Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка вебзастосунку з використанням технології WebRTC для проведення медичних онлайн-консультацій

Виконав: студент IV курсу, групи СП-42
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник (підпис) Цуприк Г.Б (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р
(підпис) (прізвище та ініціали)
« 6 » квітня 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Павлову Дмитру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку з використанням технології WebRTC для проведення медичних онлайн-консультацій.

Керівник роботи к.т.н., доцент Цуприк Г.Б.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *6* » *квітня* 2026 року № *4/9-170*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.
1 Аналіз вимог до вебзастосунку.
2 Проєктування вебзастосунку
3 Тестування та верифікація вебзастосунку
4 Безпека життєдіяльності, основи охорони праці.
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 6 квітня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу 1 «Аналіз вимог до вебзастосунку»		
4	Виконання розділу 2 «Проєктування та розробка вебзастосунку»		
5	Виконання розділу 3 «Тестування та верифікація вебзастосунку»		
6	Виконання розділу 4 «Безпека життєдіяльності, основи охорони праці»		
7	Оформлення пояснювальної записки		
8	Оформлення графічного та презентаційного матеріалу		
9	Перевірка на академічний плагіат		
10	Попередній захист кваліфікаційної роботи		
11	Захист		

Студент _____ Павлов Д.І
(підпис) (прізвище та ініціали)

Керівник роботи _____ Цуприк Г.Б
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка вебзастосунку з використанням технології WebRTC для проведення медичних онлайн-консультацій // Кваліфікаційна робота освітнього рівня «бакалавр» // Павлов Дмитро Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // С. - 83, табл. - 0, рис. - 22, бібліогр. – 27, додат. – 3.

Ключові слова: телемедицина; WebRTC; P2P-передача; відеоконсультація; вебзастосунок; React; Node.js; Socket.io; MySQL.

Кваліфікаційна робота присвячена дослідженню та розробці вебзастосунку для проведення віддалених медичних онлайн-консультацій із використанням технології WebRTC. Актуальність роботи обумовлена необхідністю створення безпечної та ефективної системи відеозв'язку.

У першому розділі проведено аналіз предметної області, та обґрунтовано доцільність використання децентралізованої технології WebRTC.

У другому розділі досліджено вимоги до системи, спроєктовано її архітектуру, структуру бази даних. Обґрунтовано вибір технологічного стеку та описано реалізацію основних модулів вебзастосунку.

У третьому розділі описано процес реалізації програмної системи, наведено результати тестування, а також досліджено питання розгортання програмного забезпечення.

У четвертому розділі розглянуто питання безпеки життєдіяльності та основ охорони праці під час експлуатації комп'ютерної техніки й інформаційних систем.

Об'єкт дослідження — процес надання та отримання віддалених медичних консультацій.

Предмет дослідження — методи, технології та програмні засоби побудови телемедичних інформаційних систем із використанням React, Node.js, WebRTC та Socket.io.

ABSTRACT

Development of a web application using WebRTC technology for conducting medical online consultations // Bachelor's Qualification Thesis // Dmytro Ihorovych Pavlov // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SP-42 // Ternopil, 2026 // Pages - 83, tables - 0, figures - 22, references - 27, appendices -3.

Keywords: telemedicine; WebRTC; P2P-transmission; video consultation; web application; React; Node.js; Socket.io; MySQL.

The qualification work is devoted to the research and development of a web application for conducting remote medical online consultations using WebRTC technology. The relevance of the work is due to the need to create a safe and effective video communication system.

The first section analyzes the subject area, and justifies the feasibility of using decentralized WebRTC technology.

The second section examines the requirements for the system, designs its architecture, database structure. Justifies the choice of the technological stack and describes the implementation of the main modules of the web application.

The third section describes the process of implementing the software system, presents the results of testing, and also examines the issue of software deployment.

The fourth section considers the issue of life safety and the basics of labor protection during the operation of computer equipment and information systems.

The object of the study is the process of providing and receiving remote medical consultations.

The subject of the study is methods, technologies and software tools for building telemedical information systems using React, Node.js, WebRTC and Socket.io.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Інтерфейс прикладного програмування (Application Programming Interface).

DOM – Об'єктна модель документа (Document Object Model).

HTTP – Протокол передачі гіпертексту (HyperText Transfer Protocol).

HTTPS – Безпечний протокол передачі гіпертексту (HyperText Transfer Protocol Secure).

ICE – Інтерактивне встановлення з'єднання (Interactive Connectivity Establishment).

IDE – Інтегроване середовище розробки (Integrated Development Environment).

JSON – Об'єктна нотація JavaScript (JavaScript Object Notation).

NAT – Перетворення мережевих адрес (Network Address Translation).

P2P – Однорангова мережа, передача від вузла до вузла (Peer-to-Peer).

SDP – Протокол опису сеансу (Session Description Protocol).

STUN – Утиліти обходу сеансів для NAT (Session Traversal Utilities for NAT).

TCP – Протокол керування передачею (Transmission Control Protocol).

TURN – Обхід з використанням ретрансляторів навколо NAT (Traversal Using Relays around NAT).

UDP – Протокол користувацьких датаграм (User Datagram Protocol).

UI – Користувацький інтерфейс (User Interface).

URL – Уніфікований покажчик ресурсу (Uniform Resource Locator).

UX – Досвід користувача (User Experience).

WebRTC – Вебкомунікації в реальному часі (Web Real-Time Communication).

ЗМІСТ

ВСТУП

У сучасному світі телемедицина стала невід'ємною частиною системи охорони здоров'я, забезпечуючи швидкий та безпечний доступ пацієнтів до медичних послуг незалежно від їхнього місцезнаходження. Проте існуючі рішення часто стикаються з проблемою перевантаження серверів через зберігання великих обсягів медичних даних (результатів аналізів, знімків МРТ) та питаннями конфіденційності. Розробка телемедичної платформи з використанням децентралізованої технології WebRTC DataChannel для P2P-передачі файлів вирішує ці проблеми, дозволяючи обмінюватися даними безпосередньо між клієнтами, що суттєво економить серверні ресурси та гарантує приватність медичної інформації. Саме тому розробка такої системи є актуальним і своєчасним завданням.

Метою кваліфікаційної роботи є проєктування та розробка сучасної телемедичної веб-системи для забезпечення віддалених відеоконсультацій, безпечного обміну медичними даними та управління розкладом прийомів.

Для досягнення поставленої мети було вирішено наступні задачі:

– Проаналізовано існуючі рішення та визначено вимоги до розроблюваної телемедичної платформи.

– Спроєктовано клієнт-серверну архітектуру та реляційну модель бази даних з підтримкою рольової моделі (пацієнт, лікар, адміністратор);

– Реалізовано модуль відеоконференцзв'язку на базі протоколу WebRTC та технології Socket.io.

– Розроблено алгоритм децентралізованої P2P-передачі медичних файлів під час консультації.

– Реалізовано модулі управління медичними висновками, рейтингами лікарів та адміністративну панель керування системою.

– Проведено функціональне тестування програмного забезпечення.

Об’єкт дослідження – процес надання та отримання віддалених медичних консультацій.

Предмет дослідження – методи, технології та інструментальні засоби React, Node.js, WebRTC, Socket.io для побудови телемедичних інформаційних систем.

Практичне значення одержаних результатів. Розроблена програмна система є повністю готовим до впровадження продуктом. Запропонована архітектура з використанням P2P-передачі файлів дозволяє медичним установам мінімізувати витрати на серверне обладнання, а вбудований функціонал "розумного" календаря, відгуків та електронних рецептів автоматизує рутинні процеси як для лікарів, так і для пацієнтів.

Наукова новизна. Вперше реалізовано підхід до побудови телемедичної платформи з використанням децентралізованої технології WebRTC DataChannel для P2P-передачі файлів, що дозволяє обмінюватися даними безпосередньо між клієнтами без збереження на проміжних серверах.

Ключові ідеї, архітектурні рішення та практичні здобутки, отримані під час виконання кваліфікаційної роботи, пройшли успішну апробацію. Їх було представлено у форматі тез доповідей та опубліковано у збірнику матеріалів IX Міжнародної студентської науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання» (24–25 квітня 2026 року, м. Тернопіль).

1 АНАЛІЗ ВИМОГ ДО ВЕБЗАСТОСУНКУ

У цьому розділі проводиться аналіз предметної області телемедицини та існуючих інформаційних систем, що надають послуги медичних онлайн-консультацій. Визначаються основні недоліки централізованих архітектур передачі даних та обґрунтовується доцільність використання технології WebRTC для їх усунення. Також формулюються основні вимоги до розроблюваного вебзастосунку, визначаються цілі проєкту та ключові варіанти використання системи різними групами користувачів.

1.1 Аналіз предметної області

Сфера охорони здоров'я на сучасному етапі переживає період активної цифровізації. Телемедицина стала одним із інструментів надання медичних послуг, що дозволяє пацієнтам отримувати кваліфіковану допомогу дистанційно. Це особливо актуально для пацієнтів з обмеженою мобільністю, жителів віддалених регіонів та в умовах підвищеного епідеміологічного ризику [1, 2].

На ринку інформаційних технологій вже існують різноманітні медичні системи та портали, які дозволяють вести електронні медичні картки, записуватися на прийом та зберігати історію хвороби. Проте більшість із цих рішень фокусуються саме на адміністративній частині та документообігу, залишаючи процес самої відеоконсультації стороннім сервісам, таким як Skype, Zoom або Google Meet [3].

Використання сторонніх рішень для відеозв'язку та обміну медичними файлами має ряд суттєвих недоліків. По-перше, виникають проблеми з конфіденційністю персональних даних. Медичні документи, результати аналізів та висновки містять чутливу інформацію. Передача таких файлів через хмарні сервери загального призначення створює ризики витоку інформації та порушує принципи лікарської таємниці.

По-друге, розробка власної системи відеозв'язку з використанням класичної клієнт-серверної архітектури вимагає значних фінансових витрат на утримання серверного обладнання. Проміжні сервери повинні обробляти та маршрутизувати гігабайти відеотрафіку та медіафайлів, що часто призводить до затримок з'єднання та збільшення вартості підтримки системи.

Для вирішення цих проблем у розроблюваному вебзастосунку пропонується використати технологію WebRTC. Цей стандарт дозволяє встановлювати пряме однорангове з'єднання між браузерами користувачів. Відеопотік та медичні файли (наприклад, знімки МРТ чи результати аналізів) передаються безпосередньо від лікаря до пацієнта і навпаки, оминаючи сервери зберігання [4, 5, 6].

Серверна частина застосунку використовується лише на етапі встановлення з'єднання та для збереження текстових медичних висновків і розкладу прийомів у базі даних. Такий децентралізований підхід гарантує максимальну конфіденційність переданих медичних даних, мінімізує затримки під час трансляції та суттєво знижує навантаження на серверну інфраструктуру [7].

1.2 Постановка завдання та критерії успішності

Головним завданням кваліфікаційної роботи є проєктування та програмна реалізація повноцінного вебзастосунку для надання медичних онлайн-консультацій, який об'єднує в собі функціонал управління розкладом, проведення відеоконференцій у реальному часі та безпечного обміну медичними даними.

Для досягнення поставленої мети необхідно вирішити наступні технічні завдання:

– Розробити клієнт-серверну архітектуру системи на базі сучасних вебтехнологій.

– Спроєктувати реляційну базу даних для зберігання облікових записів, розкладу прийомів, медичних висновків та відгуків.

– Реалізувати рольову модель доступу до системи з розділенням прав для пацієнтів, лікарів та адміністраторів.

– Розробити модуль «розумного» календаря, який автоматично фільтрує доступні слоти часу та переносить завершені консультації до архіву.

– Інтегрувати технологію WebRTC для забезпечення стабільного відео- та аудіозв'язку.

– Реалізувати децентралізований механізм передачі медичних файлів безпосередньо між користувачами для зниження навантаження на сервер.

– Розробити адміністративну панель для модерації системи та перегляду статистики.

Критеріями успішності розробленої програмної системи є:

– Безперебійна робота відеозв'язку між двома клієнтами з мінімальною затримкою.

– Успішна пряма передача файлів обсягом до 2 мегабайт без збереження на проміжному сервері.

– Коректне функціонування алгоритму бронювання, що унеможливлює запис двох різних пацієнтів на один і той самий час до одного лікаря.

– Адаптивність інтерфейсу користувача, що забезпечує коректне відображення всіх елементів системи на мобільних пристроях.

1.3 Визначення акторів та варіантів використання

Для формалізації функціональних вимог до розроблюваної телемедичної системи використано метод аналізу прецедентів (варіантів використання). У системі виділено три основні групи користувачів (акторів), кожна з яких має власний набір прав та можливостей: Пацієнт, Лікар та Адміністратор.

Актор Пацієнт є основним споживачем медичних послуг у системі. Його головна мета полягає у швидкому пошуку потрібного спеціаліста та отриманні кваліфікованої консультації.

До варіантів використання для цього актора належать:

– Реєстрація та авторизація в системі із зазначенням персональних даних (вік, стать)

– Перегляд списку доступних лікарів та їхніх рейтингів.

– Вибір вільного часу та бронювання онлайн-консультації.

– Підключення до кімнати відеозв'язку у запланований час.

– Обмін текстовими повідомленнями та медичними файлами під час дзвінка.

– Перегляд архіву минулих консультацій та залишених лікарем медичних висновків (рецептів).

– Оцінювання лікаря та написання відгуку після завершення прийому.

– Редагування власного профілю.

Актор Лікар є надавачем медичних послуг, який використовує платформу для комунікації з пацієнтами та ведення історії хворих. Цей користувач має наступні варіанти використання:

– Авторизація в системі.

– Перегляд власного розкладу та списку запланованих консультацій з відображенням віку та статі пацієнта.

– Підключення до кімнати відеозв'язку.

– Обмін повідомленнями та файлами з пацієнтом.

– Формування та збереження медичного висновку за результатами консультації.

– Перегляд відгуків, залишених пацієнтами про його роботу.

Актор Адміністратор є системним користувачем, який не бере безпосередньої участі в медичних консультаціях, але забезпечує контроль якості та працездатність платформи.

Варіанти використання для адміністратора включають:

– Вхід до захищеної панелі керування.

– Перегляд загальної статистики системи (кількість зареєстрованих пацієнтів, лікарів, проведених прийомів та залишених відгуків).

– Модерація платформи, що включає можливість видалення некоректних або спам-відгуків.

Для візуалізації взаємодії визначених акторів із системою будується діаграма варіантів використання мовою UML що наведена на рисунку 1.1.

Рисунок 1.1 — Діаграма варіантів використання телемедичної системи

Наведена діаграма наочно ілюструє повний спектр функціональних можливостей розробленої системи та чіткий розподіл відповідальності між її користувачами.

1.4 Опис ключових варіантів використання

Для більш детального розуміння функціонування системи наведемо специфікації найважливіших прецедентів: бронювання консультації та проведення відеодзвінка з обміном медичними файлами.

Специфікація варіанта використання «Бронювання часу прийому»

Актор: Пацієнт.

Ціль: успішно запланувати відеоконсультацію з обраним лікарем на вільний слот часу.

Передумови: пацієнт авторизований у системі; обраний лікар має вільні слоти у своєму розкладі.

Основний потік подій:

1. Пацієнт переходить до розділу списку фахівців на головній сторінці кабінету.
2. Пацієнт обирає потрібного спеціаліста та натискає кнопку обрання часу.
3. Система відображає календар із доступними датами (починаючи від поточної).
4. Пацієнт обирає дату, після чого система автоматично фільтрує слоти та відображає лише вільні проміжки часу, виключаючи ті, що вже минули або заброньовані іншими користувачами.
5. Пацієнт обирає доступний часовий слот.
6. Система генерує унікальний ідентифікатор кімнати для відеозв'язку та створює новий запис у базі даних.

