким лікар підтверджує або скасовує запис конкретного пацієнта (таблиця 2.9).

Таблиця 2.9.

Опис прецеденту «Підтвердження/скасування прийому»

Назва варіанту використання	Підтвердження/скасування прийому
Основні актори	Лікар
Короткий опис	Авторизований лікар може переглянути список запланованих прийомів та підтвердити або скасувати запис пацієнта залежно від обставин.

Для організації власної роботи лікарю необхідний доступ до повного переліку записаних до нього пацієнтів. Цей прецедент описує функцію перегляду всіх здійснених записів із відповідною інформацією про дату, час та пацієнта (таблиця 2.10).

Таблиця 2.10.

Опис прецеденту «Перегляд здійснених записів»

Назва варіанту використання	Перегляд здійснених записів
Основні актори	Лікар
Короткий опис	Авторизований лікар може переглянути список усіх записів пацієнтів до нього, включаючи інформацію про дату, час та пацієнта.

Формування кадрового складу поліклініки в системі покладається на адміністратора. Цей прецедент описує процес додавання нового лікаря з заповненням його особистих даних та визначенням спеціалізації (таблиця 2.11).

Таблиця 2.11.

Опис прецеденту «Додавання лікарів у систему»

Назва варіанту використання	Додавання лікарів у систему
Основні актори	Адміністратор
Короткий опис	Авторизований адміністратор може додавати нових лікарів до системи, заповнюючи їхні особисті дані, спеціалізацію.

Ключовою адміністративною функцією є керування робочим часом лікарів. Цей прецедент описує процес формування розкладу та автоматичної генерації часових слотів, доступних для запису пацієнтів (таблиця 2.12).

Таблиця 2.12.

Опис прецеденту «Створення розкладу та слотів для запису»

Назва варіанту використання	Створення розкладу та слотів для запису
Основні актори	Адміністратор
Короткий опис	Авторизований адміністратор може формувати розклад роботи лікарів, визначаючи доступні дати та часові слоти для запису пацієнтів.

Адміністратор також відповідає за контроль облікових записів усіх користувачів системи. Цей прецедент описує функції перегляду переліку лікарів і пацієнтів та їх видалення за потреби (таблиця 2.13).

Таблиця 2.13.

Опис прецеденту «Перегляд/видалення користувачів з системи»

Назва варіанту використання	Перегляд/видалення користувачів з системи
Основні актори	Адміністратор
Короткий опис	Авторизований адміністратор може переглядати список усіх користувачів системи — лікарів та пацієнтів — а також видаляти їх із системи за необхідності.

Інформаційна підтримка пацієнтів реалізується через розділ новин, контент якого формує адміністратор. Цей прецедент описує процес додавання нової публікації, що згодом стає доступною для перегляду пацієнтами (таблиця 2.14).

Опис прецеденту «Додавання новин»

Назва варіанту використання	Додавання новин
Основні актори	Адміністратор
Короткий опис	Авторизований адміністратор може додавати новини та оголошення поліклініки, які відображатимуться для пацієнтів у відповідному розділі системи.

Оскільки пацієнт є основним користувачем системи, для нього передбачено найширший набір функцій з точки зору взаємодії з медичним закладом. Після реєстрації в системі пацієнт отримує доступ до особистого кабінету, де може переглядати актуальний список спеціалізацій поліклініки та лікарів, що працюють у закладі. Це дозволяє пацієнту самостійно обрати потрібного спеціаліста без необхідності звертатися до реєстратури.

Ключовою функцією для пацієнта є онлайн-бронювання прийому. Пацієнт обирає лікаря, переглядає доступні часові слоти та записується на зручний час. Після успішного бронювання система автоматично формує електронний талон у форматі PDF, який містить інформацію про дату, час та лікаря. Також пацієнт має доступ до історії своїх відвідувань, де зберігаються дані про всі попередні прийоми. Окрім цього, пацієнт може переглядати новини та оголошення поліклініки, що публікуються адміністратором.

Лікар є другою категорією користувачів системи, функціонал якого орієнтований на організацію власної роботи та керування записами пацієнтів. Після авторизації лікар отримує доступ до особистого кабінету, де відображається актуальна інформація про заплановані прийоми.

Основною функцією лікаря є перегляд списку записаних пацієнтів із зазначенням дати, часу та контактної інформації. На основі цих даних лікар може підтвердити або скасувати запис пацієнта залежно від обставин.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

Адміністратор є користувачем із найширшими правами доступу в системі, який відповідає за керування всіма основними процесами роботи поліклініки. Після авторизації адміністратор отримує доступ до повнофункціонального кабінету керування системою.

Основними функціями адміністратора є додавання нових лікарів до системи із зазначенням їхніх особистих даних та спеціалізації, а також формування розкладу роботи лікарів шляхом створення доступних часових слотів для запису пацієнтів. Це забезпечує актуальність розкладу та коректне відображення доступних часів для бронювання. Крім того, адміністратор має можливість переглядати список усіх користувачів системи — як лікарів, так і пацієнтів — та видаляти їхні облікові записи за необхідності. Додатковою функцією адміністратора є публікація новин та оголошень поліклініки, які відображаються у відповідному розділі для пацієнтів.

Основною функціональною вимогою даної інформаційної системи є реалізація реєстрації та авторизації користувачів із розмежуванням прав доступу відповідно до ролі — пацієнт, лікар або адміністратор. Кожен користувач повинен мати доступ лише до тих функцій системи, які відповідають його ролі. Система повинна забезпечувати зберігання та керування даними про користувачів, зокрема особистими даними пацієнтів і медичного персоналу, з можливістю їх перегляду та редагування. Обов'язковою функцією є онлайн-бронювання прийому пацієнтами з можливістю вибору лікаря та зручного часового слоту, а також автоматичне формування електронного талона у форматі PDF після успішного запису.

Нефункціональні вимоги визначають якісні характеристики системи, які забезпечують її зручність, надійність та ефективність у процесі експлуатації.

З точки зору зручності використання, інтерфейс системи повинен бути інтуїтивно зрозумілим для всіх категорій користувачів. Час навчання звичайного користувача, пацієнта — не повинен перевищувати 2 години, тоді як для медичного персоналу та адміністратора достатньо 1 години завдяки чіткій структурі інтерфейсу.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

З точки зору продуктивності, час відгуку системи для типових операцій — таких як авторизація, перегляд розкладу або бронювання прийому — не повинен перевищувати 15 секунд. Для складніших запитів, пов'язаних з обробкою великих обсягів даних, допустимий час відгуку становить не більше 30 секунд. Система повинна забезпечувати стабільну роботу для мінімум 40 одночасних користувачів, які працюють із загальною базою даних.

З точки зору надійності, доступність системи повинна бути на високому рівні — час простою не повинен перевищувати 6% від загального часу роботи. Середній час безвідмовної роботи системи має становити не менше 15 робочих днів, а максимально допустима кількість помилок — не більше 1 на 10 тисяч рядків коду.

Розглянемо діаграму класів, яка представлена на рисунку 2.2 та відображає основну структуру об'єктно-орієнтованої моделі інформаційної системи автоматизації роботи приватної поліклініки.

В основі системи лежить клас User, який відповідає за автентифікацію користувачів. Він зберігає базову інформацію — ідентифікатор, ім'я користувача, електронну пошту та хеш паролю, а також надає методи login() і logout() для входу та виходу з системи. Кожен користувач обов'язково має одну роль, яку описує клас Role. Залежно від ролі, користувач може мати пов'язаний профіль — пацієнта або лікаря. Зв'язок між User та цими класами позначений як 0..1, тобто профіль є необов'язковим, що дозволяє системі підтримувати користувачів, які не є ні лікарями, ні пацієнтами. У створеній інформаційній системі це — адміністратор.

Клас Patient зберігає персональні дані пацієнта: ім'я, прізвище, номер телефону, дату народження та адресу. Він надає методи updateProfile() для оновлення власних даних та viewAppointments() для перегляду записів на прийом.

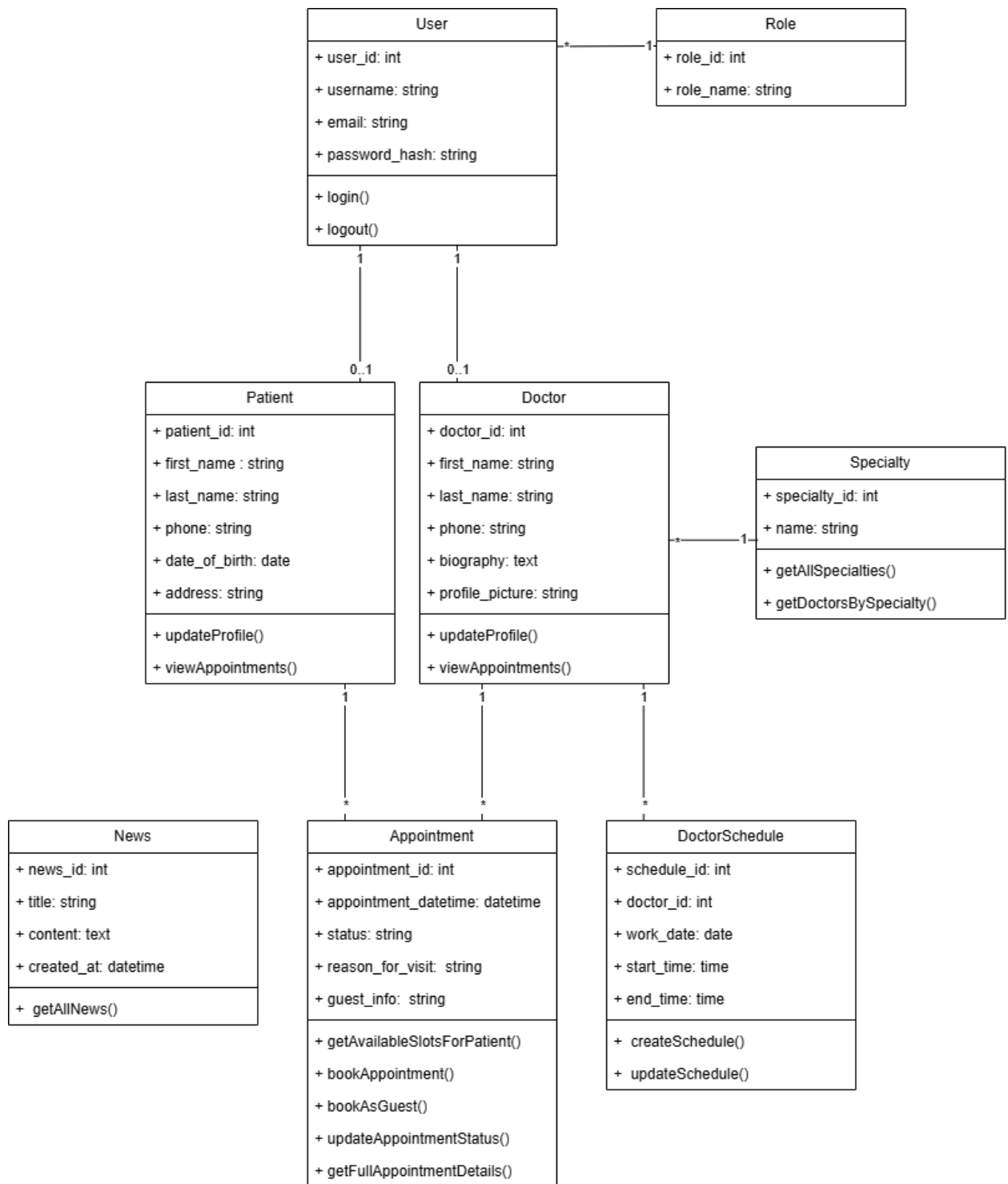


Рис. 2.2. Діаграма класів

Клас Doctor має схожу структуру, але доповнений додатковими для лікаря полями — біографією та посиланням на фото профілю. Методи збігаються з методами пацієнтів: `updateProfile()` і `viewAppointments()`. Кожен лікар обов'язково належить до певної медичної спеціалізації, яку описує клас `Specialty`. Він зберігає назву спеціальності та надає методи `getAllSpecialties()` і `getDoctorsBySpecialty()` для навігації по каталогу лікарів.

Центральним класом системи є Appointment, який описує факт запису пацієнта до лікаря. Він зберігає дату та час прийому, поточний статус запису, причину візиту, а також поле guest_info для зберігання даних незареєстрованого відвідувача. Це поле свідчить про те, що система підтримує гостьовий запис без необхідності реєстрації — за це відповідає метод bookAsGuest(). Для зареєстрованих пацієнтів передбачено метод bookAppointment(), а також getAvailableSlotsForPatient() для перегляду вільних часових слотів. Крім того, клас містить методи updateAppointmentStatus() для зміни статусу прийому та getFullAppointmentDetails() для отримання повної інформації про запис. Appointment пов'язаний одночасно з Patient і Doctor через зв'язок «один до багатьох» — тобто один пацієнт або лікар може мати багато записів.

Для управління робочим графіком лікаря використовується клас DoctorSchedule. Він зберігає дату роботи, час початку та завершення прийому. Методи createSchedule() і updateSchedule() дозволяють формувати розклад. Зв'язок з Doctor є «один до багатьох», оскільки лікар може мати розклад на декілька різних дат.

Клас News є незалежним модулем системи і не пов'язаний з жодним іншим класом. Він зберігає заголовок, текст публікації та дату створення. Метод getAllNews() дозволяє отримати перелік усіх новин для відображення на сторінці з новинами системи.

2.2. Розробка бази даних системи

За результатами аналізу вимог до інформаційної системи автоматизації роботи приватної поліклініки було визначено ключові критерії для порівняння СКБД. Оскільки система передбачає зберігання медичних даних пацієнтів, управління розкладом лікарів та обробку записів на прийом, при виборі СКБД важливо враховувати як технічні, так і економічні аспекти [5, 6]. Результати порівняння представлені в таблиці 2.15.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння систем управління базами даних

Критерій	MySQL	PostgreSQL	MS SQL Server
Ліцензія	Відкрита	Відкрита	Комерційна
Вартість	Безкоштовна	Безкоштовна	Платна
Продуктивність	Висока	Висока	Висока
Складність налаштування	Низька	Середня	Середня
Інтеграція з PHP	Відмінна	Добра	Задовільна
Масштабованість	Середня	Висока	Висока
Підтримка хостингами	Широка	Обмежена	Обмежена
Безпека даних	Висока	Висока	Висока
Підтримка транзакцій	Повна	Повна	Повна

За результатами порівняльного аналізу для розробки бази даних інформаційної системи автоматизації роботи приватної поліклініки було обрано MySQL. Основними аргументами на користь цього вибору є безкоштовне розповсюдження, висока продуктивність при операціях читання, відмінна інтеграція з PHP та локальним середовищем розробки XAMPP, повна підтримка ACID-транзакцій [20], а також надійні механізми захисту даних. За сукупністю визначених критеріїв MySQL є найбільш доцільним вибором для медичного закладу рівня приватної поліклініки та повністю задовольняє функціональні й технічні вимоги розроблюваної системи.