7. Система виводить повідомлення про успішне бронювання та оновлює список активних прийомів у кабінетах пацієнта та лікаря. Постумови: консультація успішно додана до розкладу обох користувачів, обраний слот часу стає недоступним для інших пацієнтів.

Специфікація варіанта використання «Участь у відеоконсультації та обмін файлами»

Актори: Пацієнт, Лікар.

Ціль: провести дистанційний прийом за допомогою відеозв'язку з можливістю P2P-передачі медичних документів (знімків, аналізів).

Передумови: обидва користувачі авторизовані в системі та надали браузеру дозволи на використання камери й мікрофона.

Основний потік подій:

1. Лікар та пацієнт натискають кнопку входу до кімнати у списку своїх активних прийомів.

2. Система перенаправляє користувачів на захищену сторінку відеокімнати за унікальним ідентифікатором.

3. За допомогою протоколу WebRTC та проміжного сигнального сервера встановлюється пряме однорангове з'єднання між клієнтами.

4. Користувачі спілкуються за допомогою відео- та аудіозв'язку, маючи можливість керувати своїми медіапристроями через плаваючу панель.

5. За необхідності пацієнт обирає медичний файл на своєму пристрої для відправки лікарю.

6. Система розбиває обраний файл на дрібні фрагменти та передає їх напряму лікарю через захищений канал даних протоколу WebRTC, минаючи центральний сервер.

7. Лікар отримує файл, бачить його прев'ю у вікні чату та завантажує на свій пристрій для аналізу.

8. Після завершення прийому користувачі натискають кнопку завершення дзвінка і повертаються до головного дашборду. Постумови: сеанс зв'язку успішно

закрито, однорангове з'єднання розірвано. Система переміщує прийом до архіву, а лікар отримує можливість сформувати медичний висновок.

1.5 Специфікація вимог до вебзастосунку

Для повноцінного проєктування архітектури та забезпечення якості кінцевого програмного продукту необхідно чітко формалізувати вимоги до телемедичної вебсистеми. Відповідно до сучасних стандартів програмної інженерії, вимоги поділяються на функціональні (що система повинна робити) та нефункціональні (з якими якісними характеристиками вона має це робити) [8].

1.5.1 Функціональні вимоги

Функціональні вимоги описують поведінку системи, її реакцію на вхідні дані та взаємодію з користувачами. З огляду на визначені раніше варіанти використання, розроблювана платформа повинна задовольняти наступні вимоги:

1. Модуль автентифікації та управління профілями:

– Система повинна надавати можливість реєстрації нових користувачів;

– Система повинна вимагати додаткові дані при реєстрації лікаря (спеціалізація) та пацієнта (стать, дата народження для автоматичного розрахунку віку);

– Система повинна забезпечувати механізм безпечної авторизації за унікальною електронною поштою та паролем;

2. Модуль управління розкладом та бронюванням:

– Система повинна відображати пацієнтам список доступних медичних фахівців із зазначенням їхньої спеціалізації та поточного рейтингу;

– Система повинна генерувати інтерактивний календар для вибору вільного часу консультації;

– Система повинна виконувати динамічну перевірку та автоматично блокувати часові слоти, які вже заброньовані іншими користувачами, для повного уникнення колізій;

– Система повинна автоматично переносити консультації, час яких минув, до розділу архіву, приховуючи можливість їх скасування;

3. Модуль відеоконференцзв'язку та обміну даними:

– Система повинна генерувати унікальне посилання (ідентифікатор кімнати) для кожного запланованого прийому;

– Система повинна запитувати та отримувати доступ до камери і мікрофона користувача перед початком дзвінка;

– Система повинна встановлювати P2P-з'єднання за допомогою сигнального сервера для трансляції відео та аудіо в реальному часі;

– Система повинна надавати інтегрований текстовий чат для обміну повідомленнями під час сеансу зв'язку;

– Система повинна забезпечувати механізм децентралізованої передачі медичних файлів безпосередньо між учасниками сеансу, оминаючи центральний сервер бази даних.

4. Модуль зворотного зв'язку та модерації:

– Система повинна дозволяти лікарю формувати та зберігати текстовий медичний висновок (електронний рецепт) після завершення дзвінка;

– Система повинна дозволяти пацієнту залишати оцінку від 1 до 5 та текстовий відгук про лікаря;

– Система повинна динамічно перераховувати середній рейтинг лікаря на основі доданих відгуків;

– Адміністративна панель повинна надавати інструменти для модерації платформи (видалення некоректних відгуків) та перегляду глобальної статистики користувачів;

1.5.2 Нефункціональні вимоги

Нефункціональні вимоги визначають атрибути якості програмного продукту, його продуктивність, безпеку та зручність супроводу.

1. Вимоги до безпеки та конфіденційност:

– Усі паролі користувачів повинні обов'язково хешуватися перед збереженням у базі даних (наприклад, із використанням алгоритму bcrypt), щоб унеможливити їх компрометацію;

– Обмін даними між клієнтом та сервером повинен здійснюватися виключно за захищеним протоколом HTTPS;

– Мультимедійний трафік та медичні файли, що передаються через протокол WebRTC, повинні автоматично шифруватися за стандартом DTLS/SRTP;

– Система не повинна зберігати жодних медичних документів на проміжних серверах задля дотримання лікарської таємниці;

2. Вимоги до продуктивності:

– Серверна частина повинна бути спроєктована таким чином, щоб витримувати одночасне підключення до 100-150 користувачів без критичної деградації часу обробки HTTP-запитів;

– Затримка при передачі відеосигналу має бути мінімальною для забезпечення комфортного спілкування, що досягається завдяки прямому P2P-маршруту;

3. Вимоги до надійності та стійкості:

– Система повинна коректно обробляти виключення та помилки мережі (наприклад, втрату з'єднання зі сторони одного з користувачів) із виведенням відповідних зрозумілих повідомлень у графічному інтерфейсі;

– У разі вичерпання пулу підключень до бази даних під час пікового навантаження, сервер повинен відхиляти нові підключення, не допускаючи повного падіння процесу Node.js;

4. Вимоги до ергономіки та зручності:

– Клієнтська частина має бути реалізована як односторінковий застосунок для забезпечення плавного та миттєвого перемикання між вкладками архіву і розкладу без повного перезавантаження сторінки;

– Графічний інтерфейс повинен відповідати принципам адаптивного дизайну, гарантуючи комфортне використання платформи на мобільних телефонах, планшетах і десктопних моніторах;

5. Вимоги до портативності:

– Клієнтський застосунок повинен стабільно працювати у всіх сучасних веббраузерах Google Chrome, Mozilla Firefox, Microsoft Edge, які підтримують стандарт HTML5 та API медіапристроїв;

– Серверний код має бути кросплатформним, що дозволить безболісно розгортати його на будь-яких хмарних Linux або Windows серверах із встановленим середовищем виконання Node.js;

1.6 Висновки до розділу 1

Проведений аналіз предметної області показав, що впровадження телемедичних інформаційних систем є критично важливим етапом розвитку сучасної сфери охорони здоров'я. Визначено основні недоліки існуючих централізованих рішень, зокрема високі ризики порушення конфіденційності та перевантаження серверної інфраструктури. Обґрунтовано доцільність використання децентралізованої технології WebRTC для створення швидкого та безпечного каналу зв'язку.

У розділі сформовано цілі та завдання дипломного проєкту, а також визначено критерії його успішності. Проведено системний аналіз вимог за допомогою методології UML: виділено ключових акторів, побудовано діаграму варіантів використання та детально описано базові прецеденти роботи системи.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ

У цьому розділі розглядається процес інженерного проєктування розроблюваної телемедичної вебсистеми. Описується загальна архітектура програмного рішення, визначаються основні компоненти та характер їхньої взаємодії. Здійснюється обґрунтування вибору технологічного стека для реалізації клієнтської та серверної частин. Важлива увага приділяється проєктуванню структури реляційної бази даних, а також об'єктно-орієнтованому моделюванню системи з використанням діаграм класів та діаграм послідовності для візуалізації складних процесів обміну медичними даними.

2.1 Вибір процесу розробки

Для створення телемедичної вебсистеми було обрано ітеративну модель розробки програмного забезпечення з використанням принципів гнучкої методології Agile. Вибір цього підходу зумовлений специфікою сучасних вебтехнологій та необхідністю раннього тестування складних компонентів, зокрема модуля відеозв'язку на базі WebRTC. На відміну від класичної каскадної моделі, ітеративний підхід дозволяє розбивати процес створення системи на невеликі функціональні етапи, кожен з яких завершується створенням робочого фрагмента програми.

Життєвий цикл розробки даного вебзастосунку було розділено на кілька послідовних фаз. На першій фазі відбувався збір та аналіз вимог, проєктування структури бази даних і визначення базової клієнт-серверної архітектури. Друга фаза включала реалізацію серверної частини для обробки запитів, управління розкладом та створення системи авторизації з рольовою моделлю.

На третій фазі розроблявся клієнтський графічний інтерфейс для різних груп користувачів та відбувалася інтеграція модуля сигналізації на базі вебсокетів. Четверта фаза була найскладнішою і повністю присвячувалася налаштуванню прямого однорангового з'єднання, передачі відеопотоку та реалізації механізму

децентралізованого обміну файлами. Остання фаза передбачала комплексне функціональне тестування, виправлення виявлених недоліків та розгортання готового продукту на хостинговому сервері. Такий поетапний процес дозволив ефективно управляти часом, мінімізувати архітектурні ризики та забезпечити високу якість кінцевого програмного продукту.

2.2 Проєктування архітектури вебзастосунку

Для реалізації телемедичної системи обрано багаторівневу клієнт-серверну архітектуру. Такий підхід забезпечує чіткий логічний поділ між інтерфейсом користувача, бізнес-логікою та рівнем зберігання даних, що суттєво підвищує масштабованість системи та спрощує процес її подальшої підтримки [9].

Глобально архітектуру розроблюваного застосунку можна розділити на три основні вузли: клієнтський вебзастосунок, сервер додатків та сервер бази даних. Клієнтська частина відповідає за відображення графічного інтерфейсу, валідацію введених користувачем даних та управління медіапристроями (камерою і мікрофоном). Сервер додатків обробляє HTTP-запити, реалізує логіку авторизації, управління розкладом та забезпечує маршрутизацію сигнальних повідомлень для встановлення відеозв'язку. Рівень даних відповідає за надійне зберігання постійної інформації про користувачів та історію їхніх консультацій.

Особливістю спроєктованої архітектури є використання гібридної моделі передачі даних. Текстова інформація, інформація про розклад та налаштування профілів передаються за класичною схемою через REST API до центрального сервера. Однак для передачі мультимедійних даних та файлів застосовується децентралізований підхід на основі технології WebRTC.

Для встановлення однорангового Peer-to-Peer з'єднання архітектура системи передбачає наявність сигнального сервера, реалізованого за допомогою технології WebSockets. Сигнальний сервер виконує роль диспетчера, який дозволяє браузеру

пацієнта та браузеру лікаря обмінятися мережевими координатами та параметрами медіапотоків так званими SDP-пропозиціями та ICE-кандидатами.

Після успішного обміну сигнальними даними через центральний сервер, клієнтські застосунки встановлюють пряме мережеве з'єднання один з одним. Починаючи з цього моменту, весь відеопотік, аудіопотік та бінарні дані медичних файлів передаються безпосередньо від пристрою лікаря до пристрою пацієнта і навпаки. Центральний сервер більше не бере участі в маршрутизації цього важкого трафіку, що усуває ризик його перевантаження та забезпечує повну конфіденційність медичних матеріалів, оскільки вони не зберігаються на жодних проміжних вузлах.

Таким чином, розроблена гібридна клієнт-серверна та P2P архітектура дозволяє створити оптимізовану, безпечну та стійку до високих навантажень телемедичну платформу.

Архітектура телемедичної системи наведена на рисунку 2.1.

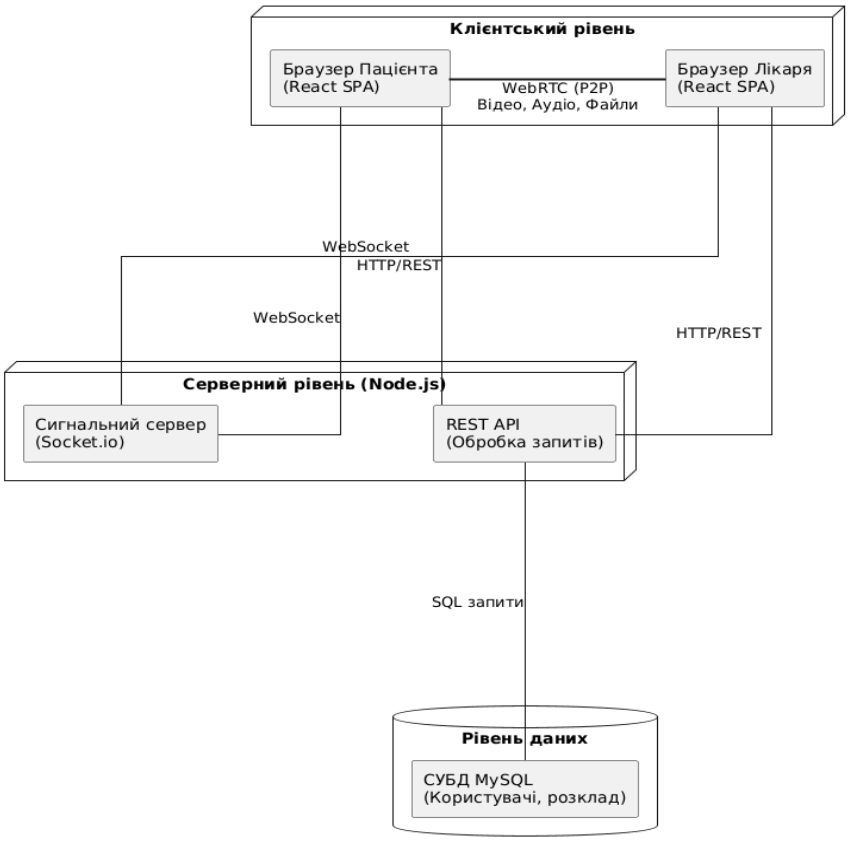

Рисунок 2.1 — Архітектура телемедичної системи

Як видно зі схеми, децентралізований канал WebRTC суттєво розвантажує серверний рівень від обробки мультимедійного трафіку.

2.3 Побудова схем бази даних

Для забезпечення надійного зберігання інформації про користувачів, розклад консультацій та медичні висновки спроєктовано реляційну базу даних під управлінням СУБД MySQL. Структура бази даних розроблена з урахуванням принципів нормалізації, що дозволяє уникнути надмірності інформації та забезпечити високу швидкість виконання запитів [10].

База даних телемедичної системи складається з трьох основних взаємопов'язаних таблиць: таблиці користувачів, таблиці запланованих прийомів та таблиці відгуків.

Таблиця користувачів зберігає облікові записи всіх учасників платформи. До її основних атрибутів входять унікальний первинний ключ, повне ім'я, електронна пошта та хешований пароль. Розділення прав доступу реалізовано через колонку ролі, яка використовує тип перелічення з допустимими значеннями: пацієнт, лікар та адміністратор. Додатково для коректного відображення профілів зберігаються демографічні дані користувачів та професійна спеціалізація для облікових записів медичних працівників.

Таблиця прийомів виконує функцію електронного розкладу та медичної картки. Вона містить зовнішні ключі, які пов'язують конкретний запис із ідентифікатором лікаря та ідентифікатором пацієнта з таблиці користувачів. Також у цій сутності зберігаються запланована дата і час консультації, згенерований унікальний ідентифікатор кімнати для ініціалізації сеансу WebRTC, а також текстове поле для збереження медичного висновку або електронного рецепта, який лікар додає після завершення відеодзвінка.