База даних системи складається з дев'яти таблиць, кожна з яких відповідає за зберігання певної категорії даних (таблиця 2.16 — 2.24).

Таблиця users зберігає основні облікові дані всіх користувачів системи незалежно від їхньої ролі (таблиця 2.16).

Таблиця 2.16

Опис таблиці users

Поле	Тип	Призначення
user_id	int	Унікальний ідентифікатор користувача
username	varchar	Ім'я користувача (електронна пошта)
password_hash	varchar	Хеш паролю користувача
email	varchar	Електронна пошта користувача
created_at	datetime	Дата та час створення запису
updated_at	datetime	Дата та час останнього оновлення запису

Таблиця roles містить перелік доступних ролей користувачів, які визначають рівень доступу до функціоналу системи (таблиця 2.17).

Таблиця 2.17

Опис таблиці roles

Поле	Тип	Призначення
role_id	int	Унікальний ідентифікатор ролі
role_name	varchar	Назва ролі (Patient, Doctor, Admin)

Таблиця user_roles реалізує зв'язок між користувачами та їхніми ролями, забезпечуючи механізм розмежування прав доступу (таблиця 2.18).

Таблиця 2.18

Опис таблиці user_roles

Поле	Тип	Призначення
user_id	int	Ідентифікатор користувача (зовнішній ключ до таблиці users)
role_id	int	Ідентифікатор ролі (зовнішній ключ до таблиці roles)

Таблиця patients зберігає персональні дані пацієнтів, необхідні для ведення медичного обліку та запису на прийом (таблиця 2.19).

Таблиця 2.19

Опис таблиці patients

Поле	Тип	Призначення
patient_id	int	Унікальний ідентифікатор пацієнта
user_id	int	Ідентифікатор користувача (зовнішній ключ до таблиці users)
first_name	varchar	Ім'я пацієнта
last_name	varchar	Прізвище пацієнта
date_of_birth	date	Дата народження пацієнта
phone	varchar	Номер телефону пацієнта
address	varchar	Адреса проживання пацієнта
created_at	datetime	Дата та час створення запису

продовж. табл. 2.19

updated_at	datetime	Дата та час останнього оновлення запису
------------	----------	---

Таблиця doctors містить інформацію про лікарів поліклініки, включаючи їхні особисті дані, спеціалізацію та біографію (таблиця 2.20).

Таблиця 2.20

Опис таблиці doctors

Поле	Тип	Призначення
doctor_id	int	Унікальний ідентифікатор лікаря
user_id	int	Ідентифікатор користувача (зовнішній ключ до таблиці users)
first_name	varchar	Ім'я лікаря
last_name	varchar	Прізвище лікаря
phone	varchar	Номер телефону лікаря
specialty_id	int	Ідентифікатор спеціалізації (зовнішній ключ до таблиці specialties)
biography	text	Біографія та опис лікаря
profile_picture	varchar	Шлях до фото профілю лікаря
created_at	datetime	Дата та час створення запису
updated_at	datetime	Дата та час останнього оновлення запису

Таблиця specialties зберігає перелік медичних спеціалізацій, які пропонує поліклініка (таблиця 2.21).

Таблиця 2.21

Опис таблиці specialties

Поле	Тип	Призначення
specialty_id	int	Унікальний ідентифікатор спеціалізації
name	varchar	Назва медичної спеціалізації

Таблиця doctor_schedules визначає робочий розклад кожного лікаря за днями тижня та доступними годинами прийому (таблиця 2.22).

Таблиця 2.22

Опис таблиці doctor_schedules

Поле	Тип	Призначення
schedule_id	int	Унікальний ідентифікатор запису розкладу
doctor_id	int	Ідентифікатор лікаря (зовнішній ключ до таблиці doctors)
day_of_week	varchar	День тижня робочого дня лікаря
start_time	time	Час початку прийому
end_time	time	Час завершення прийому
is_available	tinyint	Ознака доступності лікаря у вказаний день

продовж. табл. 2.22

created_at	datetime	Дата та час створення запису
updated_at	datetime	Дата та час останнього оновлення запису

Таблиця appointments фіксує факт запису пацієнта на прийом до конкретного лікаря (таблиця 2.23).

Таблиця 2.23

Опис таблиці appointments

Поле	Тип	Призначення
appointment_id	int	Унікальний ідентифікатор запису на прийом
patient_id	int	Ідентифікатор пацієнта (зовнішній ключ до таблиці patients)
doctor_id	int	Ідентифікатор лікаря (зовнішній ключ до таблиці doctors)
appointment_datetime	datetime	Дата та час запланованого прийому
reason_for_visit	varchar	Причина звернення пацієнта
status	varchar	Поточний статус запису на прийом
created_at	datetime	Дата та час створення запису
updated_at	datetime	Дата та час останнього оновлення запису

Таблиця news призначена для зберігання новин та оголошень поліклініки, що публікуються адміністратором (таблиця 2.24).

Таблиця 2.24

Опис таблиці news

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор новини
title	varchar	Заголовок новини
content	text	Текст публікації
image_path	varchar	Шлях до зображення новини
created_at	datetime	Дата та час публікації новини

Зв'язки між таблицями бази даних наочно представлені на рисунку 2.3.

Центральною таблицею системи є users, яка пов'язана з таблицями patients та doctors через зовнішній ключ user_id, що забезпечує прив'язку профілю до облікового запису. Зв'язок між users та user_roles реалізує механізм призначення ролей, а таблиця roles зберігає їх перелік. Таблиця doctors пов'язана з таблицею specialties через поле specialty_id, що визначає медичну спеціалізацію кожного лікаря, а також з таблицею doctor_schedules через поле doctor_id, яке зберігає його робочий розклад. Таблиця appointments є сполучною між patients та doctors — вона містить зовнішні ключі обох таблиць, фіксуючи факт запису конкретного пацієнта до конкретного лікаря. Таблиця news є єдиною незалежною таблицею, яка не має зовнішніх ключів і функціонує як окремий інформаційний модуль.

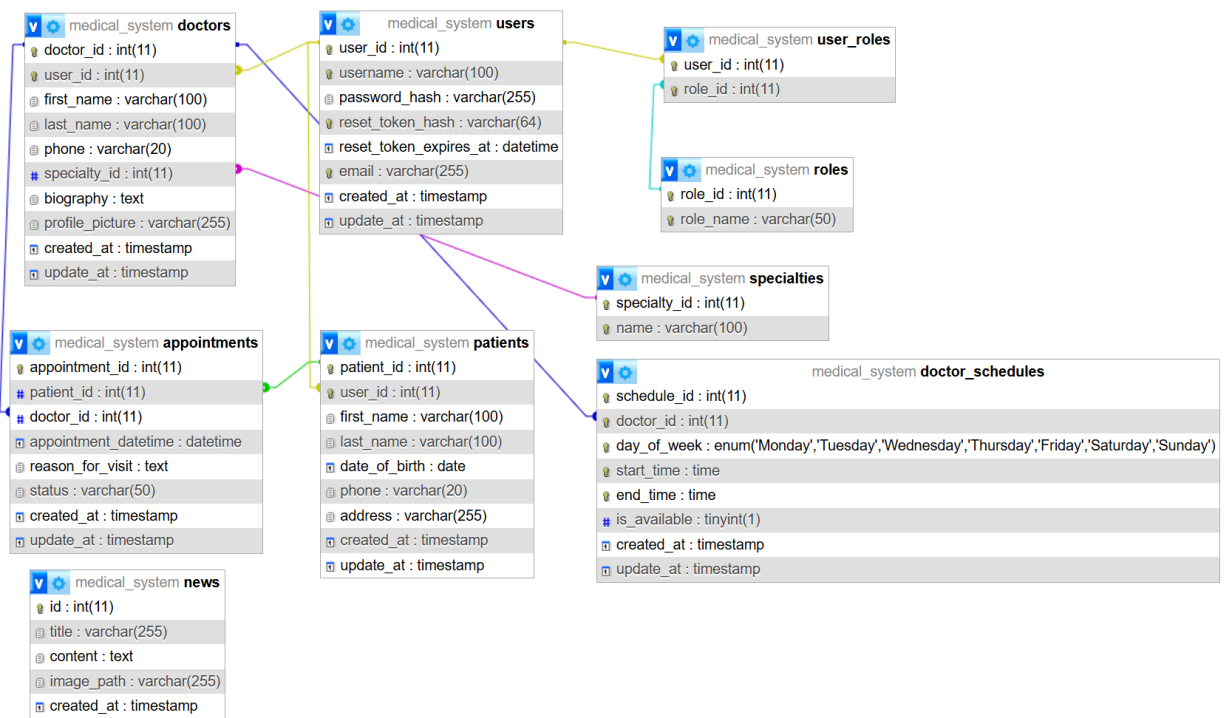


Рис. 2.3. Структура бази даних

2.3. Проєктування та реалізація алгоритмів роботи системи

Інформаційну систему автоматизації роботи приватної поліклініки спроектовано з використанням архітектурного патерну MVC, що забезпечує чіткий розподіл обов'язків між компонентами системи та дозволяє ефективно організувати алгоритми обробки даних [4]. Узагальнений алгоритм функціонування системи передбачає послідовну обробку подій, що ініціюються користувачами, та подальшу зміну стану об'єктів у базі даних. Взаємодія елементів системи розпочинається з надсилання запиту від клієнтської частини до серверного ядра. У процесі автентифікації об'єкт класу User виконує перевірку облікових даних, порівнюючи введені значення з хешованими паролями у сховищі. Успішна верифікація дозволяє системі розмежувати права доступу через визначення ролі користувача, що в подальшому регулює набір доступних методів для виклику. Вся подальша взаємодія програмних елементів відбувається в межах сесії, де контролери координують обмін даними між інтерфейсом та бізнес-сутностями.

Для розуміння загальної архітектури взаємодії користувачів із платформою першочергово розроблено узагальнений алгоритм розподілу прав

доступу та маршрутизації. Цей процес візуалізовано на загальній діаграмі активності системи (рис. 2.4), яка демонструє логіку поведінки системи від моменту входу користувача на сайт до його перенаправлення у відповідний функціональний блок [27].

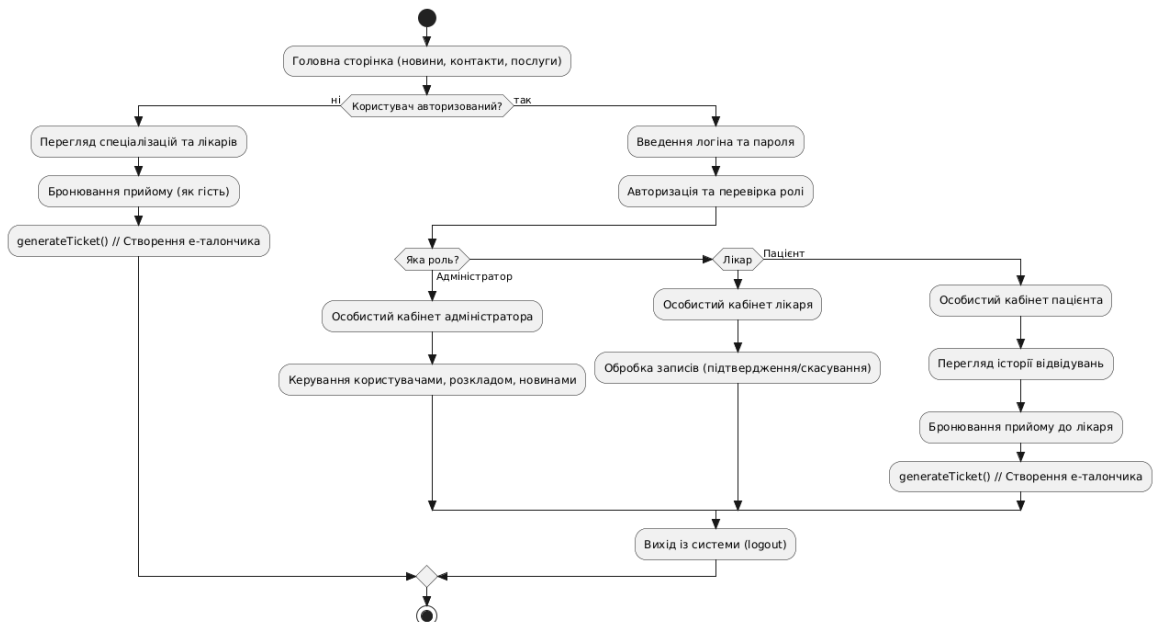


Рис. 2.4. Загальна діаграма активності: маршрутизація користувачів та розподіл ролей

Центральним бізнес-процесом у роботі комплексу є алгоритм бронювання прийомів, що інтегрує функціонал класів Patient, Doctor, DoctorSchedule та Appointment. Важливою особливістю системи є багаторівневий доступ до цього функціоналу, що реалізується через розмежування прав зареєстрованих (авторизованих) пацієнтів та незареєстрованих користувачів (гостей). Деталізовану логіку взаємодії користувача із серверною частиною під час резервування часу прийому наведено на діаграмі активності (рис. 2.5).

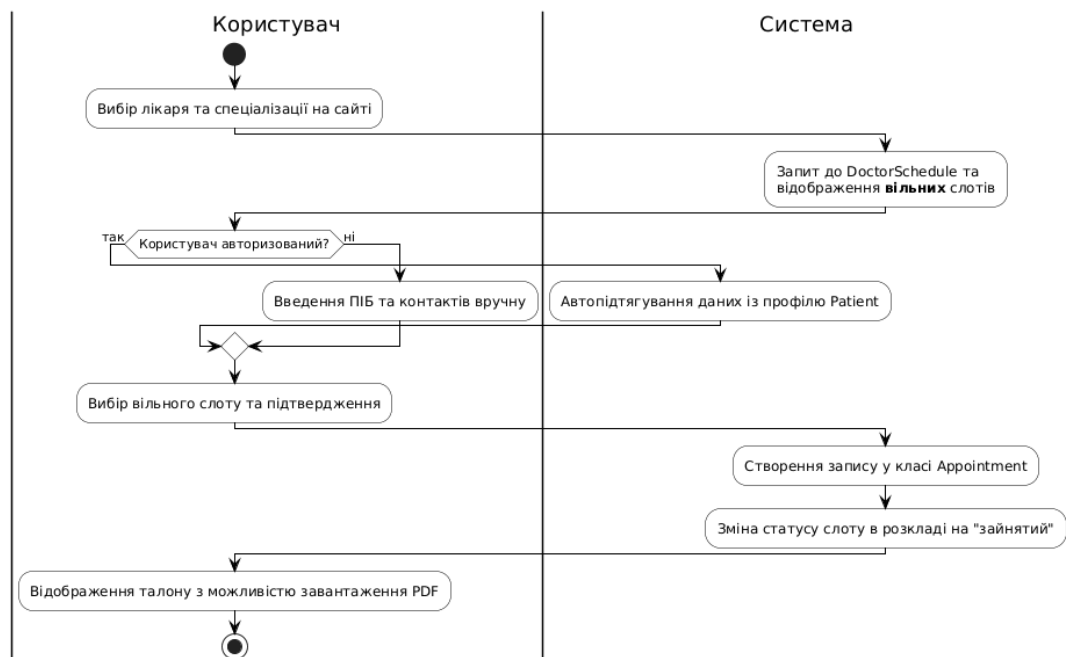


Рис. 2.5. Діаграма активності процесу бронювання прийому пацієнтом

Алгоритм, представлений на рис. 2.5, відображає оптимізований шлях користувача, де система мінімізує кількість помилкових дій. Процес розпочинається з вибору лікаря та його спеціалізації на клієнтській частині сайту. Після цього контролер надсилає запит до класу DoctorSchedule, здійснюючи фільтрацію даних у режимі реального часу, завдяки чому користувачеві виводяться виключно вільні часові слоти.

Після фіксації вибору конкретного вільного слоту та підтвердження транзакції, об'єкт класу Appointment ініціює створення нового запису про прийом, паралельно змінюючи статус відповідного часового інтервалу в DoctorSchedule на «зайнятий». Завершенням алгоритму є формування екранної форми талону, яка містить інтегровану функцію генерації та завантаження документа у форматі PDF для збереження користувачем.

Окрім клієнтської частини, важливу роль у функціонуванні системи відіграє внутрішній алгоритм керування робочим часом медичного персоналу, який реалізується через робоче місце адміністратора. Логіку формування графіків та автоматизованої генерації інтервалів для запису відображено на діаграмі активності (рис. 2.6).

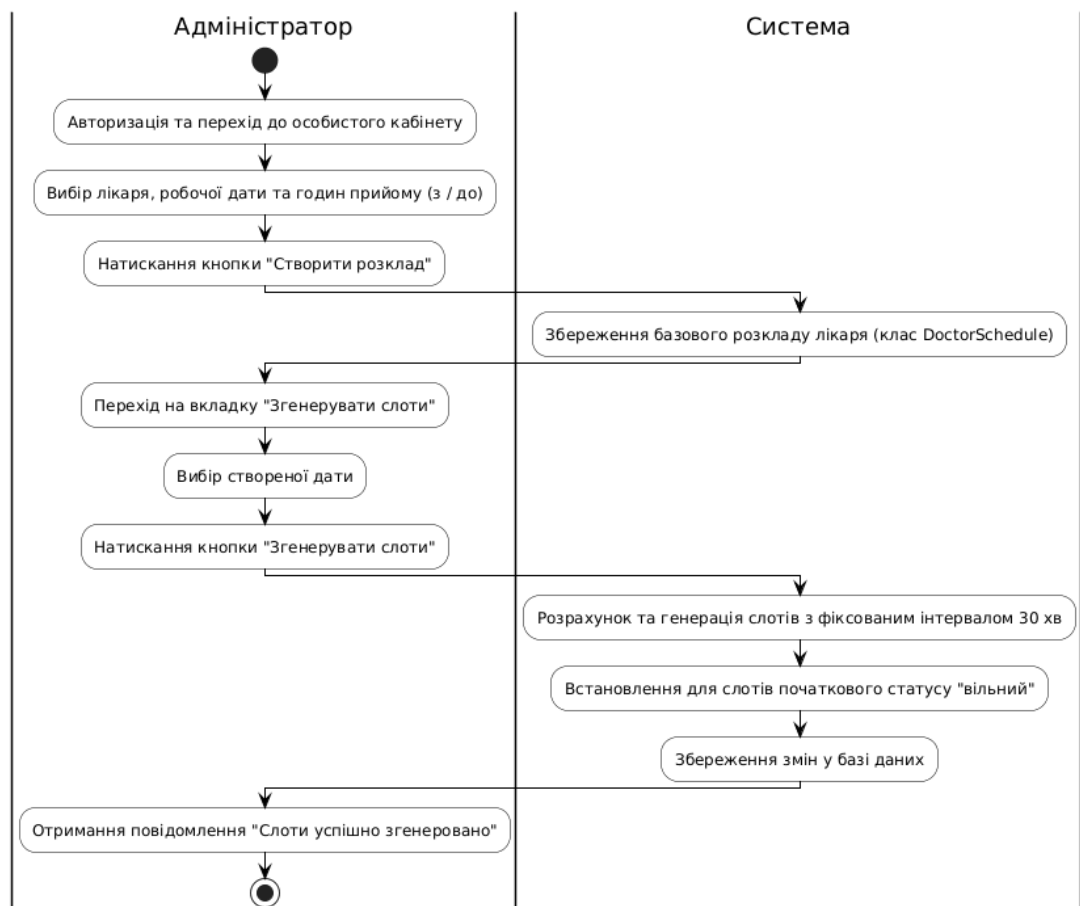


Рис. 2.6. Діаграма активності процесу формування розкладу та генерації слотів адміністратором

Як показано на рис. 2.6, процес адміністрування розкладу складається з двох послідовних та взаємопов'язаних етапів, що виконуються в особистому кабінеті адміністратора після проходження перевірки прав доступу.

На першому етапі адміністратор створює базовий розклад (робочий день): обирає конкретного лікаря, календарну дату та вказує часові межі зміни (години початку та завершення роботи). Ці дані передаються на сервер, де об'єкт класу *DoctorSchedule* реєструє новий робочий день у системі.

Другий етап передбачає перехід до спеціалізованої вкладки «Згенерувати слоти». Адміністратор вказує раніше створену дату та ініціює автоматичний процес «нарізки» робочого часу. Системний алгоритм автоматично, без додаткового залучення користувача, розраховує та генерує масив часових слотів із зафіксованим у коді системи інтервалом тривалістю 30 хвилин. Кожному новоствореному об'єкту слота присвоюється початковий

статус «вільний», після чого пакет даних зберігається в базі даних. Успішне завершення алгоритму супроводжується генерацією та виведенням на екран адміністратора спливаючого інформаційного повідомлення, а згенеровані слоти стають доступними для вибору пацієнтами на клієнтській частині платформи.

2.4. Реалізація функціоналу системи

Реалізацію функціоналу системи доцільно розглянути на прикладі основних програмних модулів, що забезпечують її роботу. Далі описано механізми авторизації користувачів, запису пацієнтів на прийом, управління профілями лікарів і пацієнтів, а також інші компоненти системи. Для пояснення логіки роботи наведено ключові фрагменти коду, а повні тексти відповідних програмних модулів розміщено в додатках.

Одним із ключових компонентів системи є модуль авторизації та реєстрації користувачів, реалізований у класі AuthModel. Його основним призначенням є перевірка облікових даних під час входу до системи, створення нових облікових записів та призначення ролей користувачам.

Під час реєстрації користувача система спочатку створює запис у таблиці users, де зберігаються основні дані для автентифікації. Пароль перед записом до бази даних хешується за допомогою алгоритму bcrypt, що забезпечує безпечне зберігання облікових даних. Після створення користувача система визначає його роль та формує відповідний запис у таблиці user_roles.

```
public function registerUser($firstName, $lastName, $email, $password,
    $roleName = 'Patient')
    {
        $this->pdo->beginTransaction();
        try {
            $passwordHash = password_hash($password, PASSWORD_BCRYPT);
            $userSql = "INSERT INTO users (username, password_hash, email)
VALUES (:username, :password_hash, :email)";
            $userStmt = $this->pdo->prepare($userSql);
            $userStmt->execute([
                ':username' => $email,
                ':password_hash' => $passwordHash,
                ':email' => $email
            ]);
            $userId = $this->pdo->lastInsertId();
            $roleSql = "SELECT role_id FROM roles WHERE role_name =
:role_name LIMIT 1";
            $roleStmt = $this->pdo->prepare($roleSql);
```

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $roleStmt->execute([':role_name' => $roleName]);
        $roleData = $roleStmt->fetch();
        $roleId = $roleData ? $roleData['role_id'] : 1;
        $userRoleSql = "INSERT INTO user_roles (user_id, role_id)
VALUES (:user_id, :role_id)";
        $userRoleStmt = $this->pdo->prepare($userRoleSql);
        $userRoleStmt->execute([':user_id' => $userId, ':role_id' =>
        $roleId]);

        if ($roleName === 'Patient') {
            $profileSql = "INSERT INTO patients (user_id, first_name,
last_name) VALUES (:user_id, :first_name, :last_name)";
            $profileStmt = $this->pdo->prepare($profileSql);
            $profileStmt->execute([
                ':user_id' => $userId,
                ':first_name' => $firstName,
                ':last_name' => $lastName
            ]);
        } else {
            $profileSql = "INSERT INTO doctors (user_id, first_name,
last_name, specialty_id)
VALUES (:user_id, :first_name, :last_name,
:specialty_id)";
            $profileStmt = $this->pdo->prepare($profileSql);
            $profileStmt->execute([
                ':user_id' => $userId,
                ':first_name' => $firstName,
                ':last_name' => $lastName,
                ':specialty_id' => 1
            ]);
        }
        $this->pdo->commit();
        return true;
    } catch (Exception $e) {
        $this->pdo->rollBack();
        return false;
    }
}

```

Особливістю реалізації є використання транзакцій, що дозволяє виконувати всі операції реєстрації як єдиний процес. У разі виникнення помилки зміни не будуть збережені, що гарантує цілісність даних у базі. Залежно від обраної ролі додатково створюється профіль пацієнта або лікаря у відповідній таблиці бази даних.

Для авторизації використовується метод `loginUser()`, який виконує пошук користувача за адресою електронної пошти та перевіряє правильність введеного пароля. У разі успішної перевірки користувач отримує доступ до функціоналу системи відповідно до своєї ролі.

```

public function loginUser($email, $password)
{
    $user = $this->findUserByEmail($email);
    if ($user && password_verify($password, $user['password_hash'])) {

```

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        unset($user['password_hash']);
        return $user;
    }
    return false;
}

```

Ще одним із ключових модулів системи є механізм запису пацієнтів на прийом до лікаря, реалізований у класі Appointment. Даний модуль відповідає за пошук доступних часових слотів, бронювання прийомів, зміну їх статусу та отримання детальної інформації про заплановані візити.

Процес бронювання реалізовано методом bookAppointment(). Після вибору користувачем конкретного часового слота система змінює його статус на Booked, зберігає ідентифікатор пацієнта та причину звернення. Додаткова перевірка статусу запису безпосередньо в SQL-запиті унеможливорює повторне бронювання вже зайнятого часу іншим користувачем.

```

    public function bookAppointment($appointment_id, $patient_id,
        $reason_for_visit = '')
    {
        $sql = "UPDATE appointments
            SET status = 'Booked',
              patient_id = :patient_id,
              reason_for_visit = :reason_for_visit
            WHERE appointment_id = :appointment_id AND status = 'Available'";

        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([
            ':patient_id' => $patient_id,
            ':reason_for_visit' => $reason_for_visit,
            ':appointment_id' => $appointment_id
        ]);
    }

```

Окремо реалізовано можливість запису без створення облікового запису. У такому випадку система зберігає контактні дані відвідувача та причину звернення у полі guest_info.

```

    public function bookAsGuest($appointment_id, $first_name, $last_name,
        $phone, $reason)
    {
        $sql = "UPDATE appointments
            SET status = 'Booked',
              reason = :reason,
              guest_info = :guest_info
            WHERE appointment_id = :id AND status = 'Available'";
        $guest_data = "$first_name $last_name, Тел: $phone";

        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([
            ':id' => $appointment_id,
            ':reason' => $reason,
            ':guest_info' => $guest_data
        ]);
    }

```

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Повна реалізація класу Appointment наведена у додатку А.

Невід'ємною частиною процесу запису на прийом є модуль підтвердження та формування електронного талона для пацієнта. Додатково пацієнту надається можливість завантажити талон у форматі PDF. Метод downloadTicket() отримує дані про запис, формує HTML-шаблон талона.

```
public function downloadTicket($appointment_id)
{
    $appointment = $this->appointmentModel->getFullAppointmentDetails($appointment_id);

    if (!$appointment) {
        die("Запис не знайдено.");
    }

    $datetime = new DateTime($appointment['appointment_datetime']);
    $formattedDate = $datetime->format('d.m.Y');
    $formattedTime = $datetime->format('H:i');
    ob_start();
    include __DIR__ . '/../views/appointments/pdf_template.php';
    $html = ob_get_clean();
    PdfGenerator::download($html, "Talon_VITO_$appointment_id.pdf");
    exit();
}
```

Повна реалізація класу AppointmentController наведена у додатку А.

Важливим компонентом системи є модуль управління розкладом лікарів та генерації часових слотів для запису, реалізований у класі AdminController. Даний модуль забезпечує налаштування робочого графіку лікарів по днях тижня та автоматичне створення доступних часових інтервалів для запису пацієнтів. Процес управління розкладом складається з двох етапів. На першому етапі адміністратор визначає робочі години лікаря для конкретного дня тижня. Перед збереженням система перевіряє наявність уже існуючого розкладу на обраний день, щоб уникнути дублювання записів у таблиці doctor_schedules.

```
if (isset($_POST['add_schedule'])) {
    $doctorId = intval($_POST['doctor_id']);
    $dayOfWeek = $_POST['day_of_week'];
    $startTime = $_POST['start_time'];
    $endTime = $_POST['end_time'];

    if (!$doctorId) {
        $error = "Оберіть лікаря!";
    } else {
        try {
            $stmtCheck = $pdo->prepare("SELECT schedule_id FROM
            doctor_schedules WHERE doctor_id = :d AND day_of_week = :w");
```

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

$stmtCheck->execute([':d' => $doctorId, ':w' =>
$dayOfWeek]);

    if ($stmtCheck->fetch()) {
        $error = "Розклад на цей день вже існує!";
    } else {
        $stmtIns = $pdo->prepare("INSERT INTO doctor_schedules
(doctor_id, day_of_week, start_time, end_time, is_available) VALUES (?, ?, ?, ?,
1)");
        $stmtIns->execute([$doctorId, $dayOfWeek, $startTime,
$endTime]);

        $message = "Розклад успішно додано!";
    }
} catch (Exception $e) {
    $error = "Помилка: " . $e->getMessage();
}
}

```

На другому етапі адміністратор генерує конкретні часові слоти на обрану дату. Система визначає день тижня для вказаної дати, завантажує відповідний розклад лікаря та автоматично створює записи у таблиці appointments з інтервалом у 30 хвилин. Додатково виконується перевірка на існування слота, що унеможливорює створення дублікатів у разі повторної генерації.