Таблиця відгуків забезпечує роботу системи оцінювання медичних фахівців. Вона фіксує зв'язок між пацієнтом-автором та лікарем, містить числову оцінку від

одного до п'яти, текст коментаря та автоматичну мітку часу створення запису. На основі цих даних серверна частина динамічно вираховує та оновлює середній рейтинг кожного лікаря.

Логічна модель розробленої бази даних у вигляді діаграми сутність-зв'язок наведена на рисунку 2.2.

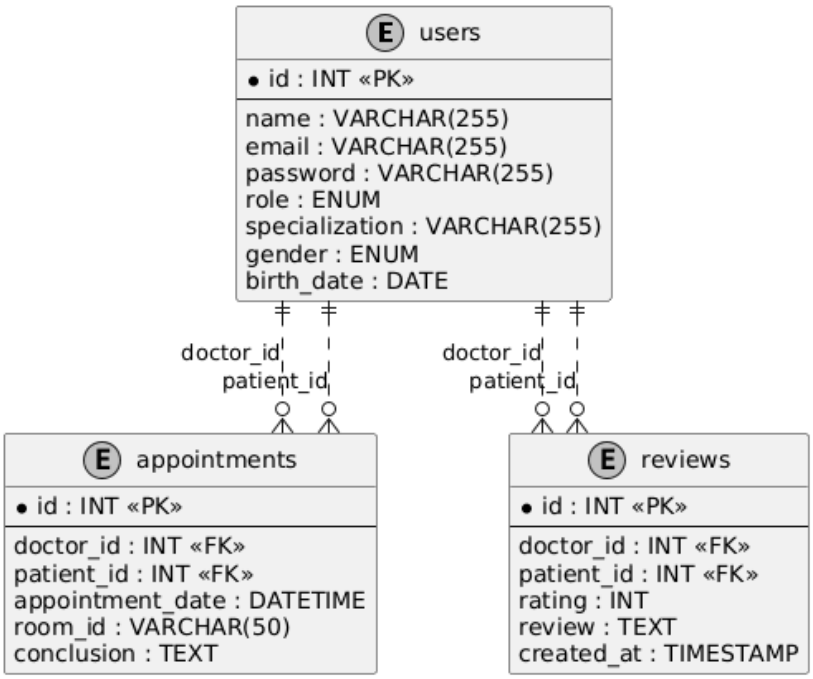

Рисунок 2.2 — ER-діаграма бази даних телемедичної системи

Спроєктована схема бази даних повністю покриває потреби бізнес-логіки розробленого телемедичного вебзастосунку та забезпечує збереження цілісності даних завдяки використанню строгих реляційних зв'язків.

2.4 Побудова UML-діаграм класів

У цьому підрозділі здійснюється об'єктно-орієнтоване проєктування розробленої телемедичної системи. Для візуалізації структури програмного коду, опису основних сутностей, їхніх атрибутів, методів та взаємозв'язків між ними використовується діаграма класів мови моделювання UML. Враховуючи обрану

архітектуру, систему можна логічно розділити на клієнтські компоненти та серверні контролери.

Серверна частина реалізована з використанням принципів патерну MVC в адаптації для REST API. Основними серверними модулями є контролери користувачів, розкладу та відгуків, які містять функції для обробки вхідних мережевих запитів. Вони взаємодіють із рівнем бази даних, виконуючи базові операції створення, читання, оновлення та видалення записів. Окрім цього, виділено окремий клас для управління вебсокетами, який відповідає за маршрутизацію сигнальних повідомлень під час встановлення з'єднання WebRTC між клієнтами.

Клієнтська частина представлена набором компонентів бібліотеки React, які відповідають за відображення інтерфейсу та управління локальним станом застосунку. Головним компонентом-контейнером є панель управління, яка завантажує розклад та керує перемиканням між вкладками архіву і активних прийомів. Ключовим модулем відеозв'язку є компонент відеокімнати, що містить методи для захоплення медіапотоку з камери та мікрофона, ініціалізації об'єкта однорангового з'єднання та управління каналом даних для децентралізованого обміну файлами.

Логічна структура основних компонентів системи та зв'язки між ними наведені на рисунку 2.3.

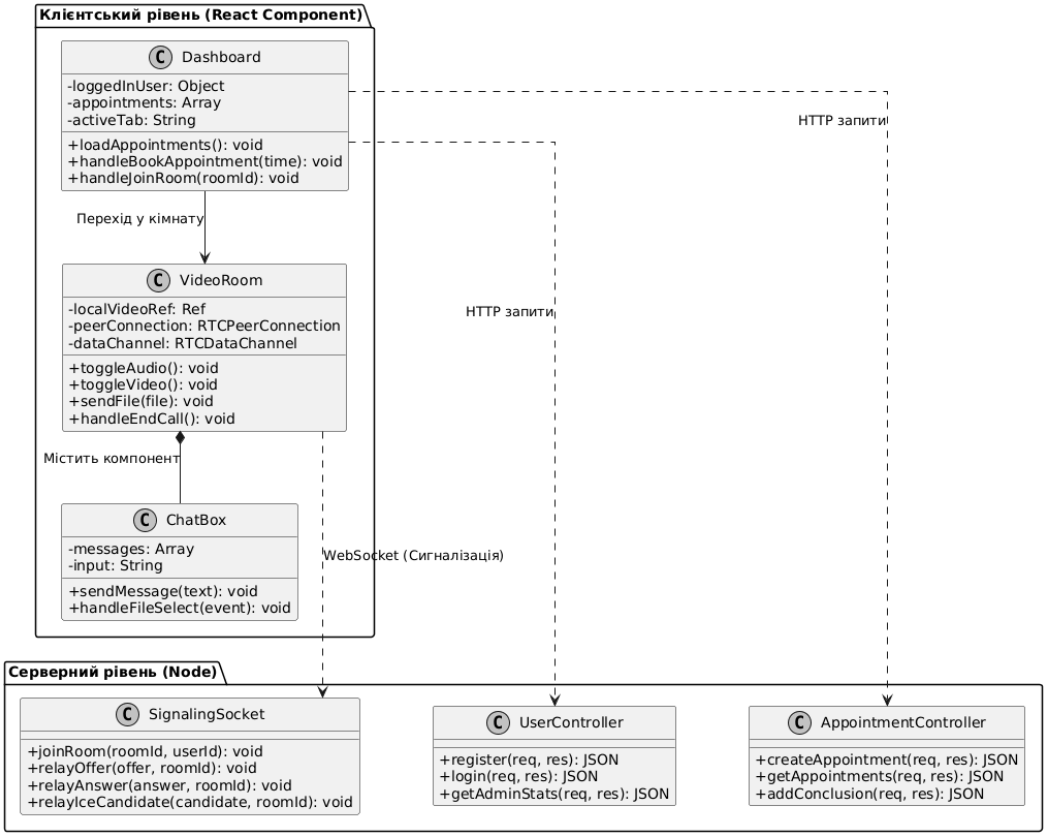

Рисунок 2.3 — UML-діаграма класів телемедичної системи

Окрім статичної структури класів, важливим аспектом об'єктно-орієнтованого проєктування є моделювання динамічної взаємодії об'єктів у часі. Найбільш складним архітектурним процесом у розробленій системі є ініціалізація однорангового з'єднання між клієнтськими компонентами відеокімнати за допомогою сигнального сервера. Для візуалізації цього процесу побудовано UML-діаграму послідовності. Діаграма послідовності встановлення WebRTC з'єднання наведена на рисунку 2.4.

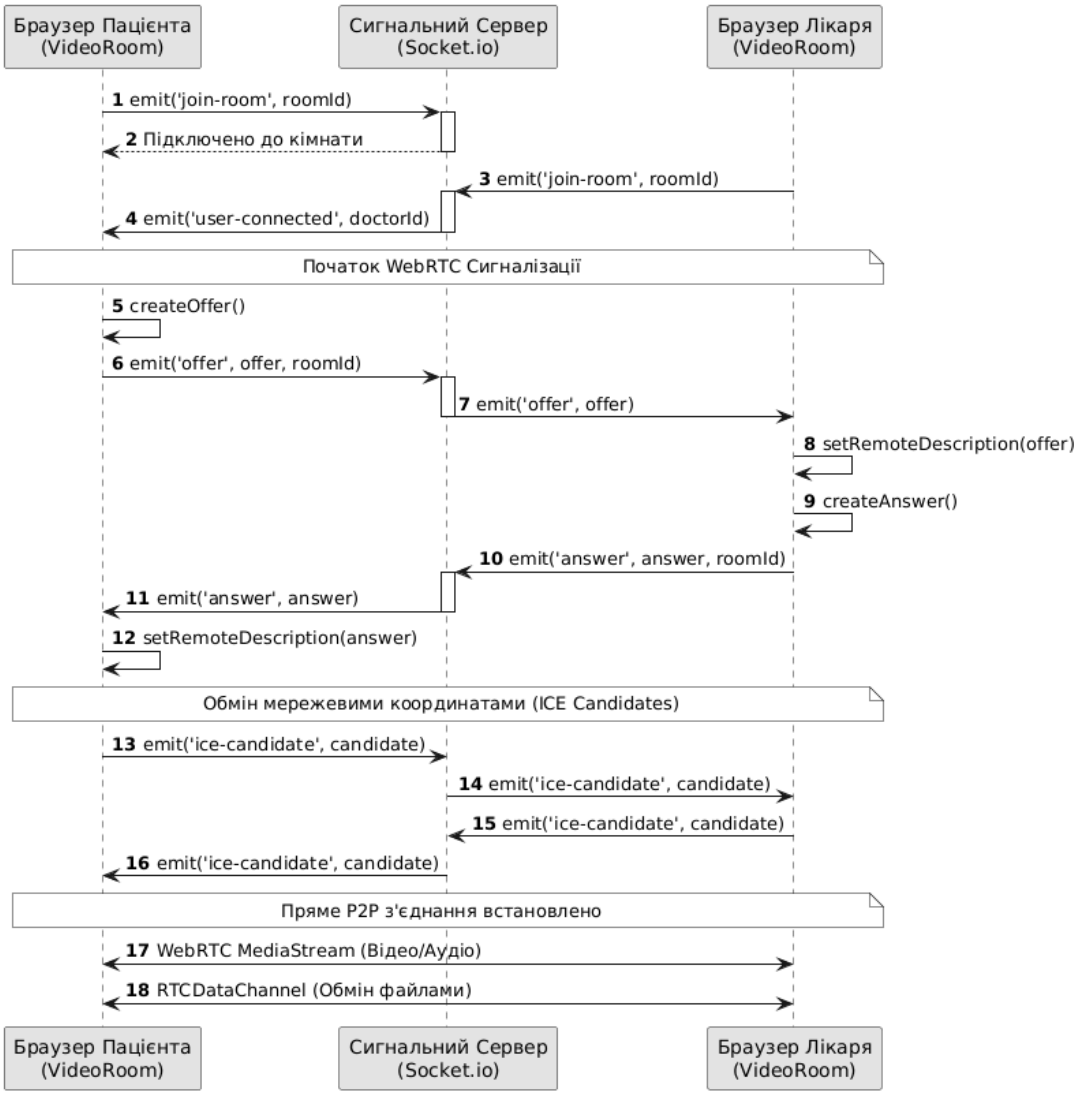

Рисунок 2.4 — UML-діаграма послідовності встановлення WebRTC з'єднання

Діаграма відображає хронологію обміну повідомленнями між браузером пацієнта, сервером на базі Socket.io та браузером лікаря. Процес починається з підключення користувачів до спільної віртуальної кімнати. Далі відбувається асинхронний обмін конфігураційними даними: ініціатор генерує SDP-пропозицію, яка через сигнальний сервер передається іншому учаснику, котрий у свою чергу формує SDP-відповідь. Після паралельного обміну ICE-кандидатами для визначення оптимального мережевого маршруту, сигнальний сервер припиняє участь у маршрутизації, і клієнти встановлюють прямий P2P-канал для обміну медіаданими.

2.5 Вибір мови та середовища розробки

Успішна реалізація архітектурних рішень багато в чому залежить від правильного вибору інструментальних засобів та технологічного стеку. Для розробки телемедичної вебсистеми було обрано екосистему JavaScript, що дозволило використовувати єдину мову програмування як для клієнтської, так і для серверної частин застосунку. Такий підхід, відомий як повностикова JavaScript розробка, суттєво зменшує час на перемикання контексту та спрощує обмін даними між компонентами [11].

Для створення клієнтського інтерфейсу було обрано бібліотеку React. Її компонентний підхід дозволяє створювати незалежні елементи інтерфейсу, які легко масштабувати та повторно використовувати в різних частинах платформи. Використання віртуального об'єктного дерева документів Virtual DOM у React забезпечує високу продуктивність та плавність роботи інтерфейсу, що є критично важливим для медичних систем реального часу [12, 13].

Серверна частина реалізована на базі платформи Node.js із використанням фреймворку Express. Висока продуктивність Node.js завдяки асинхронній моделі вводу-виводу ідеально підходить для розробки систем, які потребують обробки великої кількості одночасних з'єднань. Для реалізації сигнального сервера було застосовано пакет Socket.io, який абстрагує роботу з протоколом WebSockets та забезпечує надійний двосторонній зв'язок між сервером і клієнтами [14, 15, 16].

Для розробки та налагодження програмного коду використовувалося інтегроване середовище розробки Visual Studio Code. Цей редактор має вбудовану підтримку сучасних стандартів мови, термінал для виконання серверних скриптів та широку екосистему розширень для перевірки якості коду. Управління залежностями та підключення зовнішніх модулів здійснювалося за допомогою стандартного пакетного менеджера npm.

Для розгортання та хостингу розробленого вебзастосунку було обрано хмарну платформу Alwaysdata. Цей сервіс надає комплексну інфраструктуру як

послугу (PaaS), що включає вбудовану підтримку середовища виконання Node.js та систему управління базами даних MySQL. Використання Alwaysdata дозволило швидко налаштувати робоче серверне оточення без необхідності глибокого адміністрування операційної системи, а також забезпечило безперебійний публічний доступ до телемедичної платформи в мережі Інтернет.

Для управління файлами на віддаленому сервері та завантаження оновлень програмного коду застосовувався протокол передачі файлів FTP. Як клієнтський інструмент було використано програму FileZilla, яка є кросплатформним програмним забезпеченням з відкритим вихідним кодом. Завдяки зручному графічному інтерфейсу та підтримці безпечного з'єднання, цей інструмент забезпечив швидку та надійну синхронізацію локальної збірки застосунку з робочим сервером.

2.6 Реалізація основних класів та методів

У цьому підрозділі детально розглядається програмна реалізація розробленої телемедичної платформи. Відповідно до обраної клієнт-серверної архітектури, програмний код логічно розділено на серверні модулі обробки даних (контролери) та клієнтські компоненти відображення інтерфейсу користувача. Для написання коду використовувався стандарт ECMAScript 6 (ES6) та вище, що дозволило застосувати сучасні конструкції мови, такі як асинхронні функції, деструктуризацію об'єктів та стрілочні функції.

Реалізація серверної частини базується на фреймворку Express. Логіка обробки HTTP-запитів інкапсульована у відповідних контролерах. Контролер управління користувачами userController містить набір методів для забезпечення авторизації, реєстрації та роботи з профілями. Метод реєстрації користувача реалізує механізм додавання нового запису в базу даних із зазначенням ролі, особистих даних та медичної спеціалізації для лікарів. Для забезпечення безпеки системи реалізовано метод входу, який звіряє облікові дані та повертає об'єкт

користувача у разі успішної авторизації. Окрім базових функцій, цей контролер містить методи для роботи з відгуками: додавання нового відгуку, отримання списку відгуків для конкретного лікаря з автоматичним обчисленням середнього рейтингу на стороні системи управління базами даних, а також метод видалення відгуку. Для потреб адміністратора в цьому ж контролері реалізовано метод збору глобальної статистики, який виконує комплексні агрегаційні SQL-запити для підрахунку загальної кількості пацієнтів, лікарів, прийомів та відгуків у системі.