```

if (isset($_POST['generate_slots'])) {
    $doctorId = intval($_POST['doctor_id']);
    $date = $_POST['date'];

    if (!$doctorId || !$date) {
        $error = "Будь ласка, оберіть лікаря та дату!";
    } elseif (strtotime($date) < strtotime(date('Y-m-d'))) {
        $error = "Неможливо генерувати слоти на минулу дату!";
    } else {
        try {
            $dayOfWeek = date('l', strtotime($date));
            $stmt = $pdo->prepare("SELECT start_time, end_time FROM
doctor_schedules
                                WHERE doctor_id = ? AND TRIM(day_of_week)
                                = ? AND is_available = 1");
            $stmt->execute([$doctorId, $dayOfWeek]);
            $schedule = $stmt->fetch(PDO::FETCH_ASSOC);

            if (!$schedule) {
                $error = "На $date (день: $dayOfWeek) у цього лікаря
немає налаштованого розкладу.";
            } else {
                $start = strtotime($schedule['start_time']);
                $end = strtotime($schedule['end_time']);
                $created = 0;

                for ($t = $start; $t < $end; $t += 1800) {
                    $timeSlot = date('H:i:s', $t);
                    $fullDatetime = $date . ' ' . $timeSlot;

                    $stmtCheck = $pdo->prepare("SELECT appointment_id
FROM appointments

```

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

WHERE doctor_id = ? AND
appointment_datetime = ?");
$stmtCheck->execute([$doctorId, $fullDatetime]);

if (!$stmtCheck->fetch()) {
    $stmtIns = $pdo->prepare("INSERT INTO appointments
(doctor_id, appointment_datetime, status) VALUES (?, ?, 'Available')");
    $stmtIns->execute([$doctorId, $fullDatetime]);
    $created++;
}
}
$message = "Успішно згенеровано $created слотів на
$date.";
}
} catch (Exception $e) {
    $error = "Помилка: " . $e->getMessage();
}
}
}

```

Висновки до другого розділу

У другому розділі проведено проєктування та технічну реалізацію інформаційної системи автоматизації роботи приватної поліклініки.

У підрозділі 2.1 визначено об'єктно-орієнтовану структуру системи. Побудовано діаграму варіантів використання, що дозволило чітко окреслити функціональні ролі пацієнта, лікаря та адміністратора. Визначено склад основних класів системи та їхню ієрархію, що заклало основу для подальшого програмного моделювання.

У підрозділі 2.2 обґрунтовано вибір MySQL як СКБД для інформаційної системи приватної поліклініки за результатами порівняльного аналізу з PostgreSQL та MS SQL Server. Наведено фізичну структуру таблиць бази даних, що складається з дев'яти таблиць («users», «roles», «user_roles», «patients», «doctors», «specialties», «doctor_schedules», «appointments», «news»), визначено зв'язки між ними та забезпечено цілісність даних через систему зовнішніх ключів.

У підрозділі 2.3 розроблено та спроектовано алгоритми роботи системи. Визначено узагальнений алгоритм обробки подій, що забезпечує коректну зміну стану об'єктів у базі даних після авторизації користувача.

Ключову увагу приділено алгоритму бронювання прийомів, який інтегрує функціонал класів Patient, Doctor, DoctorSchedule та Appointment. Проєктування системи враховує багаторівневий доступ до функціоналу: від

обмежених можливостей незареєстрованих користувачів (гостей) до повного керування даними для пацієнтів, лікарів та адміністратора.

У підрозділі 2.4 здійснено програмну реалізацію ключових функціональних модулів інформаційної системи. Реалізовано механізми реєстрації та авторизації користувачів із розмежуванням прав доступу відповідно до ролей, а також забезпечено безпечне зберігання облікових даних шляхом використання хешування паролів. Крім того, реалізовано механізм запису пацієнтів на прийом до лікаря, який включає пошук доступних часових слотів, бронювання прийомів та зміну їх статусів. Використані програмні рішення забезпечують цілісність даних, запобігають повторному бронюванню одного й того самого часу та створюють необхідні умови для надійної роботи інформаційної системи.

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

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ АВТОМАТИЗАЦІЇ РОБОТИ ПРИВАТНОЇ ПОЛІКЛІНІКИ

3.1. Порядок встановлення та налаштування параметрів системи.

Перед початком встановлення необхідно переконатися, що апаратне та програмне забезпечення відповідає мінімальним вимогам для коректної роботи системи. Для локального розгортання потрібна операційна система Windows 10 або новіша, не менше 8 ГБ оперативної пам'яті, а також встановлений пакет XAMPP актуальної версії, що включає сумісні між собою версії Apache, PHP та MySQL. Рекомендується використовувати актуальні стабільні версії всіх компонентів, оскільки застарілі версії PHP можуть не підтримувати окремі синтаксичні конструкції та функції, застосовані у коді системи, а також не отримують оновлень безпеки. Аналогічно, версія MySQL має забезпечувати повну підтримку зовнішніх ключів, транзакцій та інших механізмів цілісності даних, які використовуються у структурі бази даних. Для тестування клієнтської частини рекомендується використовувати актуальну версію будь-якого сучасного браузера — Google Chrome, Mozilla Firefox або Microsoft Edge.

Для забезпечення коректної роботи інформаційної системи необхідно попередньо підготувати серверне середовище, що підтримує виконання PHP-скриптів та роботу з системою керування базами даних MySQL. Як локальне серверне середовище використовується пакет XAMPP, який включає вебсервер Apache, інтерпретатор PHP та СКБД MySQL, що дозволяє розгорнути повноцінне робоче середовище на локальному комп'ютері без необхідності налаштовувати кожен компонент окремо [7].

Процес розгортання системи розпочинається із запуску панелі керування XAMPP та активації двох ключових сервісів — Apache і MySQL. Apache забезпечує обробку HTTP-запитів та виконання PHP-скриптів, тоді як MySQL відповідає за зберігання та обробку даних системи. Після успішного

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

запуску обох сервісів їх статус у панелі керування змінюється на активний, що підтверджує готовність середовища до подальшого налаштування.

Наступним етапом є підготовка бази даних. Для цього необхідно відкрити у браузері інтерфейс phpMyAdmin та створити нову базу даних. Після створення бази даних необхідно перейти до вкладки «Імпорт» та завантажити SQL-файл, що входить до складу дистрибутиву системи. Цей файл містить повну структуру бази даних — усі необхідні таблиці та логічні зв'язки між ними, а також початкові дані, зокрема перелік медичних спеціальностей та обліковий запис адміністратора системи, що дозволяє розпочати роботу одразу після встановлення без необхідності вносити базові дані вручну. Після успішного імпорту в базі даних автоматично формуються всі таблиці, необхідні для функціонування системи.

Після налаштування бази даних необхідно встановити з'єднання між серверною частиною системи та СКБД. Для цього у файлі `/config/db.php` слід вказати актуальні параметри підключення: адресу хоста (за замовчуванням `localhost`), ім'я користувача бази даних, пароль та назву створеної бази даних. Саме цей файл є єдиною точкою конфігурації підключення до бази даних у системі — усі інші компоненти звертаються до нього для отримання параметрів з'єднання, тому коректність вказаних даних є обов'язковою умовою працездатності системи.

Завершальним кроком інсталяції є розміщення вихідних файлів програмного комплексу в кореневій директорії вебсервера. У випадку використання XAMPP такою директорією є папка `htdocs`, розташована у каталозі встановлення пакету. Усі файли та папки проєкту необхідно скопіювати безпосередньо до цієї директорії, зберігаючи оригінальну структуру проєкту.

Після успішного встановлення системи адміністратор може увійти до свого облікового запису, використовуючи облікові дані, що містяться у імпортованому SQL-файлі. Після авторизації адміністратор автоматично

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

потрапляє на сторінку особистого кабінету, яка є центральною точкою керування системою і надає доступ до всіх адміністративних функцій.

Першочерговим завданням після встановлення, як правило, є наповнення системи даними про лікарів. Для цього адміністратор переходить до відповідного розділу кабінету, де може додати нового лікаря до системи, вказавши його персональні дані та спеціальність. Після додавання лікаря необхідно сформулювати для нього розклад роботи — створити часові слоти, які визначають доступні години для запису пацієнтів. Кожен слот прив'язується до конкретного лікаря та дати, що дозволяє системі коректно відображати доступний час для запису.

Паралельно з налаштуванням розкладу адміністратор має можливість керувати інформаційним наповненням системи — додавати новини або оголошення, які відображатимуться на сторінці з новинами сайту і можуть містити актуальну інформацію для пацієнтів: зміни у режимі роботи, оновлення переліку послуг тощо. Для поточного керування системою адміністратор має доступ до повного списку зареєстрованих пацієнтів та лікарів. У цьому розділі він може переглядати детальну інформацію про кожного користувача, а також видаляти облікові записи у разі необхідності.

Таким чином, особистий кабінет адміністратора об'єднує всі ключові функції керування системою в єдиному інтерфейсі, що робить процес адміністрування зручним і не потребує технічних знань від персоналу закладу.

Для коректної роботи системи у виробничому середовищі, при розгортанні на реальному сервері замість локального, процес встановлення є аналогічним, проте має деякі особливості. Файли проєкту розміщуються у кореневій директорії хостингу, а параметри підключення у файлі `/config/db.php` оновлюються відповідно до даних, наданих хостинг-провайдером. Більшість сучасних хостингів надають доступ до phpMyAdmin для керування базою даних, тому процес імпорту SQL-файлу залишається незмінним [25]. Після оновлення параметрів конфігурації система стає

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

доступною за реальною адресою домену та готова до повноцінної експлуатації.

3.2. Структура інтерфейсу

Ефективність програмного продукту значною мірою залежить від якості реалізації інтерфейсу користувача, який має забезпечувати швидкий доступ до ключових функцій та бути інтуїтивно зрозумілим для різних категорій користувачів [3].

Вітальну сторінку інформаційної системи спроектовано як головну точку входу, що забезпечує швидку навігацію та формування першого враження про сервіс. У верхній частині розміщено навігаційну панель (Header), яка містить логотип системи та основні пункти меню: «Про нас», «Новини», «Послуги», «Контакти» та кнопку «Увійти». Таке розташування дозволяє користувачеві миттєво зорієнтуватися в структурі сайту та отримати доступ до особистого кабінету або інформаційних розділів (рис. 3.1).

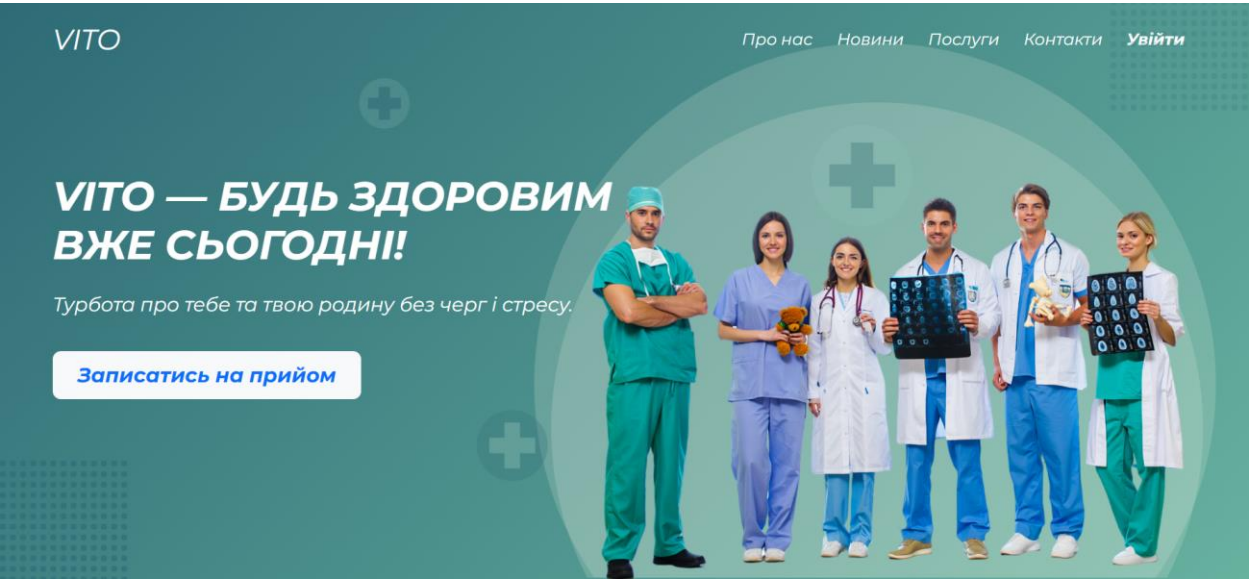


Рис. 3.1. Головна сторінка інформаційної системи

Важливим функціональним елементом є кнопка «Записатись на прийом», виділена візуально для акцентування уваги відвідувача. Вона ініціює перехід до алгоритму вибору лікаря та часового слота, що є критично важливим для реалізації основної мети системи — автоматизації запису на прийом.

Інформаційний блок «Про нас» спрямований на формування довіри до клініки та ознайомлення користувача з місією закладу. Дизайн розділу виконано в лаконічному стилі з чітким поділом на текстовий та візуальний контент. У лівій частині подано опис концепції клініки, де наголошується на доступності, комфорті та дотриманні протоколів доказової медицини. Права частина інтерфейсу містить акцентний блок з ключовими показниками успішності: досвідом роботи (понад 8 років) та кількістю фахівців (понад 20 кваліфікованих лікарів) (рис. 3.2).

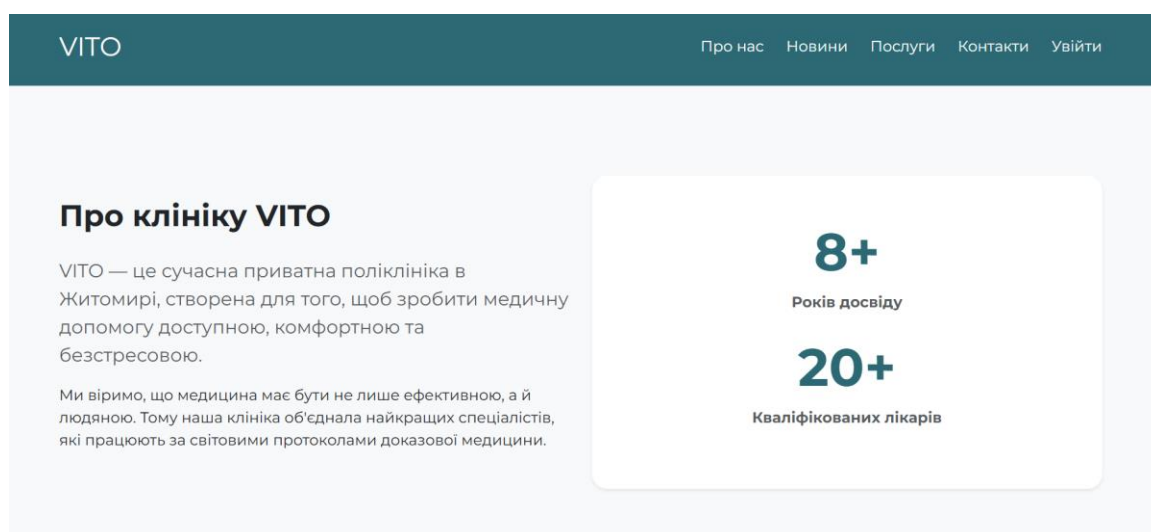


Рис. 3.2. Структура сторінки «Про нас»

Розділ «Контакти» є критично важливим елементом інтерфейсу для забезпечення прямого зв'язку пацієнта з клінікою. Структура сторінки організована за принципом швидкого доступу до необхідної контактної інформації: фізичної адреси з описом орієнтирів, номерів телефонів реєстратури (з підтримкою месенджерів Viber та Telegram) та електронної пошти. Для зручності користувачів окремий блок відведено під графік роботи, де наочно відображено години прийому для буднів та вихідних днів. Виділення неділі червоним кольором як вихідного дня дозволяє пацієнту миттєво зорієнтуватися в часових обмеженнях для планування візиту. Такий підхід до компонування інформації мінімізує кількість запитань від користувача та сприяє ефективній комунікації між адміністрацією клініки та пацієнтами (рис. 3.3).

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

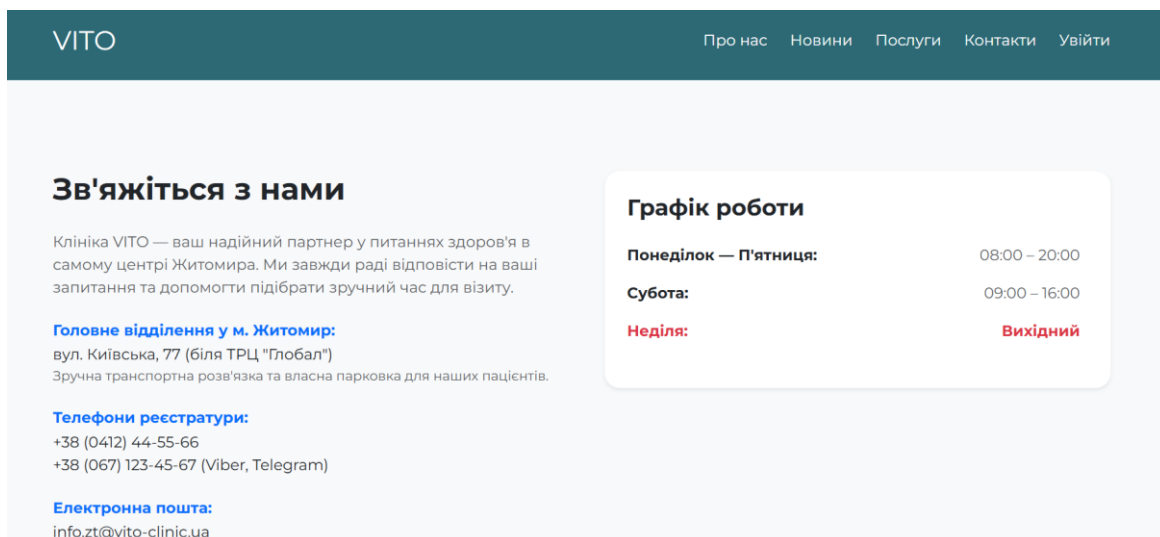


Рис. 3.3. Структура сторінки «Контакти»

Сторінку «Послуги» спроектовано як навігаційний хаб, що дозволяє пацієнту легко знайти спеціаліста згідно з необхідним профілем допомоги. Інтерфейс реалізовано у вигляді сітки з картковим відображенням (рис. 3.4). У межах кожної картки міститься коротка інформаційна довідка про напрямок та кнопка «Переглянути лікарів», яка є ключовим елементом взаємодії. Активуючи цю кнопку, користувач переходить до фільтрованого списку фахівців, що працюють у обраній галузі, що мінімізує кількість зусиль для пошуку відповідного лікаря та пришвидшує процес переходу до бронювання прийому. Візуальне оформлення розділу характеризується простотою та акцентом на контенті, що забезпечує високу зручність використання інтерфейсу.

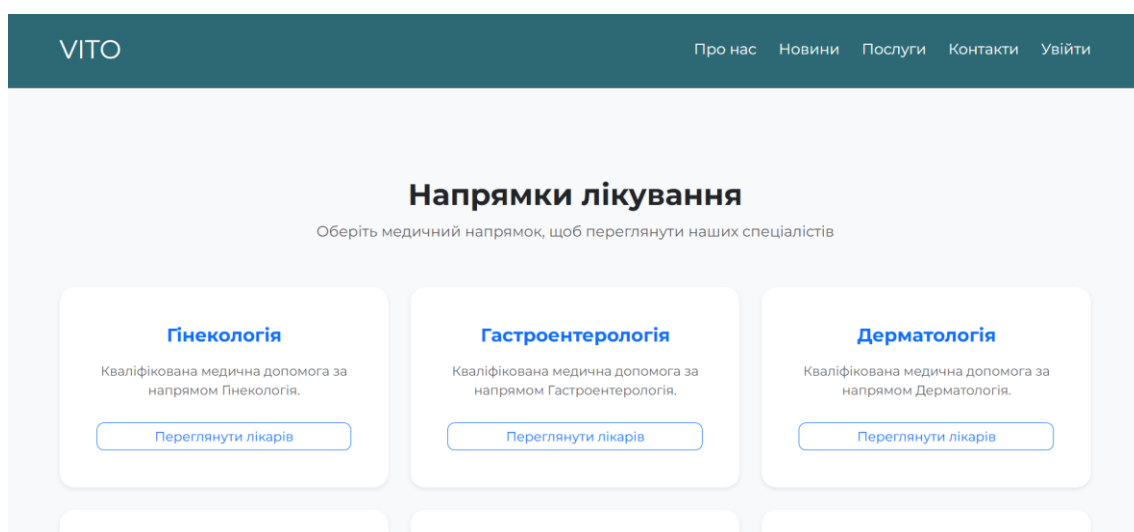


Рис. 3.4. Структура сторінки «Послуги»

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка деталізації напрямку лікування є проміжним етапом у процесі пошуку фахівця. Вона відображає обрану користувачем категорію (наприклад, «Гастроентерологія») (рис. 3.5) та забезпечує зворотну навігацію до загального переліку послуг через посилання «Назад до послуг», що сприяє зручності переміщення між розділами системи.

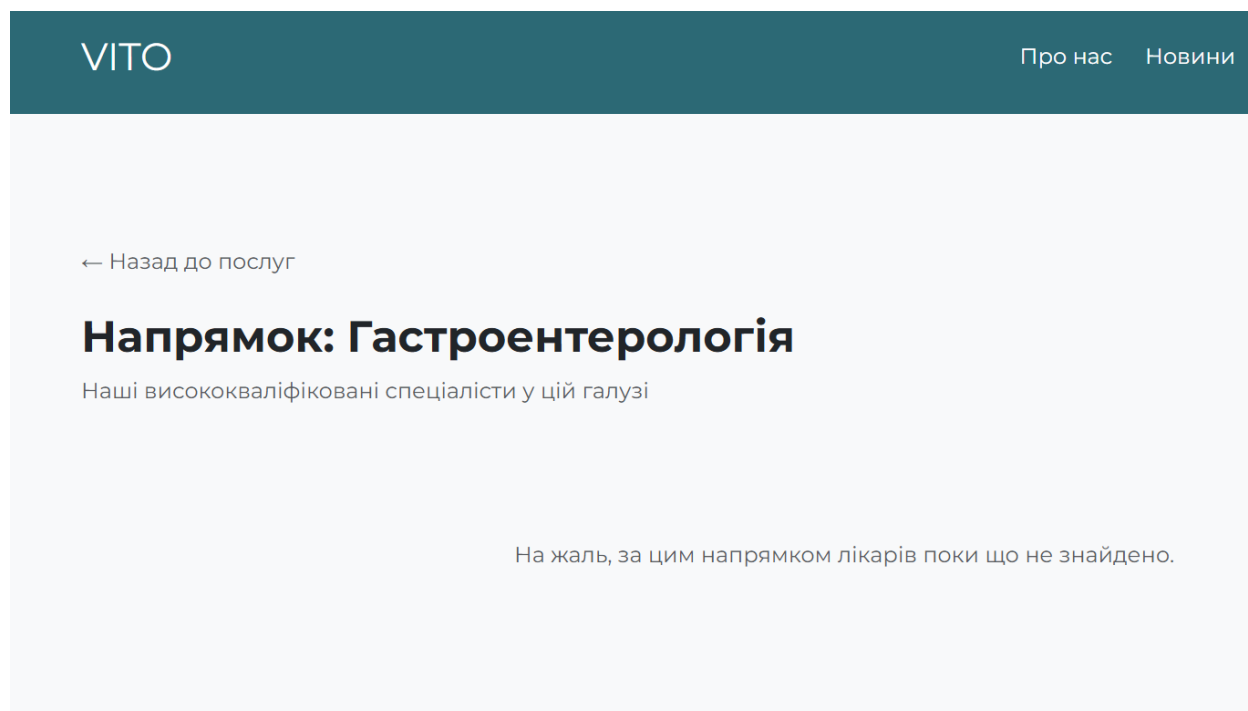


Рис. 3.5. Сторінка деталізації напрямку лікування у випадку відсутності лікарів за даним напрямком

Інтерфейс сторінки передбачає динамічне формування контенту: у випадку наявності спеціалістів система відображає їхні картки з інформацією (рис. 3.6), проте, якщо дані відсутні, передбачено обробку виключних ситуацій — користувачу виводиться відповідне повідомлення про те, що лікарів за цим напрямком поки що не знайдено. Таке рішення підвищує інформативність системи та запобігає виникненню «порожніх» станів сторінки, які могли б ввести користувача в оману щодо працездатності сервісу. Завдяки такому підходу користувач одразу отримує зворотний зв'язок і може повернутися до попереднього етапу для вибору іншого напрямку або скористатися іншими розділами сайту.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

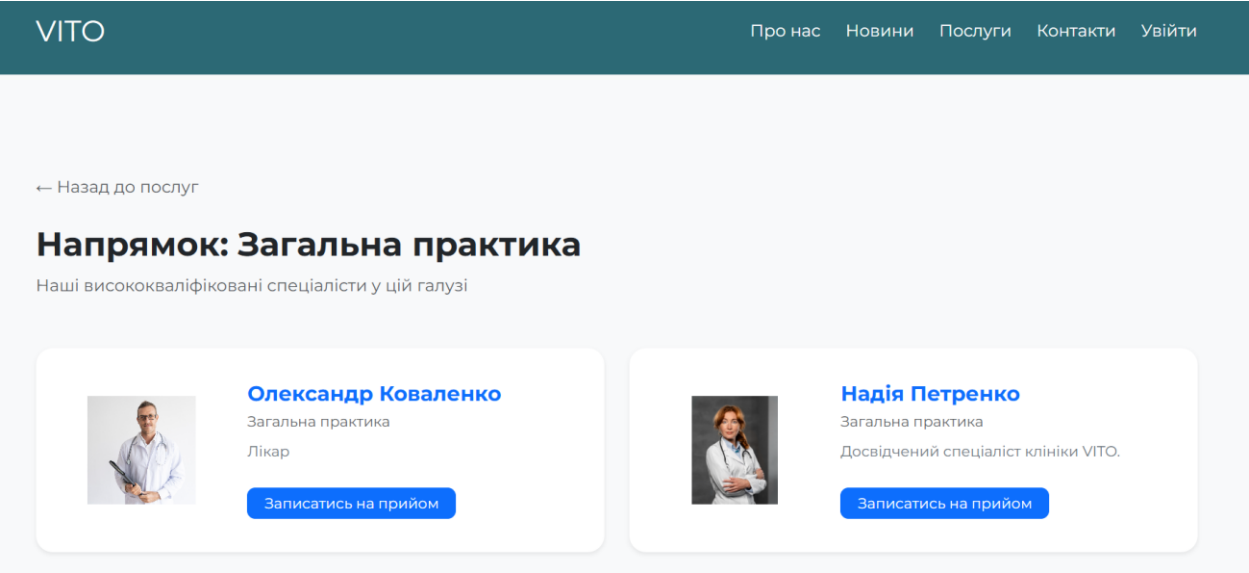


Рис. 3.6. Сторінка деталізації напрямку лікування у випадку наявності лікарів за даним напрямом

Після натискання кнопки «Записатися на прийом» до певного лікаря, пацієнту з’являється форма запису на прийом (рис. 3.7). Форма запису на прийом є центральним функціональним елементом системи, спроектованим для швидкого та безпомилкового бронювання запису на прийом до лікаря.

Запис на прийом

 **Олександр Коваленко**
Загальна практика

Ви записуєтесь як гість. Якщо у вас є акаунт, [увійдіть](#).

Ім'я

Іван

Прізвище

Іванов

Телефон

+380XXXXXXXX

Email (необов'язково)

example@mail.com

Оберіть дату:

10.06

11.06

12.06

10:00

10:30

11:00

11:30

12:00

12:30

13:00

13:30

14:00

15:00

15:30

Рис. 3.7. Форма запису на прийом до лікаря

Верхня частина форми містить інформацію про обраного спеціаліста (ПІБ та спеціалізація), що підтверджує правильність вибору пацієнта перед

внесенням даних. Інтерфейс форми реалізує гнучкий підхід до авторизації користувача: для незареєстрованих відвідувачів (гостей) передбачено поля для введення персональних даних (ім'я, прізвище, телефон та необов'язкова електронна пошта). Для зареєстрованих користувачів система пропонує посилання на вхід («увійдіть»), що дозволяє автоматизувати заповнення даних профілю (рис. 3.8).

Рис. 3.8. Форма автентифікації користувачів

Після успішного бронювання візиту система автоматично генерує електронний талон (рис. 3.9). Інтерфейс талона містить вичерпну інформацію, необхідну для ідентифікації візиту: унікальний номер запису, ПІБ лікаря, його спеціалізацію, а також точну дату та час прийому. Важливим елементом дизайну є текстовий блок із рекомендацією зберегти талон, що спрямовує користувача до виконання наступної логічної дії.