Контролер управління розкладом appointmentController відповідає за життєвий цикл медичної консультації. Найбільш критичним з точки зору бізнес-логіки є метод створення нового запису на прийом. Перед додаванням інформації в базу даних цей метод виконує перевірку наявності колізій: він робить попередній запит до таблиці прийомів, щоб переконатися, що обраний пацієнтом час у конкретного лікаря ще не зайнятий. Програмна реалізація цього алгоритму перевірки наведена в лістингу 2.1.

Лістинг 2.1 – Метод створення прийому з перевіркою колізій часу

```
exports.createAppointment = async (req, res) => {
  const {doctor_id, patient_id, appointment_date, room_id} =req.body;
    try {
        const [existing] = await db.query(
            'SELECT id FROM appointments WHERE doctor_id = ? AND appointment_date = ?',
            [doctor_id, appointment_date]
        );
        if (existing.length > 0) {
            return res.status(400).json({ message: 'Цей час у лікаря вже зайнятий!' });
        }
        const [result] = await db.query(
            'INSERT INTO appointments (doctor_id, patient_id, appointment_date, room_id) VALUES (?, ?, ?, ?)',
```

```
                [doctor_id, patient_id, appointment_date, room_id]
            );
            res.status(201).json({ message: 'Консультацію успішно заплановано!', id: result.insertId });
        } catch (error) {
            res.status(500).json({ message: 'Помилка при створенні запису' });
        }
    };
```

Якщо слот вільний, метод генерує унікальний ідентифікатор кімнати для майбутньої відеоконференції і зберігає запис. Метод отримання розкладу містить розгалужену логіку: залежно від ролі користувача лікар чи пацієнт, він формує SQL-запит таким чином, щоб повернути не лише час прийому, але й дані протилежної сторони ім'я, вік, стать пацієнта для лікаря, або ім'я та спеціалізацію лікаря для пацієнта. Також реалізовано метод оновлення медичного висновку, який дозволяє лікарю прикріпити текстовий рецепт до запису прийому після завершення дзвінка.

Модуль сигналізації signaling реалізовано з використанням бібліотеки Socket.io. Цей файл не працює з базою даних, а виконує виключно роль диспетчера повідомлень реального часу. У ньому реалізовано прослуховування подій підключення до віртуальної кімнати. Коли клієнт ініціює створення P2P-з'єднання, сигнальний сервер отримує події обміну конфігураціями сеансу (SDP offer/answer) та ICE-кандидатами й асинхронно маршрутизує їх між учасниками. Фрагмент коду, що забезпечує трансляцію медіаконфігурацій, наведено в лістингу 2.2.

Лістинг 2.2 – Маршрутизація сигнальних повідомлень WebRTC на сервері

```
socket.on('join-room', (roomId, userId) => {
    socket.join(roomId);
    socket.to(roomId).emit('user-connected', userId);

    socket.on('offer', (offer, roomId) => {
```

```
            socket.to(roomId).emit('offer', offer);
        });

        socket.on('answer', (answer, roomId) => {
            socket.to(roomId).emit('answer', answer);
        });

        socket.on('ice-candidate', (candidate, roomId) => {
            socket.to(roomId).emit('ice-candidate', candidate);
        });
    });
```

Така логіка гарантує, що сигнальні повідомлення передаються виключно тим користувачам, які знаходяться в тій самій кімнаті, забезпечуючи ізоляцію даних та безпеку між різними медичними консультаціями.

Клієнтська частина реалізована за допомогою бібліотеки React та функціональних компонентів із використанням хуків життєвого циклу. Основним компонентом робочої області є Dashboard. Цей компонент керує загальним станом сеансу користувача. У ньому реалізовано логіку перемикання між активними прийомами та архівом за допомогою фільтрації масиву даних: система автоматично порівнює поточний системний час із часом запланованої консультації та переносить завершені прийоми у стан архіву, блокуючи можливість їх випадкового видалення. Також у цьому компоненті міститься алгоритм генерації доступних часових слотів для запису до лікаря, який відкидає минулий час та вже заброньовані іншими пацієнтами години.

Окремим ізольованим модулем є компонент AdminDashboard. Він відповідає за надання адміністративного доступу до платформи. Його методи дозволяють отримувати масив статистичних даних із бекенду та відображати повний перелік усіх залишених на платформі відгуків з можливістю їх глобального видалення в один клік задля модерації спаму або некоректної поведінки.

Найбільш технологічно складним класом у системі є компонент VideoRoom, який реалізує відеозв'язок та передачу медичних файлів за протоколом WebRTC. У методі ініціалізації компонента відбувається звернення до браузерного інтерфейсу для захоплення медіапотоку з веб-камери та мікрофона. Далі створюється об'єкт однорангового з'єднання з налаштуванням публічних серверів для проходження через брандмауери NAT. Особливої уваги заслуговує реалізований метод відправки файлів. Для забезпечення передачі важких медичних документів наприклад, сканів аналізів без використання сервера, метод зчитує файл, конвертує його в текстовий рядок і розбиває на малі фрагменти по 16 кілобайт. Ці фрагменти по черзі відправляються через відкритий канал даних WebRTC із мінімальною часовою затримкою для запобігання переповненню буфера. Повний текст програмної реалізації компонента відеокімнати наведено в додатку Б.

Компонент ChatBox виконує функцію візуального відображення повідомлень та інтерфейсу вибору файлів. Він отримує методи відправки повідомлень та файлів від батьківського компонента через механізм властивостей props. У ньому реалізовано метод автоматичного прокручування списку повідомлень до найновішого та логіку відображення прев'ю зображень або кнопок завантаження для документів формату PDF, що отримані від співрозмовника.

Описана реалізація класів, методів та алгоритмів утворює надійний, безпечний та легко масштабований програмний комплекс. Завдяки ізоляції функціоналу по окремих модулях та контролерах, код системи залишається читабельним і повністю готовим до подальшого розширення новим медичним функціоналом.

2.7 Розробка інтерфейсу користувача

Проєктування графічного інтерфейсу користувача UI та користувацького досвіду UX є одним із найважливіших етапів створення медичних інформаційних систем. Враховуючи те, що користувачами платформи є люди різного віку та рівня

цифрової грамотності, головними критеріями при розробці візуальної частини стали інтуїтивна зрозумілість, мінімалізм та доступність. Клієнтський інтерфейс реалізовано у вигляді односторінкового застосунку SPA, що виключає необхідність повного перезавантаження сторінок і робить взаємодію з системою швидкою та плавною.

Візуальний стиль платформи побудовано на основі сучасних принципів вебдизайну. Для фону обрано нейтральний світло-сірий відтінок, який знижує навантаження на очі користувача під час тривалого використання. Основні контентні блоки оформлено у вигляді карток із м'якими заокругленими кутами та розсіяними тінями, що створює ефект глибини інтерфейсу. Для кнопок та інтерактивних елементів використано контрастні кольори: синій для основних дій, зелений для підтверджень та переходу до архіву, червоний для скасування прийомів або завершення виклику.

Першим етапом взаємодії користувача з платформою є інтерфейс авторизації та реєстрації. Форма входу спроєктована у вигляді компактної централізованої картки, яка містить текстові поля для введення електронної пошти та пароля. Для нових учасників системи реалізовано інтерактивний режим реєстрації, який адаптивно підлаштовує свій вміст залежно від обраної ролі. У разі вибору ролі пацієнта, форма вимагає обов'язкового введення демографічних даних, таких як стать та дата народження, що необхідно для автоматичного розрахунку віку в кабінеті лікаря.. Зовнішній вигляд форми авторизації та реєстрації наведено на рисунку 2.5.

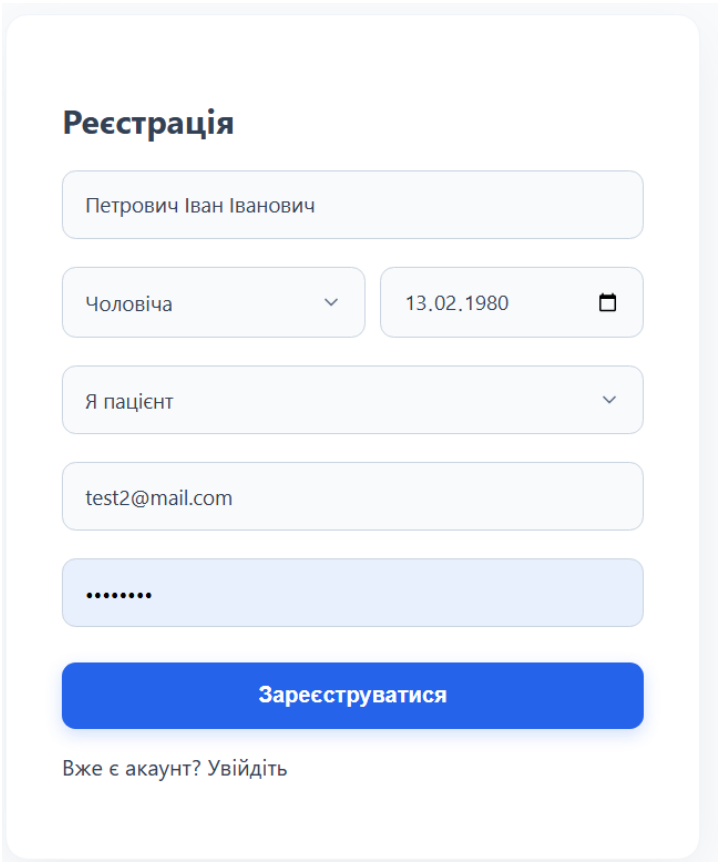

Рисунок 2.5 — Інтерфейс реєстрації користувачів

Головним робочим простором після успішного входу є панель управління Dashboard. Її структура динамічно змінюється залежно від ролі авторизованого користувача. У верхній частині розміщено привітання та бейдж із зазначенням статусу пацієнт або лікар. Центральну частину екрана займає блок розкладу, який реалізовано у вигляді двох вкладок: активні прийоми та архів висновків. Такий розподіл дозволяє користувачам швидко фокусуватися на майбутніх консультаціях, не втрачаючи при цьому доступу до історичних медичних даних. Для пацієнтів додатково реалізовано сітку карток медичних фахівців із відображенням їхньої спеціалізації, рейтингу та інтерактивним календарем для вибору вільного часу. Зовнішній вигляд панелі управління пацієнта наведено на рисунку 2.6.

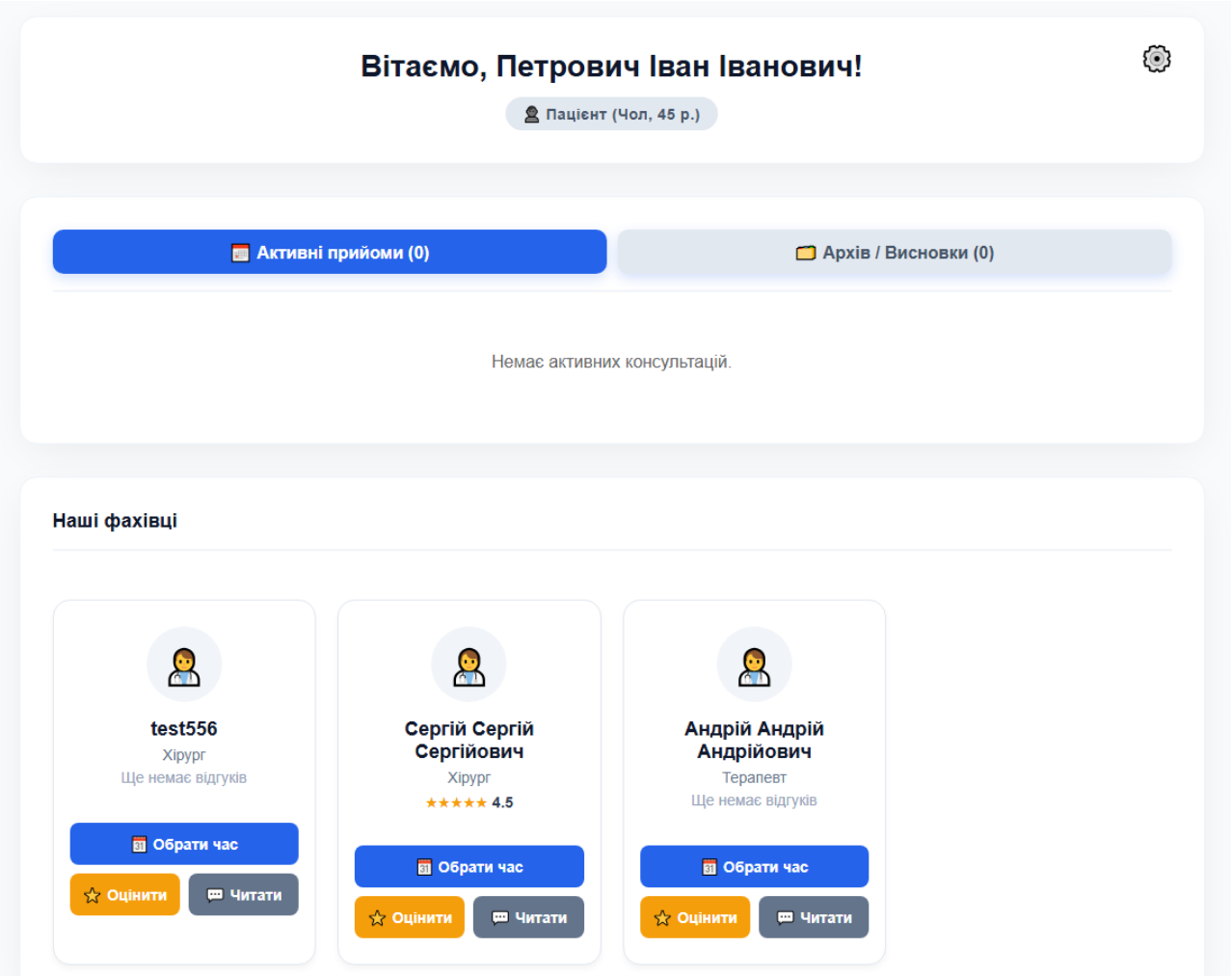

Рисунок 2.6 — Графічний інтерфейс головної панелі користувача

Дизайн модуля відеоконсультації VideoRoom спроєктовано з орієнтацією на максимальну концентрацію уваги на співрозмовнику. Інтерфейс імітує визнані індустріальні стандарти, такі як Google Meet. Екран розділено на дві основні зони: візуальну та комунікаційну. Зона відеозв'язку має темний фон для створення контрасту з відеопотоками. Локальне відео відображається у зменшеному масштабі, тоді як відео лікаря або пацієнта займає основну частину екрана. У нижній частині відеозони розміщено плаваючу панель управління медіапристроями з кнопками увімкнення/вимкнення мікрофона та камери, а також кнопкою завершення виклику.

Комунікаційна зона розташована у правій частині екрана та являє собою інтерактивну панель чату. Вона містить історію текстових повідомлень та елементи

управління для надсилання файлів. Використання спеціальних іконок (наприклад, зображення скріпки) робить процес прикріплення медичних документів очевидним для користувача. Інтерфейс відеокімнати під час проведення консультації наведено на рисунку 2.7.

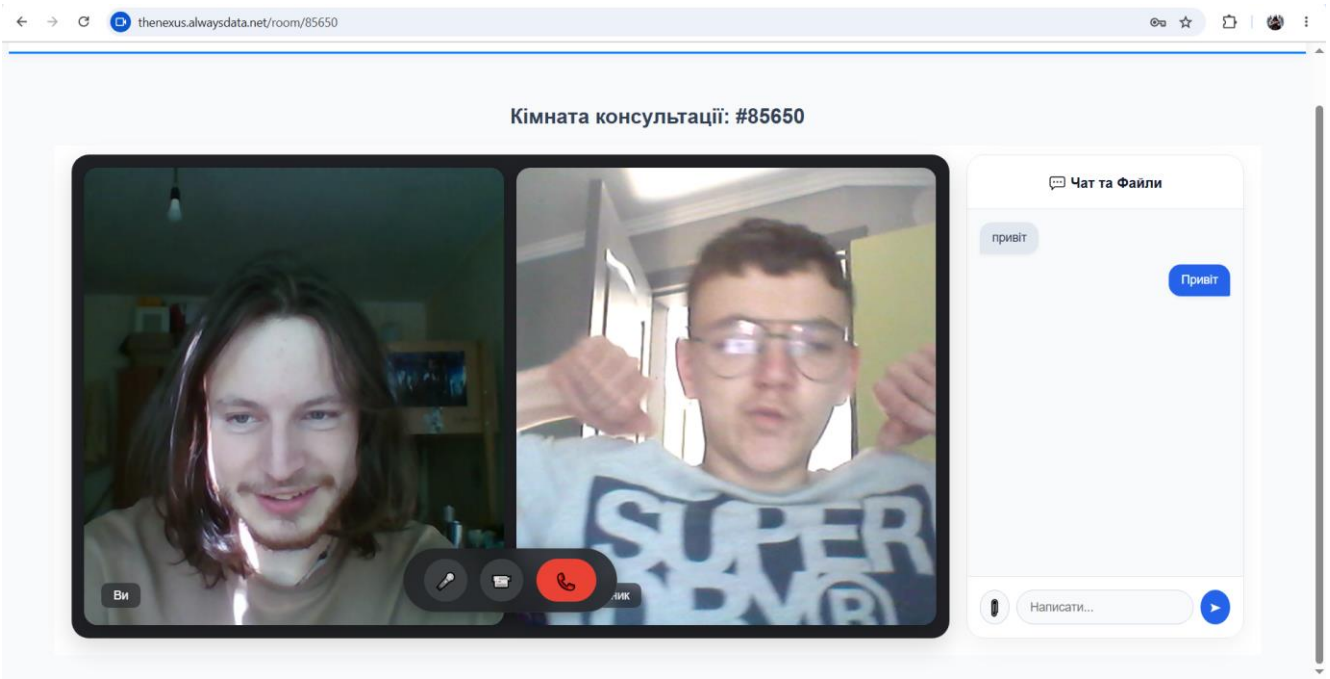

Рисунок 2.7 — Інтерфейс кімнати відеозв'язку та чату

Для внутрішнього контролю та модерації платформи розроблено окремий ізольований інтерфейс адміністративної панелі управління (AdminDashboard). Вхід до цього модуля захищений окремою формою автентифікації. Графічне представлення панелі адміністратора складається з двох ключових областей. Верхня область містить інформаційну матрицю статистичних показників системи, де у вигляді аналітичних карток відображається загальна кількість зареєстрованих пацієнтів, активних лікарів, створених консультацій та доданих відгуків. Нижня частина відведена під інструменти модерації та контролю якості: вона містить повний інтерактивний список усіх залишених на платформі відгуків пацієнтів із відображенням імен авторів, виставлених оцінок, текстових коментарів та спеціальної кнопки для миттєвого видалення некоректних записів чи спаму. Інтерфейс адміністративної панелі відображено на рисунку 2.8.

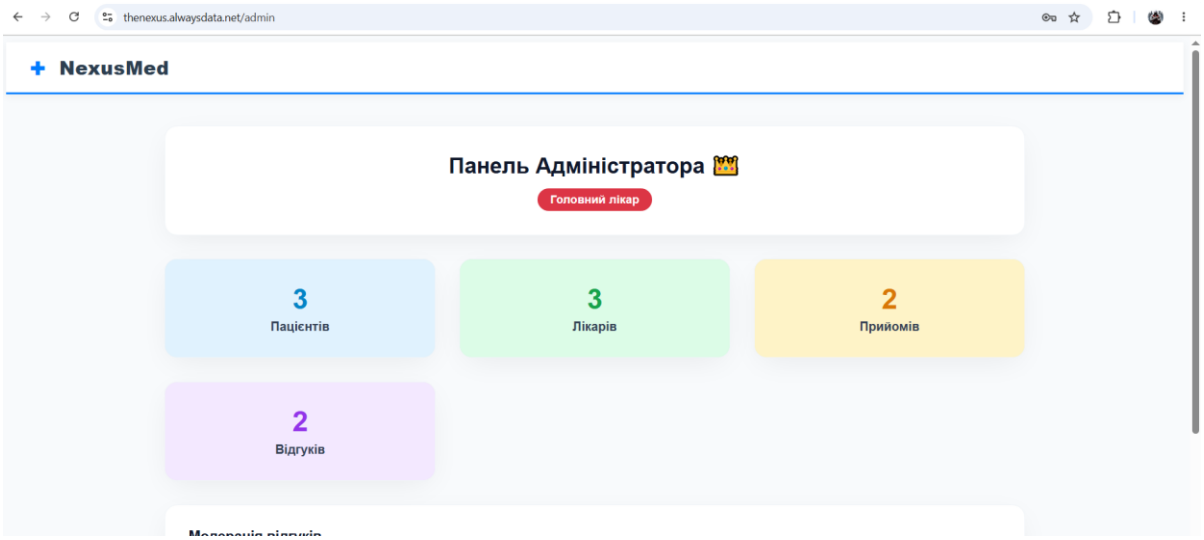

Рисунок 2.8 — Інтерфейс адміністративної панелі управління

Окрему увагу під час верстки інтерфейсу було приділено принципам адаптивного дизайну Responsive Web Design. За допомогою каскадних таблиць стилів CSS та медіазапитів Media Queries забезпечено коректне відображення всіх вищеописаних модулів платформи на пристроях із різною роздільною здатністю екрана. На мобільних пристроях сітка карток автоматично трансформується в одну колонку, блоки статистики адаптуються під ширину екрана, а у відеокімнаті панель чату переміщується під відеопотік, що гарантує комфортне використання телемедичної системи зі смартфонів та планшетів.

2.8 Висновки до розділу 2

У другому розділі виконано комплексне проєктування та реалізацію телемедичної вебсистеми. Обґрунтовано вибір ітеративної методології Agile, що дозволило ефективно розділити розробку на логічні етапи. Спроєктовано багаторівневу клієнт-серверну архітектуру, яка відокремлює обробку бізнес-логіки від візуального представлення, а також інтегрує децентралізований механізм WebRTC для P2P-передачі мультимедійних даних.

Розроблено нормалізовану реляційну базу даних MySQL, яка надійно зберігає інформацію про облікові записи, електронний розклад та медичні висновки. За допомогою UML-діаграм класів та послідовності детально задокументовано об'єктно-орієнтовану структуру системи та хронологію обміну сигнальними повідомленнями. Програмна реалізація, виконана мовою JavaScript з використанням бібліотеки React та платформи Node.js, підтвердила ефективність обраного технологічного стеку. Створений графічний інтерфейс відповідає сучасним вимогам ергономіки та адаптивності, забезпечуючи комфортну взаємодію користувачів із розробленим програмним продуктом.

3 ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЯ ВЕБЗАСТОСУНКУ

Заключний етап розробки будь-якого програмного забезпечення включає обов'язкові процеси оцінки його якості, розгортання в продуктивному середовищі та формування чітких рекомендацій щодо подальшої експлуатації. У цьому розділі розглядаються методи та інструменти практичних випробувань створеної телемедичної вебсистеми за допомогою засобів автоматизації та навантажувального моделювання. Описано покрокову процедуру конфігурування хмарного хостингу Alwaysdata, імпорту реляційної бази даних, а також визначено технічні вимоги для клієнтської та серверної частин інформаційного комплексу. Підсумкова верифікація дозволяє оцінити ступінь готовності програмного продукту до реального впровадження у медичну практику.

3.1 Тестування вебзастосунку

Процес тестування розробленого вебзастосунку є ключовим механізмом забезпечення надійності закладених алгоритмів, який дозволяє виявити приховані дефекти, логічні помилки у вихідному коді та потенційні вразливості системи до моменту її передачі кінцевим користувачам. Специфіка медичних онлайн-платформ висуває підвищені вимоги до безвідмовності та швидкості обробки запитів, оскільки стабільність відеозв'язку та конфіденційність трафіку безпосередньо впливають на якість надання консультаційних послуг. Оцінка працездатності розробленої системи проводилася комплексно, охоплюючи перевірку ізольованої роботи окремих серверних контролерів, стійкості пулу з'єднань СУБД та поведінки графічного інтерфейсу в умовах інтенсивного навантаження.

3.1.1 Види та план тестування

Для забезпечення високої якості програмного продукту було обрано комплексний підхід, який включає наступні види тестування:

1. Функціональне тестування: Перевірка коректності роботи бізнес-логіки системи. Включає тестування модулів реєстрації/авторизації, запису на прийом, системи відгуків.

2. Навантажувальне тестування: Оцінка стабільності роботи вебсервісу при одночасному підключенні великої кількості користувачів.

3. Автоматизоване тестування: Перевірка критичних шляхів взаємодії користувача з платформою шляхом імітації його реальних дій у веббраузері. Цей вид тестування дозволяє гарантувати безпомилкову інтеграцію клієнтської частини з серверним програмним інтерфейсом та базою даних.

План тестування розроблено таким чином, щоб охопити як нефункціональні, так і функціональні вимоги до телемедичної системи. На першому етапі заплановано проведення навантажувального тестування за допомогою інструменту Apache JMeter для визначення пропускної здатності серверної архітектури та пулу з'єднань MySQL. На другому етапі виконується автоматизоване тестування ключових користувацьких сценаріїв, зокрема процесу автентифікації та відмальовки панелі управління, із використанням бібліотеки Selenium WebDriver. Такий план дозволяє своєчасно виявити вузькі місця в продуктивності та підтвердити готовність системи до розгортання.

3.1.2 Функціональне тестування

Функціональне тестування телемедичної вебсистеми проводилося методом "чорного ящика" з метою перевірки відповідності реальної поведінки застосунку закладеним бізнес-вимогам. Для систематизації процесу верифікації було розроблено набір тестових сценаріїв, які покривають як успішні шляхи взаємодії (Positive), так і поведінку системи при некоректних діях користувача (Negative). Нижче наведено приклади ключових тестових сценаріїв, за якими здійснювалася перевірка.

Тест-кейс 1

Summary: Перевірка успішного бронювання медичної консультації (Positive)

Steps to reproduce:

1. Відкрити вебзастосунок та авторизуватися під роллю пацієнта.

2. На головній панелі управління обрати лікаря з доступного списку.

3. У календарі вибрати вільну дату та натиснути на доступний часовий слот.

4. Підтвердити створення запису.

Expected results: Система успішно створює прийом, генерує унікальний ідентифікатор кімнати для відеозв'язку. Нова консультація миттєво з'являється у вкладці активних прийомів пацієнта та відповідного лікаря.

Відображення створеної консультації наведено на рисунку 3.1.

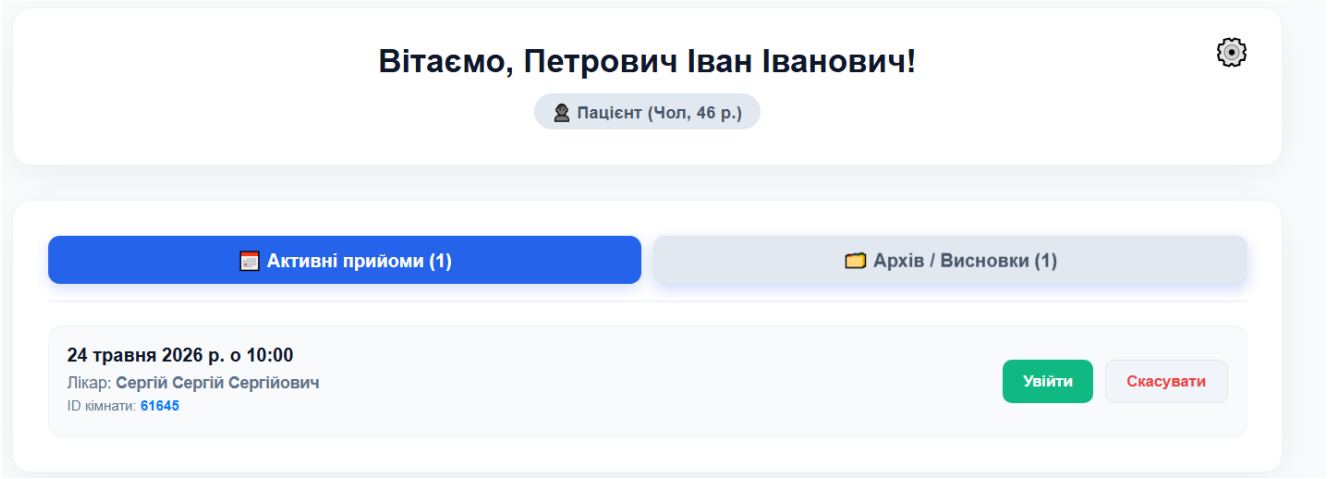

Рисунок 3.1 – Успішне бронювання медичної консультації

Тест-кейс 2

Summary: Перевірка динамічного виключення заброньованих часових слотів із розкладу лікаря (Positive)

Steps to reproduce:

1. Відкрити сайт та авторизуватися під роллю першого пацієнта.

2. Обрати лікаря, конкретну дату та забронювати певний часовий слот (наприклад, 09:00).

3. Вийти з акаунта та авторизуватися в системі під роллю другого пацієнта.

4. Перейти до профілю того самого лікаря та обрати ту саму дату для запису.

Expected results: Раніше заброньований часовий слот (09:00) повністю зникає з переліку доступних варіантів у сітці годин для другого пацієнта, що повністю унеможливлює спробу повторного запису на цей час.

Динамічне виключення забронюваного часу наведено на рисунку 3.2.

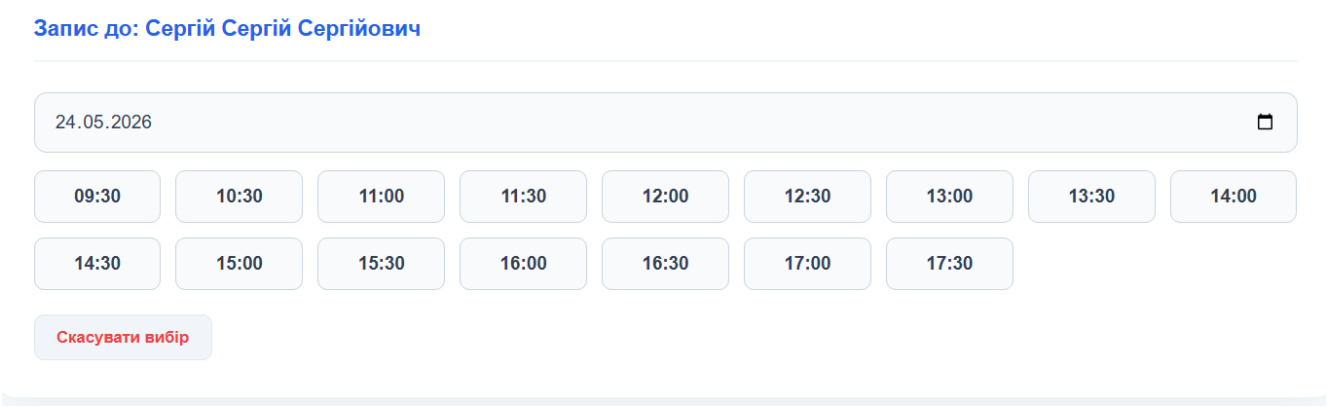

Рисунок 3.2 – Динамічне виключення забронюваного часового слоту

Тест-кейс 3 Summary: Перевірка авторизації з незаповненими обов'язковими полями (Negative)

Steps to reproduce:

1. Відкрити сторінку авторизації телемедичної системи.

2. Залишити поля електронної пошти та пароля порожніми.