VITO
 Медична клініка

Талон запису №82

Лікар: **Олександр Коваленко**

Спеціальність: **Загальна практика**

Дата: **10.06.2026**

Час: **10:00**

VITO — ми дбаємо про ваш час.
 Збережіть цей талон для візиту до лікаря.

Завантажити талон (PDF)

На головну

Рис. 3.9. Електронний талон запису до лікаря

Для забезпечення зручності пацієнта інтерфейс передбачає функціональну кнопку «Завантажити талон (PDF)», яка дозволяє зберегти дані у форматі, зручному для друку або пред'явлення на мобільному пристрої під час візиту до поліклініки (рис. 3.10).

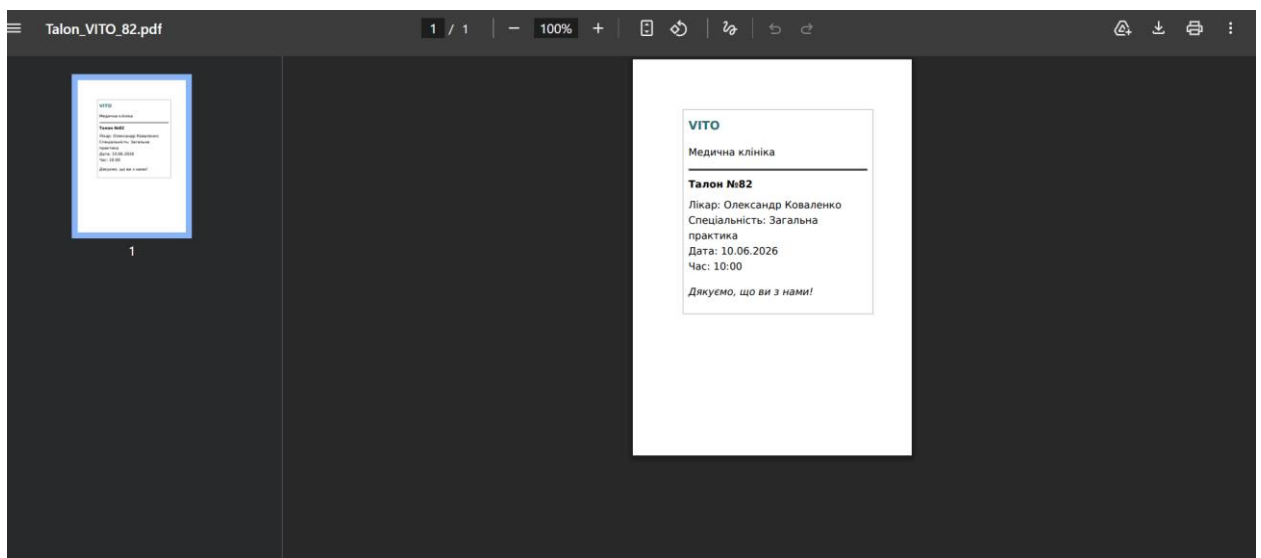


Рис. 3.10. Можливість завантажити/надрукувати електронний талон у pdf форматі

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

У разі входу в систему лікаря, йому відображається його особистий кабінет (рис. 3.11). Особистий кабінет лікаря спроектовано як багатофункціональний робочий простір, що забезпечує ефективне керування професійною діяльністю та взаємодію з пацієнтами. Інтерфейс кабінету поділений на дві основні зони: панель налаштувань профілю та робочу область з оперативними даними.

У лівій частині розміщено блок управління профілем, що включає фотографію спеціаліста, можливість її оновлення та поля для редагування персональних даних (ПІБ, телефон, біографія) [16]. Це дозволяє лікареві самостійно актуалізувати інформацію, яка відображається для пацієнтів.

Права частина інтерфейсу містить таб-панель для перемикання між розділами «Записи пацієнтів» та «Мій графік роботи». Розділ «Записи пацієнтів» реалізовано у вигляді таблиці, яка надає повну візуалізацію робочого розкладу: дата та час прийому, ПІБ пацієнта, причина звернення та статус візиту. Важливою особливістю інтерфейсу є використання колірного кодування статусів (наприклад, зелений для «Підтверджено», сірий для «Скасовано»), що дозволяє лікареві миттєво оцінити стан свого графіку. Для записів, що потребують реакції, реалізовано інтерактивні кнопки підтвердження або скасування прийому, що забезпечує швидке адміністрування візитів безпосередньо з робочої сторінки.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

VITO

Онлайн

На головну

Вийти з акаунту

Загальна практика

Олександр Коваленко

Змінити фото

Налаштування профілю

Ім'я

Олександр

Прізвище

Коваленко

Телефон

+380971234567

Біографія

Лікар

Записи пацієнтів

Мій графік роботи

Ваш поточний графік прийому пацієнтів:

Дата та час	ПІБ Пацієнта	Причина візиту	Статус
03.06.2026 09:00	Кохан Тетяна	Хворію	Підтверджено
10.06.2026 09:00	Гість	Гість: Оля Прекрасна, Тел: +380981234567	Скасовано
10.06.2026 09:30	Гість	Гість: вауа укауук, Тел: +380682760121	Підтверджено
10.06.2026 10:00	Гість	плановий огляд Гість: Ілона Тихенька, Тел: +380678901234	☒ ☐

Рис. 3.11. Особистий кабінет лікаря

Для запобігання помилковим діям лікаря, зокрема випадковому скасуванню візитів, в інтерфейс інтегровано механізм підтвердження через модальне вікно. Якщо пацієнт не з'явився на прийом до лікаря, то лікар може скасувати цей прийом і натиснути відповідну кнопку. Після чого з'являється відповідне повідомлення, яке уточнює чи дійсно лікар хоче позначити цей прийом як той, що не відбувся (рис. 3.12).

Змінити фото

Налаштування профілю

Олександр

Коваленко

Скасування прийому

Ви впевнені, що хочете позначити цей прийом як скасований/не відбувся?

Закрити

Так, скасувати

Дата та час	ПІБ Пацієнта	Причина візиту	Статус
03.06.2026 09:00	Кохан Тетяна	Хворію	Підтверджено

Рис. 3.12. Модальне вікно, яке з'являється у разі скасування запису лікарем

В іншому випадку, якщо прийом відбувся то лікар також відповідним чином може це підтвердити натиснувши на відповідну кнопку. Після чого аналогічно з'являється повідомлення, яке підтверджує поточні дії лікаря (рис. 3.13).

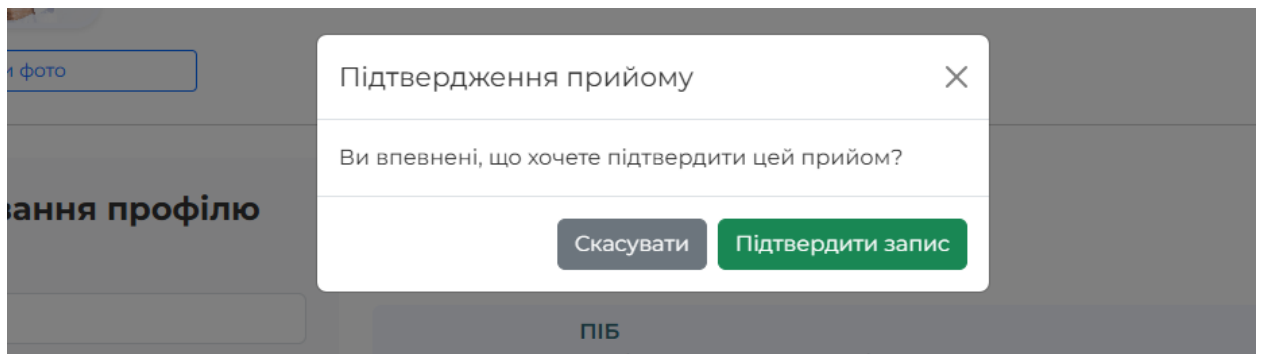


Рис. 3.13. Модальне вікно, яке з'являється у разі підтвердження запису лікарем

У разі входу пацієнта у систему, він потрапляє до особистого кабінету (рис. 3.14). Інтерфейс розділено на дві функціональні зони: блок керування профілем та таблицю поточних записів. У лівій частині розташовано форму редагування контактних даних, яка дозволяє користувачеві самостійно оновлювати інформацію про себе, зокрема ім'я, прізвище, номер телефону, дату народження та адресу, з подальшим збереженням змін через відповідну кнопку. Права частина інтерфейсу містить таблицю «Ваші записи на прийом», де структуровано відображаються дані про заплановані візити: ПІБ лікаря, його спеціалізація, дата, час та причина візиту, а також статус («Confirmed/Cancelled»), що забезпечує користувачеві зручний моніторинг стану його медичних звернень — відбувся прийом чи ні.

VITO

Записатись до лікаря

Вийти з акаунту →

Мій профіль

Пацієнт

Ви можете змінити свої контактні дані нижче:

Ім'я

Тетяна

Прізвище

Кохан

Телефон

+380682760121

Дата народження

22.08.2005

Адреса

Зберегти зміни

Ваші записи на прийом

Лікар	Спеціалізація	Дата та час	Причина візиту	Статус
Олександр Коваленко	Загальна практика	03.06.2026 09:00	Хворію	Confirmed
Олександр Коваленко	Загальна практика	12.06.2026 10:00		Confirmed

Рис. 3.14. Особистий кабінет пацієнта

У разі входу у систему адміністратора, він потрапляє до особистого кабінету (рис. 3.15).

Рис. 3.15. Особистий кабінет адміністратора

Верхня частина інтерфейсу містить інформаційні картки, що відображають загальну статистику системи, а саме: кількість лікарів та пацієнтів, зареєстрованих у базі даних. Центральний функціональний вузол інтерфейсу організовано у вигляді навігаційної панелі з вкладками та відповідними іконками [14], що дозволяють адміністратору виконувати ключові операції. В першу чергу, адміністратор створює розклад (рис. 3.16).

Рис. 3.16. Форма створення розкладу для лікарів

Далі він має можливість згенерувати слоти для запису на основі попередньо створеного розкладу (рис. 3.17). Тут він обирає лікаря, дату та натискає кнопку згенерувати сітку прийомів.

Рис. 3.17. Форма створення слотів для запису

Оскільки саме адміністратор додає лікарів у систему, у його робочій панелі дана форма теж присутня (рис. 3.18). Тут необхідно ввести ім'я, прізвище, пошту, яка і буде логіном для входу у систему, придумати пароль та обрати спеціальність з випадаючого списку.

Рис. 3.18. Форма додавання лікарів у систему

Дана інформаційна система також має сторінку з новинами, тому адміністратор має відповідну форму для додавання новин (рис. 3.19). Адміністратор заповнює відповідні поля, натискає кнопку опублікувати, після чого новина успішно зберігається у базу даних та відображається на відповідній сторінці.

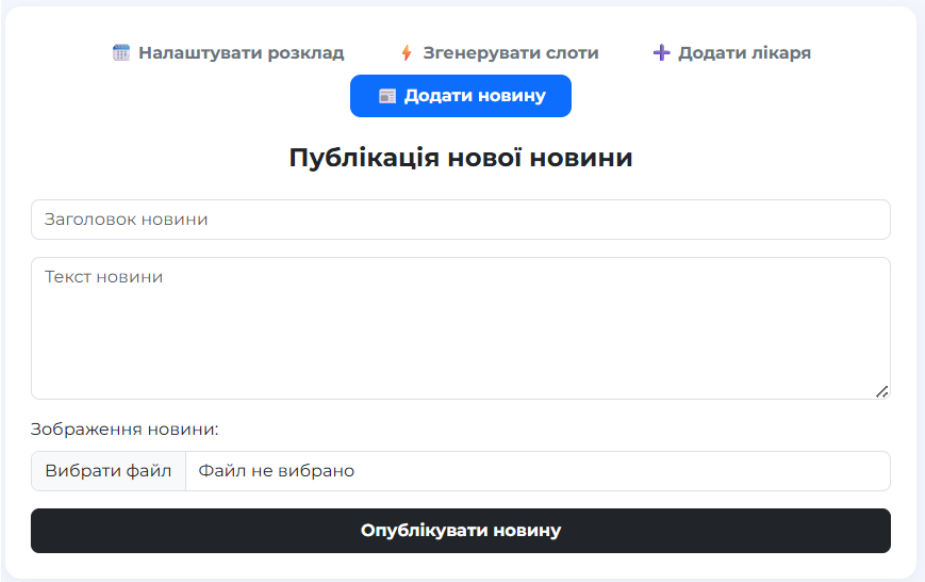


Рис. 3.19 Форма додавання новини

Для забезпечення ефективного адміністрування кадрового складу клініки в системі реалізовано модуль перегляду та керування списком лікарів (рис. 3.20). Інтерфейс представлений у вигляді таблиці, яка містить структуровані дані про кожного зареєстрованого лікаря: унікальний ідентифікатор (ID), ім'я та прізвище, контактні дані (Email, телефон), короткий опис (біографія), а також дату створення облікового запису. таблиця забезпечує адміністративні можливості керування через колонку «Дії», де для кожного запису передбачено кнопку «Видалити» з виразним червоним колірним акцентом. Це дозволяє оперативно коригувати базу даних лікарів у разі зміни персоналу.

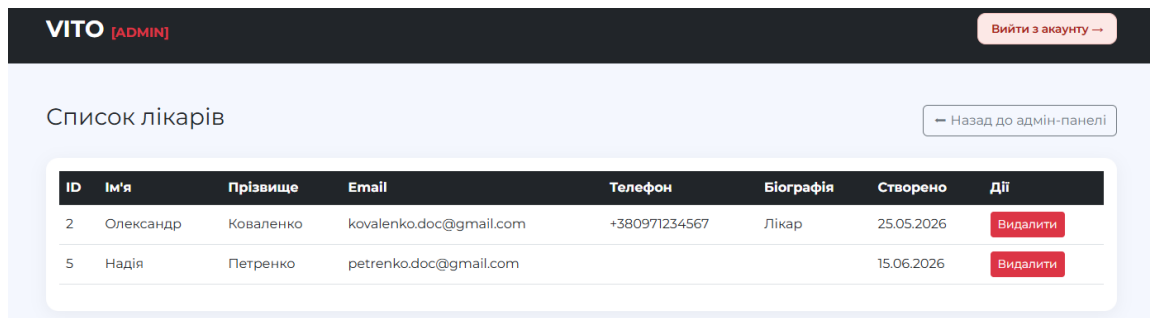


Рис. 3.20 Модуль перегляду та керування списком лікарів системи

Так само адміністратор може контролювати базу пацієнтів в адміністративній частині системи (рис. 3.21).