3. Натиснути кнопку "Увійти".

Expected results: Система не відправляє запит на сервер, залишається на сторінці авторизації та фокусує увагу користувача на порожніх полях, вимагаючи їх обов'язкового заповнення згідно з правилами валідації HTML5.

Валідація форми при вході з порожніми полями наведена на рисунку 3.3.

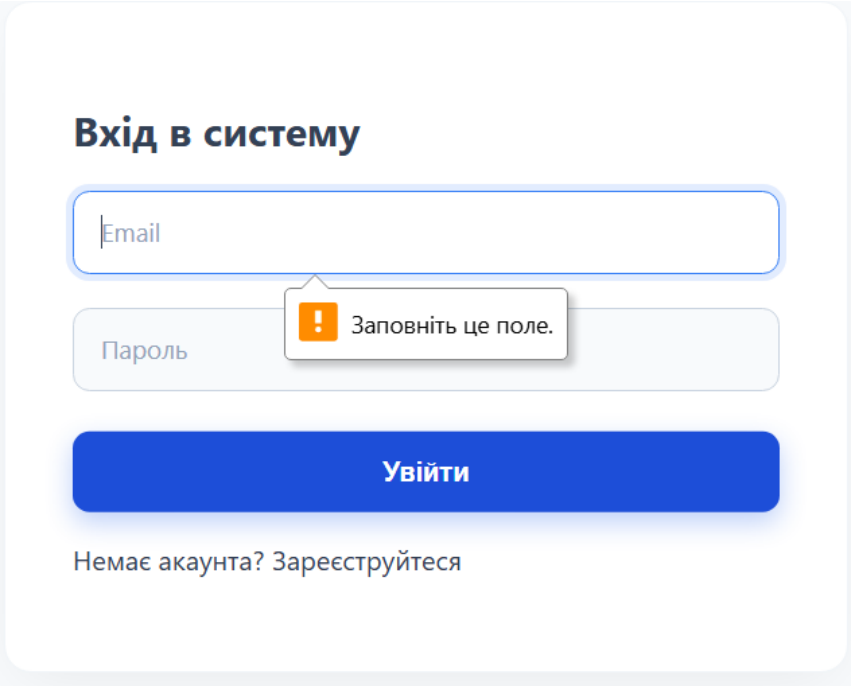

Рисунок 3.3 – Валідація форми при спробі входу з порожніми полями

Загалом, результати функціонального тестування підтвердили, що всі ключові модулі системи, включаючи механізми бронювання та перевірку колізій, працюють коректно і відповідають поставленим вимогам. Валідація форм на стороні клієнта успішно перехоплює помилки та запобігає відправці некоректних або неповних даних на сервер, що додатково забезпечує стабільність бізнес-логіки платформи.

3.1.3 Навантажувальне тестування

Для проведення навантажувального тестування було обрано інструмент Apache JMeter. Об'єктом тестування виступав розгорнутий на платформі Alwaysdata серверний програмний інтерфейс, зокрема маршрут отримання списку лікарів, який виконує ресурсомісткі агрегаційні запити до бази даних MySQL.

Сценарій тестування передбачав жорстке стресове навантаження, було налаштовано групу з 400 віртуальних користувачів, кожен з яких виконував по 2 послідовні запити протягом 5 секунд. Загальна кількість згенерованих запитів склала 800.

Налаштування параметрів HTTP-запиту та параметрів групи потоків в інструменті Apache JMeter наведено на рисунку 3.4 та рисунку 3.5.

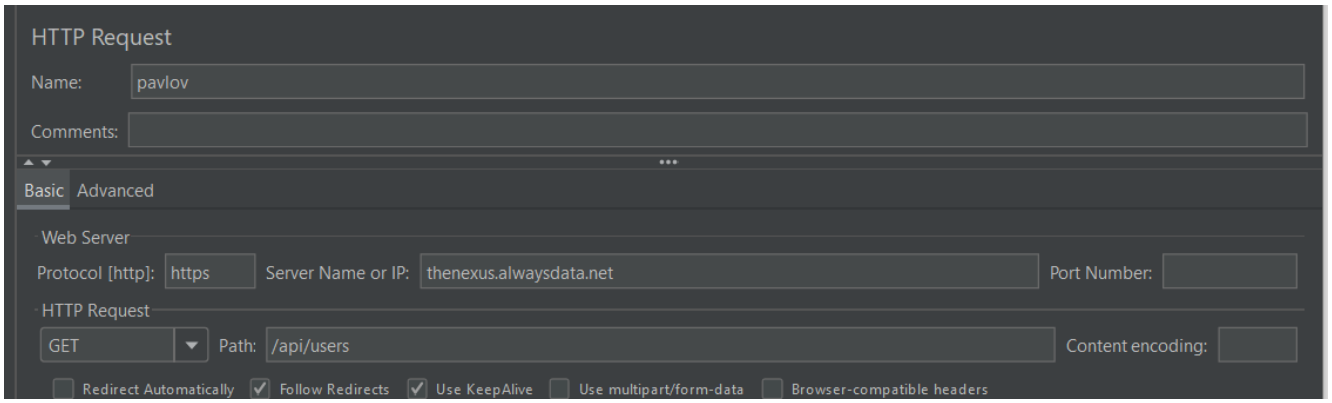

Рисунок 3.4 – Налаштування параметрів HTTP-запиту в інструменті Apache JMeter

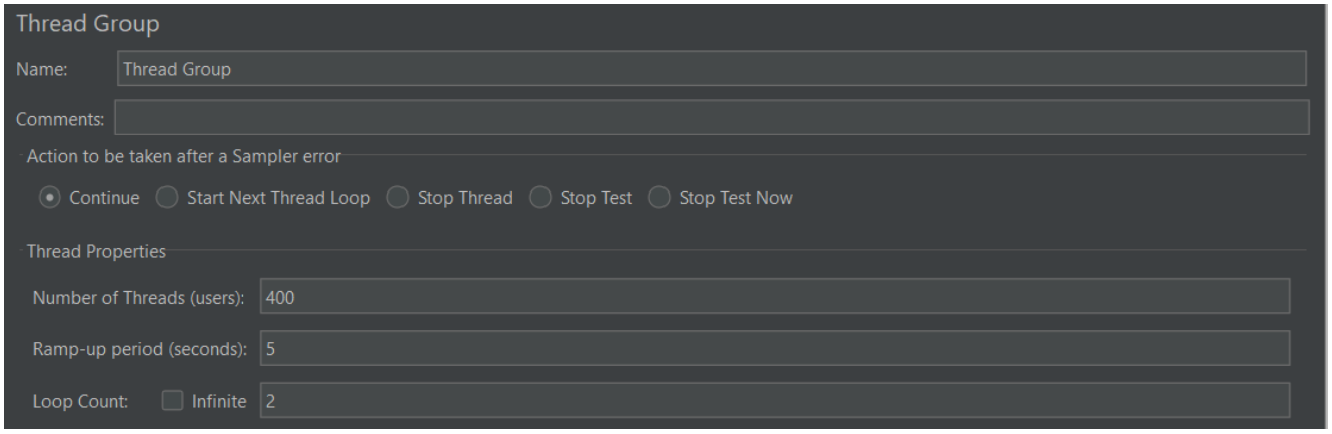

Рисунок 3.5 – Налаштування параметрів групи потоків в Apache JMeter.

Середній час обробки запиту в умовах пікового стресу склав 15346 мілісекунд. При цьому система успішно обробила 36% підключень, тоді як для 64% запитів було зафіксовано помилку.

Детальний аналіз дерева результатів показав, що причиною помилок стала не внутрішня відмова коду Internal Server Error 500, а виключення мережевого рівня — Connection timed out. Це свідчить про те, що після вичерпання ліміту пулу з'єднань з базою даних, балансувальник навантаження платформи Alwaysdata почав автоматично відхиляти нові підключення. Така поведінка є стандартною для

систем із захистом від DDoS-атак: сервер не вимкнувся повністю, а просто обмежив вхідний трафік, зберігши цілісність бази даних. Зведений звіт навантажувального тестування наведено на рисунку 3.6.

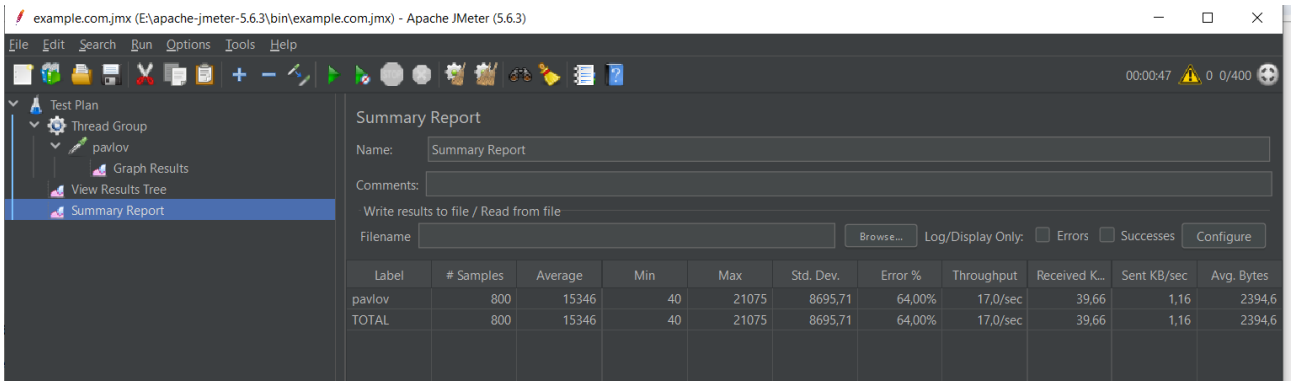

Рисунок 3.6 — Зведений звіт навантажувального тестування

Для візуалізації деградації продуктивності системи під час штучного перевантаження було побудовано графік результатів. На графіку чітко простежується лінійне зростання середнього часу відповіді та медіани по мірі збільшення кількості активних потоків, після чого графік виходить на плато через спрацювання механізмів мережевого обмеження хостингу. Візуалізацію зміни часу відповіді наведено на рисунку 3.7.

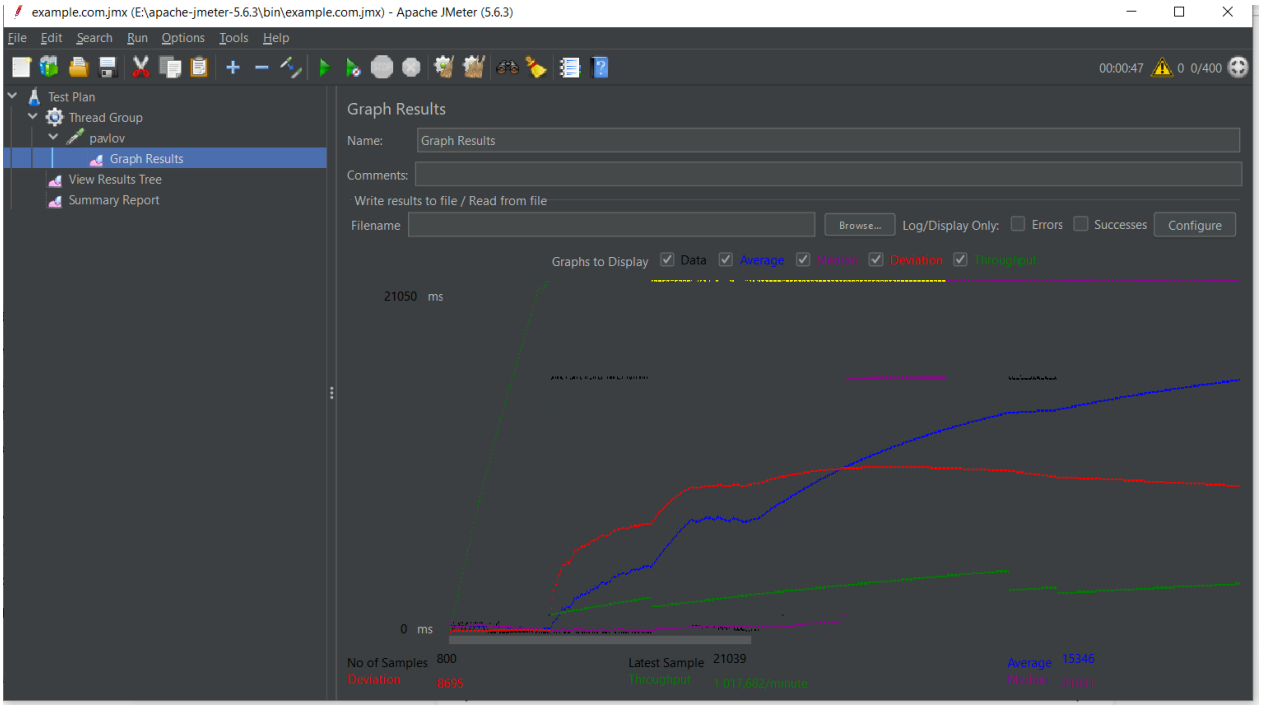

Рисунок 3.7 — Графік зміни часу відповіді сервера при піковому навантаженні

Проведене стрес-тестування довело, що закладена архітектура успішно захищає себе від перевантажень. Для реальних умов експлуатації в рамках однієї медичної клініки поточної пропускної здатності до 100-150 одночасних користувачів, цілком достатньо. У разі необхідності масштабування платформи на національний рівень, розроблений код не потребує змін, однак виникне необхідність міграції бази даних на виділений сервер із більшим пулом підключень.

3.1.4 Автоматизоване тестування

Для реалізації автоматизованого тестування було використано інструментарій Selenium WebDriver. Розроблений тестовий сценарій охоплює як негативні, так і позитивні випадки використання медичної платформи, а також перевіряє коректність роботи динамічного інтерфейсу.

Розроблений тестовий сценарій логічно поділяється на кілька послідовних етапів, кожен з яких ізольовано перевіряє окремий аспект роботи системи.

Розглянемо детальніше структуру та призначення програмних блоків даного скрипта.

На початку роботи скрипта створюється екземпляр браузера Google Chrome із застосуванням спеціальних аргументів командного рядка (log-level=3 та silent). Ці параметри приховують системні попередження рушія браузера, що дозволяє отримувати чистий консольний вивід результатів тестування без стороннього шуму.