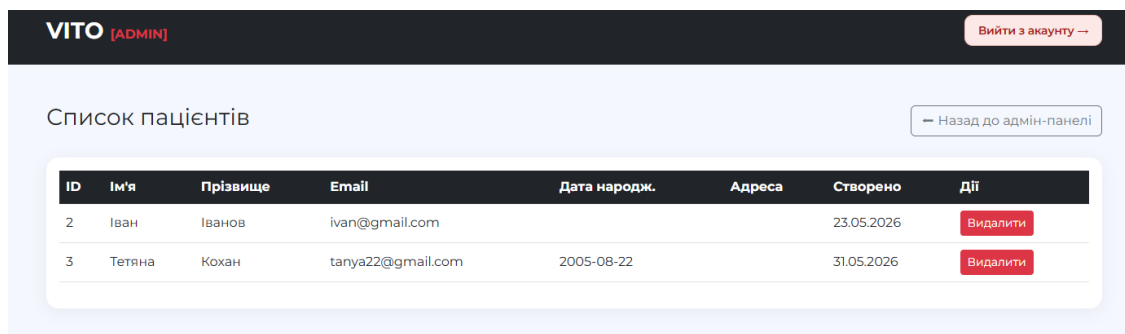


Рис. 3.21. Модуль перегляду та керування списком пацієнтів системи

Якщо пацієнт бажає зареєструватися у системі то при натисканні кнопки увійти, знизу у формі міститься посилання на форму реєстрації і навпаки (рис. 3.22). Форма містить необхідні поля для ідентифікації користувача, включаючи: ім'я, прізвище, email, пароль та повторний пароль. Кнопка «Створити акаунт» має виразний колірний акцент, що спрямовує користувача до виконання цільової дії.

Рис. 3.22. Форма реєстрації

3.3. Тестування роботи

У процесі тестування інформаційної системи автоматизації роботи приватної поліклініки було проведено комплексний аналіз функціональності, коректності відображення інтерфейсу та зручності роботи для різних категорій користувачів. Тестування виконувалося шляхом послідовної перевірки всіх функціональних можливостей системи — від роботи форм та валідації даних до коректності відображення графічних елементів і адаптивності інтерфейсу на різних пристроях. Метою тестування було підтвердження відповідності системи поставленим вимогам, виявлення можливих помилок та оцінка загального рівня якості програмного продукту.

Для проведення тестування було застосовано метод ручного функціонального тестування, який передбачає безпосередню взаємодію тестувальника з системою через графічний інтерфейс користувача. Перевірка охоплювала позитивні сценарії, коректне введення даних із очікуваним результатом та негативні сценарії, введення некоректних, неповних або конфліктуючих даних з метою перевірки стійкості системи.

Першим етапом тестування була перевірка форм автентифікації та реєстрації. Форма входу перевірялася на коректну обробку різних сценаріїв:

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

введення правильних облікових даних, введення неіснуючого логіну, введення неправильного пароля, а також спроба надіслати порожню форму. В усіх випадках система відреагувала очікувано — при некоректних даних користувач отримував відповідне повідомлення про помилку, а порожні поля не дозволяли надіслати форму на сервер (рис. 3.23).

The screenshot shows a login interface for 'VITO'. The title is 'Вхід у систему'. There is an 'Email' label above a text input field containing 'olia'. A red error message box is displayed below the input field, stating: 'Електронна адреса має містити знак "@". В електронній адресі "olia" знака "@" немає.' Below the input field is a password field with masked characters. At the bottom is a dark blue button labeled 'Увійти' and a link 'Немаєте акаунту? Зареєструватися'.

Рис. 3.23. Перевірка коректності введених даних

Аналогічно перевірялася форма реєстрації нового пацієнта: тестувалося введення некоректного формату електронної пошти, а також спроба зареєструватися з уже існуючою в системі електронною адресою (рис. 3.24). Система коректно виявляла кожну з цих ситуацій та інформувала користувача про характер помилки — зокрема відображала відповідні підказки безпосередньо під проблемними полями.

The image shows a web form for creating an account on the VITO platform. At the top, the logo 'VITO' is displayed in blue. Below it, the title 'Створити акаунт' (Create account) is centered. A red error message box states: 'Помилка при реєстрації (користувач з таким email вже існує)' (Error during registration (user with this email already exists)). The form includes input fields for 'Ім'я' (Name), 'Прізвище' (Surname), 'Email', 'Пароль' (Password), and 'Підтвердження' (Confirmation). A dark blue button labeled 'Створити акаунт' is positioned below the inputs. At the bottom, a link says 'Вже маєте акаунт? Увійти' (Already have an account? Log in).

Рис. 3.24. Помилка при реєстрації користувача з вже існуючим таким самим email

Окремо тестувалася форма запису пацієнта на прийом. Перевірялося коректне відображення доступних часових слотів для обраного лікаря з урахуванням його розкладу роботи. Особлива увага приділялася перевірці того, що пацієнт не має можливості обрати вже зайнятий слот або час, який уже минув. Система коректно фільтрувала недоступні часові проміжки та відображала лише актуальні варіанти.

Після успішного запису перевірялося коректне формування електронного талона у форматі PDF — документ містив усі необхідні дані: прізвище та ім'я, спеціалізацію лікаря, дату та час прийому, а також унікальний ідентифікатор запису.

Тестування адміністративних функцій включало перевірку операцій створення, перегляду та видалення записів. Перевірялося коректне додавання нового лікаря до системи із заповненням усіх обов'язкових полів — прізвища,

імені, по батькові, спеціалізації, контактної інформації та завантаження фотографії. Спроба зберегти форму з незаповненими обов'язковими полями не давала результату — система виводила відповідні повідомлення про помилки. Тестувалося створення розкладу роботи лікаря та автоматичне формування часових слотів на основі заданих параметрів — робочих днів, часу початку та завершення прийому, а також тривалості одного сеансу. Після налаштування розкладу перевірялося їх коректне відображення на сторінці запису для пацієнтів. Усі сформовані слоти відповідали введеним параметрам, жодних дублікатів або некоректних часових проміжків виявлено не було.

Окремо перевірялася функція управління статусами записів на прийом із боку лікаря. Тестувалася можливість лікаря підтвердити або скасувати запис пацієнта через особистий кабінет. Після зміни статусу перевірялося його коректне відображення в особистому кабінеті пацієнта. Таким чином пацієнт у режимі реального часу отримує актуальну інформацію щодо свого запису: чи підтверджено прийом лікарем, чи його було скасовано. Система коректно обробляла обидва сценарії та відображала відповідні позначки у розділі записів пацієнта.

Окремим напрямком тестування була перевірка коректності відображення графічних елементів інтерфейсу. Перевірялося відображення фотографій лікарів у картках та особистих кабінетах, іконок та інших графічних елементів на всіх сторінках системи. Усі зображення завантажувалися коректно та відображалися у відповідних розмірах без спотворень. Окремо перевірялася поведінка інтерфейсу у разі відсутності завантаженого фото — система відображала стандартне зображення-заглушку.

Тестувалося коректне відображення сторінок при різній роздільній здатності екрану. Система зберігала читабельність та зручність використання на всіх розмірах екрана завдяки адаптивній верстці на основі Bootstrap.

Також перевірялася кросбраузерна сумісність системи. Тестування проводилося у браузерах Google Chrome та Microsoft Edge [13, 26]. В усіх

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

перевірених браузерів інтерфейс відображався коректно, функціональність зберігалася в повному обсязі, відмінностей у відображенні критичних елементів виявлено не було.

Система демонструє високу швидкість відгуку при виконанні основних операцій. Завантаження сторінок та перехід між розділами здійснюється без відчутних затримок, що досягається завдяки оптимізованій структурі запитів до бази даних. Генерація динамічного контенту — перегляд списків лікарів, формування електронного талона, оновлення статусів записів — не викликає відчутних затримок і забезпечує комфортний темп роботи як для пацієнтів, так і для адміністративного персоналу.

Дизайн системи розроблено з урахуванням принципів інтуїтивної навігації. Використана кольорова гама є приємною для сприйняття, не викликає втоми при тривалій роботі та дозволяє користувачу легко виокремити головні елементи керування. Структура сторінок є логічною і послідовною — користувач може знайти потрібний розділ без необхідності вивчати інструкцію. Всі інтерактивні елементи (кнопки, поля вводу, випадаючі списки) мають достатній розмір для зручної взаємодії.

За результатами проведеного ручного тестування можна зробити висновок, що розроблена інформаційна система є стабільною та надійною. Система коректно обробляє як стандартні, так і нестандартні сценарії введення даних, грамотно інформує користувача про помилки, не допускаючи втрати або пошкодження даних. Усі перевірені функціональні модулі — автентифікація, реєстрація, запис на прийом, адміністративне управління — відповідають поставленим функціональним вимогам та працюють без критичних збоїв.

Висновки до третього розділу

У третьому розділі було проведено повний цикл проєктування інтерфейсу інформаційної системи автоматизації роботи приватної поліклініки.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

У підрозділі 3.1 розроблено алгоритм розгортання та налаштування системи. Визначено основні кроки для підготовки програмного середовища, конфігурації підключення до бази даних та первинного налаштування робочих параметрів сервера, що забезпечує готовність інформаційної системи до експлуатації.

У підрозділі 3.2 описано архітектуру інтерфейсу користувача. Завдяки використанню модульного підходу створено інтуїтивно зрозуміле середовище, розділене на функціональні зони для пацієнтів, лікарів та адміністраторів. Зокрема, розроблено та детально описано форми автентифікації, особисті кабінети та інструменти адміністрування, що дозволяє користувачам ефективно виконувати свої завдання.

У підрозділі 3.3 проведено тестування функціональності та ергономіки системи. Підтверджено коректність обробки даних у формах, надійність роботи механізмів підтвердження критичних дій та високу швидкодію системи. Оцінка UI/UX довела відповідність інтерфейсу принципам зручності та доступності для різних категорій користувачів.

Таким чином, розроблені та протестовані інтерфейсні рішення дозволили створити стабільну та ефективну інформаційну систему, що повністю задовольняє вимоги автоматизації процесів приватної поліклініки.

ВИСНОВКИ

У даній кваліфікаційній роботі бакалавра було спроектовано та розроблено інформаційну систему автоматизації роботи приватної поліклініки. Основною метою роботи було створення програмного комплексу, що забезпечує оптимізацію процесу онлайн-запису пацієнтів та ефективне ведення обліку відвідувань.

У процесі виконання роботи було проведено аналіз предметної області, який дозволив виявити критичні точки в роботі реєстратури та організації розкладу, що безпосередньо обґрунтувало необхідність впровадження автоматизованої системи. Для реалізації програмного продукту було обрано сучасний стек технологій, що включає мову програмування PHP з використанням архітектури MVC, яка забезпечує чітке розділення логіки обробки даних, бізнес-логіки та представлення. Для побудови інтерфейсу було використано інструменти фронтенд-розробки, зокрема фреймворк Bootstrap, що гарантує адаптивність та зручність відображення системи на різних пристроях. Управління даними здійснюється за допомогою реляційної бази даних, що забезпечує цілісність та високу швидкість обробки запитів. Спроектований інтерфейс користувача, адаптований окремо для пацієнтів, лікарів та адміністратора, гарантує інтуїтивно зрозумілу навігацію.

Окрім цього, у системі було реалізовано механізм розмежування прав доступу, що забезпечує захищеність персональних даних, а проведене тестування підтвердило стабільність роботи всіх форм, коректність валідації даних та високу швидкодію інтерфейсу. Усі поставлені завдання були виконані в повному обсязі, що відповідає вимогам до функціональності системи.

Практичне значення одержаних результатів полягає у створенні готового до впровадження програмного продукту, який автоматизує ключові бізнес-процеси приватної поліклініки. Впровадження цієї системи дозволяє мінімізувати вплив «людського фактору», уникнути помилкових накладок у розкладі лікарів та суттєво зменшити навантаження на реєстратуру.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

Прозорість адміністративних процесів та доступність онлайн-сервісів для пацієнтів безпосередньо впливають на підвищення якості медичного обслуговування, що підтверджує ефективність розробленого рішення.

Перспективи подальшого розвитку програмного комплексу спрямовані на перетворення його з інструменту запису на повноцінну медичну екосистему. Зокрема, створення модуля електронної медичної картки, що дозволить зберігати повну історію хвороби пацієнта, результати аналізів та призначень у цифровому вигляді, забезпечуючи лікарям швидкий доступ до інформації про стан здоров'я пацієнта, що в цілому сприятиме комплексній цифровізації послуг медичного закладу.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		81

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moz.gov.ua. Цифровізація медицини у 2024 році: підсумки та ключові досягнення. URL: <https://moz.gov.ua/uk/cifrovizaciya-medicini-u-2024-roci-pidsumki-ta-klyuchovi-dosyagnennya> (дата звернення: 11.04.2026).
2. Moz.gov.ua. Як медичному закладу підійти до вибору медичної інформаційної системи. URL: <https://moz.gov.ua/article/news/jak-medichnomu-zakladu-pidijti-do-viboru-medichnoi-informacijnoi-sistemi> (дата звернення: 11.04.2026).
3. Пасічник О. Г., Пасічник О. В., Стеценко І. В. Основи веб-дизайну: навч. посіб. Київ: Вид. група ВНУ, 2009. 336 с.
4. Пономаренко В. С. Проектування інформаційних систем: навчальний посібник. Київ: ВЦ «Академія», 2002. 461 с.
5. Трофименко О. Г. та ін. Організація баз даних: навчальний посібник. Одеса: Фенікс, 2019. 246 с.
6. Харів Н. О. Базы даних та інформаційні системи: навчальний посібник. Рівне: НУВГП, 2018. 127 с.
7. Apache HTTP Server Documentation. URL: <https://httpd.apache.org/docs/> (дата звернення: 02.05.2026).
8. Benefit of using MVC. URL: <https://www.geeksforgeeks.org/software-engineering/benefit-of-using-mvc/> (дата звернення: 02.05.2026).
9. Bootstrap Documentation. URL: <https://getbootstrap.com/> (дата звернення: 02.05.2026).
10. Composer — менеджер залежностей для PHP. URL: <https://getcomposer.org/> (дата звернення: 30.05.2026).
11. Dompdf : документація проєкту. URL: <https://github.com/dompdf/dompdf> (дата звернення: 30.05.2026).
12. Draw.io — безкоштовний інструмент для побудови UML-діаграм онлайн. URL: <https://app.diagrams.net/> (дата звернення: 14.06.2026).

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Emulate mobile devices (Device Emulation) : Microsoft Edge Developer documentation. URL: <https://learn.microsoft.com/en-us/microsoft-edge/devtools/device-mode/> (дата звернення: 14.06.2026).

14. Full Emoji List : Unicode® Technical Standard. URL: <https://unicode.org/emoji/charts/full-emoji-list.html> (дата звернення: 14.06.2026).

15. Git Documentation. URL: <https://git-scm.com/doc> (дата звернення: 14.06.2026).

16. jQuery Documentation. URL: <https://api.jquery.com/> (дата звернення: 14.06.2026).

17. MDN Web Docs — CSS. URL: <https://developer.mozilla.org/uk/docs/Web/CSS> (дата звернення: 02.05.2026).

18. MDN Web Docs — HTML. URL: <https://developer.mozilla.org/uk/docs/Web/HTML> (дата звернення: 02.05.2026).

19. MDN Web Docs — JavaScript. URL: <https://developer.mozilla.org/uk/docs/Web/JavaScript> (дата звернення: 02.05.2026).

20. MySQL Documentation. URL: <https://dev.mysql.com/doc/> (дата звернення: 02.05.2026).

21. Nixon R. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites. O'Reilly Media, 2021. 823 p.

22. PHP Documentation. URL: <https://www.php.net/manual/en/> (дата звернення: 02.05.2026).

23. PHP-FIG — стандарти взаємодії PHP-компонентів (PSR). URL: <https://www.php-fig.org/psr/> (дата звернення: 02.05.2026).

24. PHP: The Right Way — сучасні практики та стандарти розробки на PHP. URL: <https://phptherightway.com/> (дата звернення: 02.05.2026).

25. phpMyAdmin Documentation. URL: <https://docs.phpmyadmin.net/> (дата звернення: 02.05.2026).

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

26. Simulate mobile devices with device mode : Chrome DevTools. URL: <https://developer.chrome.com/docs/devtools/device-mode/> (дата звернення: 14.06.2026).

27. UML 2.5 Diagrams Overview. URL: <https://www.uml-diagrams.org/uml-25-diagrams.html> (дата звернення: 14.06.2026).

28. Visual Studio Code — Documentation. URL: <https://code.visualstudio.com/> (дата звернення: 02.05.2026).

29. W3Schools — PHP Tutorial. URL: <https://www.w3schools.com/php/> (дата звернення: 02.05.2026).

30. XAMPP — Documentation. URL: <https://www.apachefriends.org/> (дата звернення: 02.05.2026).

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б-121-26-ІПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг класу Appointment — модуль запису на прийом:

```

<?php

class Appointment
{
    private $pdo;

    public function __construct()
    {
        $dbConfig = dirname(__DIR__) . '/config/db.php';
        require_once $dbConfig;
        $this->pdo = connectDB();
    }

    public function getAvailableSlotsForPatient($doctor_id, $date)
    {
        $sql = "SELECT appointment_id, appointment_datetime
        FROM appointments
        WHERE doctor_id = :doctor_id
        AND DATE(appointment_datetime) = :date
        AND status = 'Available'
        ORDER BY appointment_datetime ASC";

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([':doctor_id' => $doctor_id, ':date' => $date]);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    public function bookAppointment($appointment_id, $patient_id,
    $reason_for_visit = '')
    {
        $sql = "UPDATE appointments
        SET status = 'Booked',
        patient_id = :patient_id,
        reason_for_visit = :reason_for_visit
        WHERE appointment_id = :appointment_id AND status = 'Available'";

        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([
            ':patient_id' => $patient_id,
            ':reason_for_visit' => $reason_for_visit,
            ':appointment_id' => $appointment_id
        ]);
    }

    public function bookAsGuest($appointment_id, $first_name, $last_name,
    $phone, $reason)
    {
        $sql = "UPDATE appointments
        SET status = 'Booked',
        reason = :reason,
        guest_info = :guest_info
        WHERE appointment_id = :id AND status = 'Available'";
        $guest_data = "$first_name $last_name, Тел: $phone";

        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([
            ':id' => $appointment_id,
            ':reason' => $reason,
            ':guest_info' => $guest_data
        ]);
    }
}

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    });
}

public function updateAppointmentStatus($id, $status)
{
    $sql = "UPDATE appointments SET status = :status WHERE appointment_id
= :id";

    $stmt = $this->pdo->prepare($sql);
    return $stmt->execute([':id' => $id, ':status' => $status]);
}

public function getFullAppointmentDetails($appointment_id)
{
    $sql = "SELECT a.*,
                d.first_name AS doc_first,
                d.last_name AS doc_last,
                s.name AS specialty_name
            FROM appointments a
            JOIN doctors d ON a.doctor_id = d.doctor_id
            JOIN specialties s ON d.specialty_id = s.specialty_id
            WHERE a.appointment_id = :app_id
            LIMIT 1";

    $stmt = $this->pdo->prepare($sql);
    $stmt->execute([':app_id' => $appointment_id]);
    return $stmt->fetch(PDO::FETCH_ASSOC);
}
}

```

Лістинг класу User — модуль управління користувачами:

```

<?php

class User
{
    private $pdo;

    public function __construct()
    {
        $dbConfig = dirname(__DIR__) . '/config/db.php';
        if (file_exists($dbConfig)) {
            require_once $dbConfig;
            $this->pdo = connectDB();
        } else {
            die("Помилка: Не вдалося знайти db.php за шляхом: " . $dbConfig);
        }
    }

    public function getPatientProfile($patientId)
    {
        $sql = "SELECT * FROM patients WHERE patient_id = :patient_id LIMIT
1";

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([':patient_id' => $patientId]);
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    public function getPatientAppointments($patientId)
    {
        $sql = "SELECT a.*, d.first_name AS doc_first, d.last_name AS
doc_last, s.name AS specialty
            FROM appointments a
            JOIN doctors d ON a.doctor_id = d.doctor_id

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        JOIN specialties s ON d.specialty_id = s.specialty_id
        WHERE a.patient_id = :patient_id
        ORDER BY a.appointment_datetime ASC";
$stmt = $this->pdo->prepare($sql);
$stmt->execute([':patient_id' => $patientId]);
return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

public function getDoctorData($doctorId)
{
    $stmt = $this->pdo->prepare("
        SELECT d.*, s.name as specialty
        FROM doctors d
        JOIN specialties s ON d.specialty_id = s.specialty_id
        WHERE d.doctor_id = :did
    ");
    $stmt->execute([':did' => $doctorId]);
    return $stmt->fetch(PDO::FETCH_ASSOC);
}

public function getDoctorByUserId($userId)
{
    $stmt = $this->pdo->prepare("SELECT doctor_id FROM doctors WHERE
user_id = :uid LIMIT 1");
    $stmt->execute([':uid' => $userId]);
    return $stmt->fetch(PDO::FETCH_ASSOC);
}

public function getDoctorAppointments($doctorId)
{
    $sql = "SELECT a.*, p.first_name AS pat_first, p.last_name AS pat_last
        FROM appointments a
        LEFT JOIN patients p ON a.patient_id = p.patient_id
        WHERE a.doctor_id = :doctor_id
        AND a.status != 'Available'
        ORDER BY a.appointment_datetime ASC";

    $stmt = $this->pdo->prepare($sql);
    $stmt->execute([':doctor_id' => $doctorId]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

public function getPatientByUserId($userId)
{
    $stmt = $this->pdo->prepare("SELECT patient_id FROM patients WHERE
user_id = :uid LIMIT 1");
    $stmt->execute([':uid' => $userId]);
    return $stmt->fetch(PDO::FETCH_ASSOC);
}

public function getPdo()
{
    return $this->pdo;
}

public function getDoctorsList()
{
    $sql = "SELECT d.doctor_id, d.first_name, d.last_name
        FROM doctors d";

    $stmt = $this->pdo->query($sql);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public function getCountDoctors()
    {
        $stmt = $this->pdo->query("SELECT COUNT(*) FROM doctors");
        return $stmt->fetchColumn();
    }

    public function getCountPatients()
    {
        $stmt = $this->pdo->query("SELECT COUNT(*) FROM patients");
        return $stmt->fetchColumn();
    }

    public function getAllDoctors()
    {
        return $this->pdo->query("SELECT d.*, u.email
                                FROM doctors d
                                JOIN users u ON d.user_id = u.user_id")->
>fetchAll(PDO::FETCH_ASSOC);
    }

    public function getAllPatients()
    {
        return $this->pdo->query("SELECT p.*, u.email
                                FROM patients p
                                JOIN users u ON p.user_id = u.user_id")->
>fetchAll(PDO::FETCH_ASSOC);
    }

    public function deleteUser($userId)
    {
        $stmt = $this->pdo->prepare("DELETE FROM users WHERE user_id = ?");
        return $stmt->execute([$userId]);
    }

    public function updatePatientProfile($patientId, $data)
    {
        $stmt = $this->pdo->prepare("UPDATE patients SET first_name=?,
last_name=?, phone=?, date_of_birth=?, address=? WHERE patient_id=?");
        return $stmt->execute([$data['first_name'], $data['last_name'],
$data['phone'], $data['date_of_birth'], $data['address'], $patientId]);
    }

    public function updateDoctorProfile($doctorId, $data)
    {
        $stmt = $this->pdo->prepare("UPDATE doctors
                                SET first_name = ?, last_name = ?, phone =
?, biography = ?
                                WHERE doctor_id = ?");
        return $stmt->execute([
            $data['first_name'],
            $data['last_name'],
            $data['phone'],
            $data['biography'],
            $doctorId
        ]);
    }

    public function updateDoctorPhoto($doctorId, $filePath)
    {

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $stmt = $this->pdo->prepare("UPDATE doctors SET profile_picture = ?
WHERE doctor_id = ?");
        return $stmt->execute([$filePath, $doctorId]);
    }
}
?>

```

Клас AppointmentController — модуль підтвердження та формування електронного талона.

```

class AppointmentController
{
    private $appointmentModel;
    private $pdo;

    public function __construct($pdo)
    {
        $this->pdo = $pdo;
        $this->appointmentModel = new Appointment($pdo);
    }

    public function showBookingPage($doctor_id)
    {
        $stmt = $this->pdo->prepare("SELECT d.*, s.name as specialty_name
                                FROM doctors d
                                JOIN specialties s ON d.specialty_id =
s.specialty_id
                                WHERE d.doctor_id = ?");
        $stmt->execute([$doctor_id]);
        $doctor = $stmt->fetch(PDO::FETCH_ASSOC);
        $is_logged_in = isset($_SESSION['patient_id']);

        $availableDates = [];
        if ($doctor) {
            $stmt = $this->pdo->prepare("SELECT DISTINCT
DATE(appointment_datetime) as date
                                FROM appointments
                                WHERE doctor_id = ? AND status =
'Available'
                                ORDER BY date ASC");
            $stmt->execute([$doctor_id]);
            $availableDates = $stmt->fetchAll(PDO::FETCH_COLUMN);
        }

        $selectedDate = $_GET['date'] ?? ($availableDates[0] ?? null);
        $slots = $selectedDate ? $this->appointmentModel-
>getAvailableSlotsForPatient($doctor_id, $selectedDate) : [];
        require_once '../app/views/appointments/book.php';
    }

    private function isSlotBooked($doctor_id, $datetime)
    {
        $stmt = $this->pdo->prepare("SELECT 1 FROM appointments WHERE
doctor_id = ? AND appointment_datetime = ? AND status = 'Booked'");
        $stmt->execute([$doctor_id, $datetime]);
        return $stmt->fetch();
    }

    public function handleBooking()
    {
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $doctor_id = $_POST['doctor_id'];

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $appointment_id = $_POST['appointment_id'];
        $reason = $_POST['reason_for_visit'] ?? '';

        if (isset($_SESSION['patient_id'])) {
            $success = $this->appointmentModel->bookAppointment($appointment_id, $_SESSION['patient_id'], $reason);
        } else {
            $first_name = $_POST['guest_first_name'];
            $last_name = $_POST['guest_last_name'];
            $phone = $_POST['guest_phone'];
            $full_reason = $reason . " | Гість: $first_name $last_name,
Тел: $phone";

            $success = $this->appointmentModel->bookAppointment($appointment_id, null, $full_reason);
        }

        if ($success) {
            $_SESSION['success_message'] = "Дякуємо! Ваш запис
підтверджено.";
        } else {
            $_SESSION['error_message'] = "Вибачте, цей час уже зайнято.";
        }

        header("Location:
index.php?action=appointment_confirmation&appointment_id=" . $appointment_id);
        exit();
    }

    public function showConfirmation($appointment_id)
    {
        $appointment = $this->appointmentModel->getFullAppointmentDetails($appointment_id);

        if (!$appointment) {
            die("Помилка: Запис не знайдено.");
        }

        $datetime = new DateTime($appointment['appointment_datetime']);
        $formattedDate = $datetime->format('d.m.Y');
        $formattedTime = $datetime->format('H:i');

        require_once
'../app/views/appointments/appointment_confirmation.php';
    }

    public function downloadTicket($appointment_id)
    {
        $appointment = $this->appointmentModel->getFullAppointmentDetails($appointment_id);

        if (!$appointment) {
            die("Запис не знайдено.");
        }

        $datetime = new DateTime($appointment['appointment_datetime']);
        $formattedDate = $datetime->format('d.m.Y');
        $formattedTime = $datetime->format('H:i');
        ob_start();
        include __DIR__ . '/../views/appointments/pdf_template.php';
        $html = ob_get_clean();
        PdfGenerator::download($html, "Talon_VITO_$appointment_id.pdf");
        exit();
    }

```

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас Specialty — модуль спеціальностей та лікарів:

```
<?php
class Specialty
{
    private $pdo;

    public function __construct()
    {
        require_once '../app/config/db.php';
        $this->pdo = connectDB();
    }

    public function getAllSpecialties()
    {
        $stmt = $this->pdo->query("SELECT * FROM specialties ORDER BY name
ASC");
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    public function getDoctorsBySpecialty($specialty_id)
    {
        $sql = "SELECT d.doctor_id, d.first_name, d.last_name, u.username,
d.biography, d.profile_picture
FROM doctors d
JOIN users u ON d.user_id = u.user_id
WHERE d.specialty_id = :specialty_id";

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([':specialty_id' => $specialty_id]);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
}
```

Клас PdfGenerator — модуль генерації PDF-талонів:

```
<?php
require_once __DIR__ . '/../../vendor/autoload.php';
use Dompdf\Dompdf;

class PdfGenerator
{
    public static function download($html, $filename = "talon.pdf")
    {
        $dompdf = new Dompdf();
        $options = $dompdf->getOptions();
        $options->set('defaultFont', 'DejaVu Sans');
        $options->set('isRemoteEnabled', true);
        $dompdf->setOptions($options);

        $dompdf->loadHtml($html, 'UTF-8');
        $dompdf->setPaper('A7', 'portrait');

        $dompdf->render();
        $dompdf->stream($filename, ["Attachment" => true]);
    }
}
```