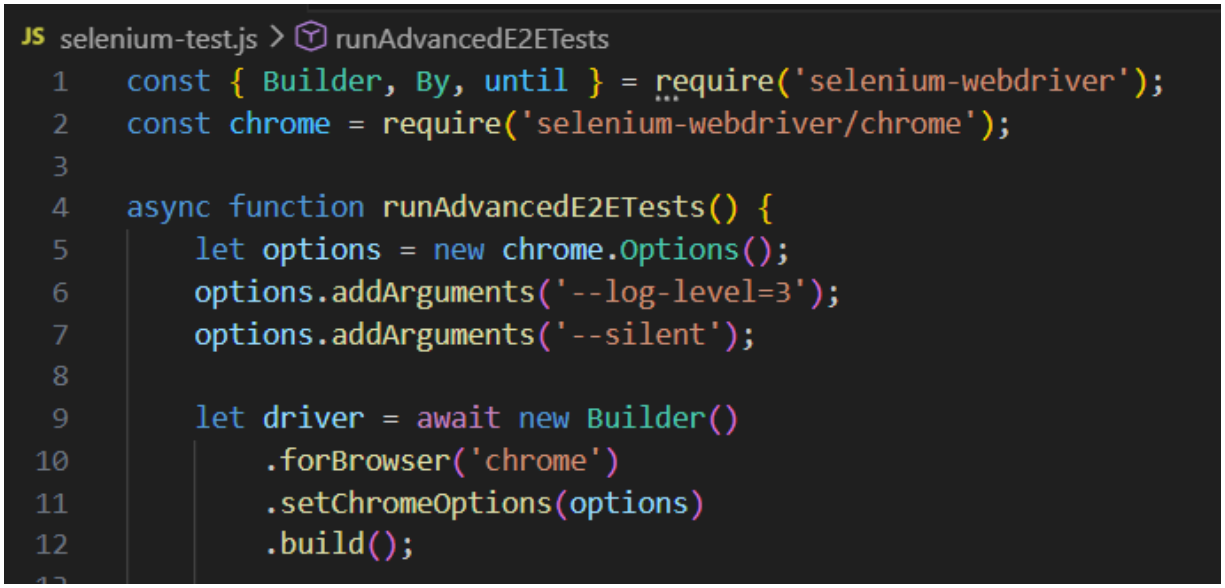

Рисунок 3.8 – Ініціалізація веб-драйвера Chrome

Перевірка обробки невірних облікових даних. Програмний бот переходить на цільову адресу застосунку, очікує завантаження DOM-дерева форми авторизації та вводить завідомо неправильний пароль. Метою цього етапу є підтвердження того, що сервер безпеки коректно відхиляє несанкціонований запит, а клієнтський застосунок перехоплює помилку HTTP та відображає спливаюче вікно користувачу без критичного збою роботи програми. Програмний сценарій перевірки обробки невірних облікових даних наведено на рисунку 3.9.

```
    // ТЕСТ 1: Негативна авторизація (неправильний пароль)
    console.log('ТЕСТ 1: Перевірка обробки помилки при введенні невірного пароля...');
    await driver.get('https://thenexus.alwaysdata.net/');
    await driver.wait(until.elementLocated(By.className('auth-form')), 10000);

    await driver.findElement(By.css('input[type="email"]')).sendKeys('test5@mail.com');
    await driver.findElement(By.css('input[type="password"]')).sendKeys('wrong_password');
    await driver.findElement(By.css('.auth-btn')).click();

    await driver.sleep(1500);
    try {
        let swalBtn = await driver.findElement(By.css('.swal2-confirm'));
        await swalBtn.click();
    } catch(e) {}
    console.log('  ✓ Успіх: Спроба входу відхилена, безпеку системи підтверджено.');
```

Рисунок 3.9 – Програмний сценарій перевірки обробки невірних облікових даних

Позитивна авторизація та обробка модальних вікон. Після негативної перевірки скрипт автоматично очищає поля введення та передає валідаційні дані реального пацієнта. Особливістю цього етапу є імітація натискання кнопки підтвердження на модальному вікні успішного входу. Це звільняє перекритий графічний інтерфейс для подальшої взаємодії та імітує усвідомлені дії людини.

```
    // ТЕСТ 2: Позитивна авторизація
    console.log('ТЕСТ 2: Авторизація користувача з коректними даними...');
    await driver.get('https://thenexus.alwaysdata.net/');

    let emailInput = await driver.findElement(By.css('input[type="email"]'));
    await emailInput.clear();
    await emailInput.sendKeys('test5@mail.com');

    let passInput = await driver.findElement(By.css('input[type="password"]'));
    await passInput.clear();
    await passInput.sendKeys('123');

    await driver.findElement(By.css('.auth-btn')).click();
    await driver.wait(until.elementLocated(By.className('dashboard-container')), 10000);


    console.log('  -> Закриття повідомлення про успішний вхід...');
    await driver.sleep(1500);
    try {
        let swalBtn2 = await driver.findElement(By.css('.swal2-confirm'));
        await swalBtn2.click();
    } catch(e) {}
    await driver.sleep(1000);

    console.log('  ✓ Успіх: Користувач авторизований, отримано доступ до Dashboard.');
```

Рисунок 3.10 – Програмний сценарій перевірки успішної авторизації

Функціональна перевірка динамічного інтерфейсу. На фінальному етапі виконується взаємодія з компонентами бібліотеки React у панелі управління. Робот знаходить на сторінці кнопки "Архів" та "Активні" за допомогою мови запитів XPath і по черзі натискає їх. Запрограмовані штучні затримки (sleep) гарантують, що асинхронні запити до сервера встигнуть виконатися, а віртуальне дерево (Virtual DOM) відмалює оновлений список медичних висновків. Програмний сценарій перевірки наведено на рисунку 3.11.

```
58
59      // ТЕСТ 3: Функціональна перевірка інтерфейсу кабінету
60      console.log('ТЕСТ 3: Перевірка взаємодії з вкладками розкладу та архіву...');
61
62      // Знаходимо кнопку перемикання на архів висновків
63      let archiveTab = await driver.findElement(By.xpath("//button[contains(text(), 'Архів')]"));
64      await archiveTab.click();
65      await driver.sleep(2000);
66      console.log('  ✓ Успіх: Вкладка архіву активована, історичні дані відрендерені.');
67
68      let activeTab = await driver.findElement(By.xpath("//button[contains(text(), 'Активні')]"));
69      await activeTab.click();
70      await driver.sleep(2000);
71      console.log('  ✓ Успіх: Повернення до активних прийомів виконано коректно.');
```

Рисунок 3.11 – Програмна взаємодія з динамічним інтерфейсом

Оскільки автоматизований драйвер Selenium виконує команди значно швидше, ніж відмальовуються інтерфейсні CSS-анімації в браузері, виникає ризик блокування подальших дій через перекриття робочих елементів екрана. Для вирішення цієї проблеми у коді реалізовано механізм асинхронних затримок (метод sleep) та безпечного фокусування на елементах через конструкцію перехоплення виключень try-catch. Програмний драйвер призупиняє виконання на час тривалості анімації появи вікна, здійснює пошук кнопки підтвердження за CSS-селектором та програмно імітує її натискання для звільнення робочої області. Використання блоку обробки помилок гарантує, що у разі мережевої затримки рендерингу або непередбачуваної відсутності модального вікна виконання основного тестового сценарію не перерветься критичним збоєм, а стабільно продовжиться згідно із заданим алгоритмом. Програмний сценарій перевірки наведено на рисунку 3.12.

```
72
73      } catch (error) {
74          console.error('❌ Помилка під час виконання сценарію:', error.message);
75      } finally {
76          await driver.sleep(3000);
77          await driver.quit();
78          console.log('ТЕСТУВАННЯ ЗАВЕРШЕНО');
79      }
80  }
```

Рисунок 3.12 – Обробка виключень під час виконання тестового сценарію

Виконання розробленого сценарію, відображене на рисунку 3.13, засвідчує високу надійність клієнтської частини та її здатність коректно обробляти динамічну зміну станів інтерфейсу.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    powershell

PS C:\Users\dimon\Desktop\test> node selenium-test

DevTools listening on ws://127.0.0.1:51037/devtools/browser/84d9759a-4b8f-479c-b634-0ee62ddbe13a
СТАРТ ТЕСТУВАННЯ
ТЕСТ 1: Перевірка обробки помилки при введенні невірного пароля...
  ✓ Успіх: Спроба входу відхилена, безпеку системи підтверджено.
ТЕСТ 2: Авторизація користувача з коректними даними...
  -> Закриття повідомлення про успішний вхід...
  ✓ Успіх: Користувач авторизований, отримано доступ до Dashboard.
ТЕСТ 3: Перевірка взаємодії з вкладками розкладу та архіву...
  ✓ Успіх: Вкладка архіву активована, історичні дані відрендерені.
  ✓ Успіх: Повернення до активних прийомів виконано коректно.
ТЕСТУВАННЯ ЗАВЕРШЕНО
PS C:\Users\dimon\Desktop\test>
```

Рисунок 3.13 — Результат автоматизованого E2E тестування

Отже, застосування автоматизованого E2E тестування дозволило верифікувати критичні користувацькі сценарії в умовах, максимально наближених до реального використання браузера. Успішне виконання скриптів підтверджує надійність клієнтської частини, стабільність роботи віртуального дерева елементів при перемиканні вкладок та безперебійну взаємодію графічного інтерфейсу із серверним API.

3.2 Розгортання та системні вимоги

Після завершення етапів програмування та успішного проходження локальних відлагоджувальних і автоматизованих тестів важливим завданням є перенесення програмного комплексу в продуктивне хмарне оточення. Процес впровадження розробленого телемедичного застосунку передбачає конфігурування віддаленого вебсервера, імпорт структури реляційної бази даних, налаштування захищених мережевих протоколів та чітке визначення вимог до апаратного й програмного забезпечення для серверної та клієнтської сторін системи.

Для розгортання серверної частини, що реалізує бізнес-логіку системи та сигнальний сервер вебсокетів, було обрано хмарну хостинг-платформу Alwaysdata. Даний вибір обґрунтований повною підтримкою середовища виконання Node.js, можливістю гнучкого налаштування процесів та наявністю вбудованих інструментів для керування базами даних MySQL. На першому етапі розгортання в адміністративній панелі хостингу було створено новий вебзастосунок, визначено робочу директорію проєкту та активовано відповідну версію рушія Node.js.

Наступним кроком стало забезпечення безпеки та ізоляції конфіденційних даних за допомогою конфігурування змінних оточення. У кореневій директорії розгорнутого сервера було створено файл параметрів середовища з розширенням env, куди було зафіксовано мережеву адресу хмарного хоста бази даних, унікальне ім'я користувача, пароль доступу та назву схеми даних. Такий підхід дозволив повністю виключити присутність секретних ключів та системних паролів у відкритому вихідному коді програми, що суттєво підвищило рівень кібербезпеки медичної платформи. Після налаштування конфігурації за допомогою веб-інтерфейсу phpMyAdmin на віддаленому сервері було виконано імпорт розробленого SQL-скрипта. Це дозволило розгорнути нормалізовану структуру таблиць користувачів, запланованих прийомів та відгуків пацієнтів, а також

автоматично встановити зв'язки між ними через механізми зовнішніх ключів із каскадним оновленням даних.

Розгортання клієнтської частини застосунку, створеної на базі бібліотеки React, вимагало проведення попередньої локальної оптимізації. За допомогою пакетного менеджера було виконано команду компіляції проєкту, яка трансформувала компонентну структуру, файли стилів та логіку маршрутизації у статичний набір файлів, мінімізованих для швидкої доставки в браузер кінцевого користувача. Оскільки архітектура розробленої системи передбачає роздачу статичного фронтенду безпосередньо сервером Express, скомпільовану папку з клієнтськими файлами було перенесено на віддалений хмарний хостинг за допомогою протоколу FTP через клієнт FileZilla. Після перезапуску процесів Node.js у панелі хостингу застосунок став доступним за публічною доменною адресою thenexus.alwaysdata.net. Для забезпечення легітимності роботи технології WebRTC було активовано та налаштовано захищений сертифікат безпеки шифрування трафіку за протоколом HTTPS, оскільки сучасні веббраузери блокують доступ до камер та мікрофонів користувачів на незахищених з'єднаннях.

Для забезпечення стабільного, безперебійного функціонування телемедичної вебсистеми та запобігання деградації продуктивності під час обслуговування користувачів було сформовано комплексні системні вимоги. Оскільки серверна частина функціонує за моделлю хмарної платформи як послуги, мінімальні вимоги до хостингового середовища включають наявність встановленого середовища Node.js версії 18.0 або вище, підтримку протоколу WebSockets для стабільної роботи сигнального сервера Socket.io, а також виділений доступ до СУБД MySQL версії 8.0 із підтримкою пулу з'єднань.

Вимоги до пристроїв кінцевих користувачів, якими є лікарі та пацієнти платформи, є мінімальними, оскільки вся логіка рендерингу інтерфейсу та первинної обробки мультимедійних потоків виконується безпосередньо всередині веббраузера. Персональний комп'ютер, ноутбук або мобільний пристрій повинен бути оснащений процесором із тактовою частотою від 1.6 гігагерца, мати не менше

4 гігабайт оперативної пам'яті, а також підключену справну веб-камеру та мікрофон для реалізації двостороннього відеозв'язку. Головною програмною вимогою є наявність встановленого сучасного браузера з повною підтримкою стандартів WebRTC, зокрема Google Chrome, Mozilla Firefox, Opera або Microsoft Edge актуальних версій. Також для забезпечення плавної трансляції аудіо- та відеосигналів високої чіткості без затримок швидкість підключення до мережі Інтернет у користувачів повинна становити не менше 5 мегабіт на секунду.

3.3 Верифікація вебзастосунку

Етап верифікації розробленого вебзастосунку полягає в систематичній перевірці відповідності створеного функціоналу початковим технічним вимогам, критеріям якості та архітектурним специфікаціям, що були визначені на етапах проєктування. Процес верифікації телемедичної платформи дозволяє документально підтвердити, що всі програмні модулі, алгоритми взаємодії реального часу та структури збереження даних функціонують коректно, забезпечуючи стабільне виконання критично важливих бізнес-процесів системи в умовах продуктивного хмарного середовища.

У межах верифікації рольової моделі доступу та користувацьких інтерфейсів було перевірено ізоляцію кабінетів пацієнта, лікаря та адміністратора. Результати випробувань показали чітке розмежування прав: авторизовані пацієнти мають доступ виключно до перегляду списку доступних фахівців, їхніх рейтингів і бронювання вільних слотів, тоді як інтерфейс лікаря успішно відображає його унікальний розклад, дані прийомів та дозволяє прикріплювати текстові медичні висновки після завершення сеансів. Перевірка адміністративної панелі підтвердила працездатність модулів агрегації системної статистики та інструментів модерації, що дозволяють оперативно видаляти некоректні відгуки пацієнтів з бази даних, очищаючи інтерфейс від спаму.

Особлива увага під час верифікації приділялася надійності роботи комунікаційного модуля реального часу. Перевірка механізмів сигналізації на базі

бібліотеки Socket.io підтвердила, що події підключення до віртуальної кімнати, обміну конфігураціями сеансу SDP та передачі ICE-кандидатів між віддаленими браузерами здійснюються асинхронно та без втрати пакетів. Після завершення сигнального етапу між браузерами пацієнта та лікаря успішно встановлюється децентралізоване пряме з'єднання, що забезпечує трансляцію потоків аудіо- та відеосигналів високої якості через сервери STUN, а інтерактивні елементи керування дозволяють стабільно вмикати та вимикати медіапристрої під час дзвінка.

Важливим елементом верифікації став захищений модуль обміну повідомленнями та документами через технологію WebRTC Data Channel. Експериментальна перевірка підтвердила коректність роботи алгоритмів фрагментації бінарних даних на стороні клієнта. Медичні документи та результати аналізів обсягом до 2 мегабайт успішно розділяються на інформаційні шматочки, транслюються безпосередньо через установлений канал даних у браузер співрозмовника та автоматично збираються у початковий файл без залучення ресурсів центрального сервера. Така децентралізована передача повністю виключає проміжне збереження конфіденційних файлів на хостингу, забезпечуючи абсолютну приватність особистої медичної інформації та відповідність міжнародним стандартам безпеки.

Наприкінці було проведено верифікацію логіки функціонування розумного календаря та блокування колізій. Перевірка підтвердила, що впроваджений на серверній та клієнтській частинах механізм контролю миттєво реагує на дії користувачів: щойно один із пацієнтів підтверджує запис на певний час, цей слот автоматично видаляється із переліку доступних годин для всіх інших користувачів, які переглядають профіль цього ж лікаря. Завдяки цьому унеможливлюється накладання прийомів або подвійне бронювання, а завершені консультації автоматично переходять до архіву із приховуванням кнопок скасування. Таким чином, результати верифікації демонструють, що розроблена телемедична

вебсистема є цілісним, безпечним та повністю працездатним програмним продуктом, який цілком відповідає завданням інженерного проєктування.

3.4 Висновки до розділу 3

У третьому розділі виконано комплексне тестування, розгортання та підсумкову верифікацію розробленої телемедичної вебсистеми.

Проведено функціональне та автоматизоване тестування End-to-End за допомогою інструментарію Selenium WebDriver, яке підтвердило стабільність відмальовки динамічного інтерфейсу на базі React та коректну обробку всіх передбачених користувацьких сценаріїв. Навантажувальне тестування з використанням Apache JMeter довело, що спроєктована архітектура має надійний захист від перевантажень, а її пропускної здатності цілком достатньо для забезпечення безперебійної роботи в реальних умовах експлуатації медичної клініки.

Успішно реалізовано розгортання серверної та клієнтської частин на хмарній хостинг-платформі Alwaysdata. Налаштовано середовище виконання Node.js, проведено імпорт реляційної бази даних MySQL, забезпечено ізоляцію конфіденційних даних через змінні оточення та активовано захищений протокол HTTPS, що є критично необхідним для функціонування технології WebRTC. Також сформовано чіткі системні вимоги до апаратного та програмного забезпечення кінцевих користувачів.

Підсумкова верифікація засвідчила безпомилкову роботу всіх ключових компонентів системи: рольової моделі доступу, «розумного» календаря для уникнення колізій при бронюванні, сигнального сервера на базі Socket.io та механізму встановлення прямого P2P-з'єднання. Підтверджено безпеку і надійність децентралізованої трансляції аудіо/відеопотоків та обміну медичними файлами через WebRTC Data Channel без їх збереження на проміжних серверах. Отже, розроблена телемедична вебсистема є цілісним, захищеним і готовим до впровадження програмним продуктом.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У процесі розробки, тестування та подальшої експлуатації телемедичного вебзастосунку ІТ-спеціалісти проводять значну частину робочого часу за персональними комп'ютерами. Робота зі складними програмними комплексами, обробка відеопотоків у реальному часі вимагають підвищеної концентрації уваги і створюють значне навантаження на організм людини. Відповідно, питання безпеки життєдіяльності та охорони праці є невід'ємною складовою успішного впровадження програмного продукту. У цьому розділі розглядаються комплекс протипожежних заходів для ІТ-офісів та медичних установ, де розгортається розроблена система а також основні вимоги до організації безпечних робочих місць користувачів ПК, створення сприятливих санітарно-гігієнічних умов.

4.1 Протипожежні заходи на підприємстві, в офісі

Сучасні ІТ-офіси розробників та кабінети телемедицини характеризуються наявністю великої кількості комп'ютерної техніки, серверного обладнання для підтримки баз даних, маршрутизаторів та щільної мережі комунікаційних кабелів. Це створює підвищене пожежне навантаження. Основними причинами пожеж в таких приміщеннях є: короткі замикання, перевантаження електромережі, несправність блоків живлення, акумулювання пилу в кулерах охолодження ПК, а також людський фактор. Базовим нормативним документом у цій сфері є «Правила пожежної безпеки в Україні» [18].

4.1.1 Організаційні протипожежні заходи

Ефективна протипожежна профілактика починається з грамотної організації робочого процесу:

– Усі працівники під час прийняття на роботу повинні проходити вступний та первинний протипожежні інструктажі, а в процесі діяльності — повторний інструктаж не рідше одного разу на рік [18, 19].

– Наказом керівника підприємства мають бути призначені особи, відповідальні за технічний стан, безпечну експлуатацію комп'ютерного обладнання та пожежну безпеку окремих кабінетів чи серверних кімнат [18].

– На видних місцях в офісі обов'язково розміщуються затверджені плани евакуації, на яких вказані основні та запасні шляхи виходу, а також місця розташування вогнегасників та кнопок пожежної сигналізації [18].

– Встановлюється сувора заборона на куріння у робочих приміщеннях (окрім спеціально відведених та обладнаних місць); для цього вивішуються відповідні заборонні знаки безпеки [18].

4.1.2 Технічні протипожежні заходи.

До технічних заходів належить забезпечення приміщень сучасними засобами виявлення та гасіння пожежі:

– Офіси повинні бути оснащені автоматичними системами пожежної сигналізації з димовими та тепловими сповіщувачами, які дозволяють виявити загоряння на ранній стадії та сповістити працівників [19].

– Електропроводка офісу має бути розрахована на сумарну потужність всієї підключеної техніки. Усі кабелі слід прокладати в спеціальних захисних коробах або гофрованих трубах, що не поширюють горіння. Крім того, необхідно регулярно (не рідше 1 разу на 2 роки) проводити заміри опору ізоляції електромережі [18].

– Забезпечення приміщень первинними засобами пожежогасіння. Для гасіння офісної та комп'ютерної техніки під напругою необхідно використовувати вуглекислотні вогнегасники. Вони мають розміщуватися у легкодоступних місцях на висоті не більше 1,5 м від рівня підлоги до верхнього краю вогнегасника та на відстані не більше 20 м від найвіддаленішого робочого місця [18].

4.1.3 Дії працівника у разі виникнення пожежі.

Готовність персоналу до дій в екстремальних ситуаціях є критично важливою. У разі виявлення пожежі або її ознак: диму, запаху горілої ізоляції працівник зобов'язаний:

– Негайно припинити роботу та знеструмити своє робоче місце.

– Повідомити Державну службу з надзвичайних ситуацій за телефоном 101, вказавши точну адресу об'єкта, місце виникнення пожежі та своє прізвище.

– Оповістити керівництво офісу чи клініки та розпочати евакуацію співробітників згідно з планом евакуації. Шляхи евакуації категорично забороняється блокувати меблями, коробками чи іншим інвентарем, а їх ширина має становити не менше 1 метра для забезпечення безперешкодного руху [19].

– Якщо загоряння невелике і не становить загрози життю, до прибуття пожежних підрозділів слід спробувати локалізувати полум'я за допомогою вуглекислотного вогнегасника, суворо дотримуючись правил власної безпеки [19].

Комплексний підхід до забезпечення пожежної безпеки в офісі та свідоме ставлення кожного працівника до вимог охорони праці дозволяють звести ризики виникнення надзвичайних ситуацій до мінімуму і гарантувати безперебійне надання телемедичних послуг.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Забезпечення безпечних умов праці для користувачів персональних комп'ютерів (ПК) та екранних пристроїв є одним із ключових завдань сучасної охорони праці. Робота за комп'ютером, зокрема під час написання програмного коду пов'язана з низкою шкідливих та небезпечних виробничих факторів: зорове напруження, гіподинамія, монотонність праці, а також вплив електромагнітного випромінювання. Тому організація робочих місць розробників та медиків повинна суворо відповідати чинним нормативним актам.

Основним документом, що регламентує безпеку та здоров'я користувачів ПК в Україні, є «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» (НПАОП 0.00-7.15-18) [20].

4.2.1 Організація та обладнання робочого місця

Робоче місце користувача ПК має бути спроектоване з урахуванням принципів ергономіки. Правильне компонування дозволяє знизити фізичне та психологічне навантаження на працівника:

– Робочий стіл та крісло: Конструкція робочого місця для виконання робіт у положенні сидячи повинна відповідати загальним ергономічним вимогам. Висота робочого столу повинна становити 700–750 мм а висота сидіння крісла 400–500 мм. Крісло працівника має бути підйомно-поворотним, з регулюванням висоти сидіння та кута нахилу спинки, що дозволяє підтримувати раціональну робочу позу [20].

– Екранні пристрої: Екран монітора повинен бути рухомим, мати можливість нахилу та повороту, кут огляду монітора 15–20° вниз, відстань до екрана 600–700 мм. Поверхня екрану — матовою або мати антиблікове покриття для уникнення віддзеркалень, які створюють додаткове напруження для зору [20].

– Клавіатура та пристрої введення: Клавіатура повинна розміщуватися на поверхні столу або на спеціальній висувній полиці на відстані 100-300 мм від краю. Це забезпечує опору для кистей рук та знижує навантаження на плечовий пояс під час написання коду чи заповнення медичних висновків [20].

4.2.2 Гігієнічні вимоги до виробничого середовища

Мікроклімат, освітлення та рівень шуму в приміщеннях, де працюють з комп'ютерною технікою, суттєво впливають на самопочуття та працездатність працівників:

– Освітлення: Робочі приміщення повинні мати природне та штучне освітлення, що відповідає сучасним будівельним нормам. Рівень освітленості

робочого місця під час роботи з екраном комп'ютера має становити 300–500 люкс. Штучне освітлення має бути загальним рівномірним, при цьому не допускається пряма та відбита блискість, що викликає швидке втомлення зорового аналізатора. Світильники слід розташовувати паралельно лінії зору користувачів [20, 21].

– Мікроклімат: У приміщеннях необхідно підтримувати оптимальні параметри мікроклімату згідно з чинними санітарними нормами. Оптимальна температура повітря повинна становити 22–25 °С, відносна вологість повітря — 40–60 %, а швидкість руху повітря — не більше 0,1 м/с. Це досягається за допомогою систем опалення, вентиляції та кондиціонування повітря [20, 21].

– Шум та вібрація: Рівень виробничого шуму не повинен перевищувати допустимі гігієнічні нормативи які для офісних приміщень і кабінетів з ПК становлять не більше 50 дБА. Системні блоки ПК, сервери маршрутизації трафіку, принтери та інше обладнання не повинні створювати надмірного шуму, що відволікає працівника і впливає на його нервову систему [20, 21].

4.2.3 Режими праці та відпочинку

Для профілактики перевтоми та збереження працездатності під час постійної роботи в системі необхідно дотримуватися раціональних режимів праці та відпочинку. Роботодавець зобов'язаний за рахунок робочого часу організовувати регламентовані перерви тривалістю 15 хвилин кожні 2 години роботи або по 10–15 хвилин кожну годину у разі інтенсивної роботи для виконання комплексу фізичних вправ, зняття зорового напруження та психофізіологічного розвантаження працівників. Крім того, для профілактики зорової втоми рекомендується робити мікропаузи тривалістю 1–2 хвилини кожні 15–20 хвилин роботи за екраном [20].

4.2 Висновки до розділу 4

У четвертому розділі було проведено комплексний аналіз питань безпеки життєдіяльності та охорони праці під час розробки, тестування та подальшої експлуатації телемедичного вебзастосунку.

На основі чинних нормативно-правових актів України обґрунтовано та сформовано перелік організаційних і технічних протипожежних заходів для ІТ-офісів та медичних установ, а також визначено чіткий алгоритм дій персоналу у разі виникнення пожеж

ВИСНОВКИ

У кваліфікаційній роботі було спроєктовано та програмно реалізовано сучасну телемедичну вебсистему для забезпечення віддалених відеоконсультацій, безпечного обміну медичними даними та управління розкладом прийомів.

Під час виконання роботи було одержано такі основні результати та сформовано висновки:

1. Проведено аналіз предметної області телемедицини та існуючих інформаційних систем. Виявлено основні недоліки централізованих рішень, зокрема високі ризики порушення конфіденційності медичних даних та перевантаження серверної інфраструктури. Обґрунтовано використання децентралізованої технології WebRTC для забезпечення оперативної та безпечної взаємодії між учасниками телемедичної системи. Визначено основних акторів системи та виконано моделювання її функціональних вимог із застосуванням UML-діаграми варіантів використання.

2. Спроєктовано архітектуру та базу даних системи. Розроблено багаторівневу клієнт-серверну архітектуру з використанням гібридної моделі передачі даних. Створено нормалізовану реляційну базу даних MySQL із підтримкою рольової моделі, яка надійно зберігає інформацію про облікові записи, електронний розклад та відгуки. За допомогою UML-діаграм класів та послідовності задокументовано об'єктно-орієнтовану структуру системи та складний процес встановлення WebRTC з'єднання.

3. Здійснено програмну реалізацію платформи з використанням сучасного JavaScript-стеку. Створено клієнтську частину у вигляді SPA-застосунку на базі React та серверну частину на базі Node.js/Express. Успішно розроблено модуль сигналізації за допомогою Socket.io та реалізовано інноваційний алгоритм децентралізованої передачі файлів через WebRTC DataChannel з розбиттям файлів до 2 МБ на фрагменти по 16 КБ для стабільної трансляції без збереження на сервері.

3. Розроблено адаптивний інтерфейс та бізнес-логіку. Створено інтуїтивно зрозумілий UI/UX дизайн із чітким розділенням кабінетів для різних ролей.

Впроваджено алгоритм «розумного» календаря, який динамічно фільтрує слоти та надійно блокує виникнення колізій при бронюванні часу, а також розроблено адміністративну панель для збору статистики та модерації відгуків.

4. Проведено комплексне тестування системи. За допомогою Selenium WebDriver виконано автоматизоване End-to-End тестування, яке підтвердило безпомилковість роботи динамічного інтерфейсу та коректну обробку сценаріїв авторизації. Навантажувальне стрес-тестування в Apache JMeter довело, що закладена архітектура має надійний захист від перевантажень, а середній час обробки запиту під піковим навантаженням склав 15346 мс, що є прийнятним для поточних масштабів.

5. Здійснено успішне розгортання вебзастосунку у продуктивному хмарному середовищі на платформі Alwaysdata. Налаштовано ізоляцію конфіденційних даних через змінні оточення, проведено імпорт бази даних та забезпечено роботу системи через захищений протокол HTTPS, що є критичною вимогою для функціонування медіапристроїв у технології WebRTC.

6. Підтверджено наукову новизну та практичне значення роботи. Вперше для подібного класу локальних систем реалізовано підхід P2P-обміну медичними даними без участі проміжних серверів. Розроблена система є повністю готовим продуктом, що дозволяє медичним установам суттєво мінімізувати витрати на серверне обладнання. Подальший розвиток проєкту може включати розширення функціоналу електронних рецептів, інтеграцію платіжних систем та впровадження підтримки групових консиліумів.

Отже, у процесі виконання кваліфікаційної роботи всі визначені завдання були успішно реалізовані, що забезпечило досягнення поставленої мети. Створена телемедична платформа підтвердила ефективність використання децентралізованої архітектури, продемонструвала надійність функціонування, належний рівень захисту даних і може слугувати основою для подальшого практичного застосування в галузі охорони здоров’я.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про внесення змін до деяких законодавчих актів України щодо функціонування телемедицини: Закон України від 09.08.2023 № 3301-IX. URL: https://zakon.rada.gov.ua/laws/show/3301-20#Text

2. World Health Organization. Consolidated telemedicine implementation guide. Geneva : World Health Organization, 2022. 68 p. URL: https://www.who.int/publications/i/item/9789240059184

3. ДСТУ EN ISO 13606-1:2023. Інформатизація охорони здоров'я. Передавання електронної медичної картки. Частина 1. Еталонна модель (EN ISO 13606-1:2019, IDT; ISO 13606-1:2019, IDT). Київ : ДП «УкрНДНЦ», 2023. 85 с.

4. Mahmoud H., Abozariba R. A systematic review on WebRTC for potential applications and challenges beyond audio video streaming. Multimedia Tools and Applications. 2024. Vol. 84. P. 2909–2946.

5. Sahu S. K., Gharbaoui M., Ruscelli A. L., Cecchetti G., Sgambelluri A., Castoldi P. Remote Medical Support in Emergency Scenarios: A WebRTC-Based Solution. IEEE Communications Magazine. 2024. P. 1–7.

6. Stolley K. Programming WebRTC: Build Real-Time Streaming Applications for the Web. Pragmatic Bookshelf, 2024. 266 p.

7. WebRTC For The Curious: Open source WebRTC guide. 2021. URL: https://webrtcforthecurious.com/.

8. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). Version 4.0 / ed. H. Washizaki. IEEE Computer Society, 2024. 411 p.

9. Ford N., Richards M., Sadalage P., Dehghani Z. Software Architecture: The Hard Parts. O'Reilly Media, 2021. 432 p.

10. MySQL 8.0 Reference Manual. Oracle Corporation. URL: https://dev.mysql.com/doc/refman/8.0/en/

11. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 4th ed. No Starch Press, 2024. URL: https://eloquentjavascript.net/

12. React Documentation. Meta Open Source. URL: https://react.dev/

13. Hasnain M., Ullah S. Learning and programming challenges of React.js open-source framework. Library Hi Tech News. 2023. Vol. 40, No. 8. P. 12–15.

14. Node.js v26.x Documentation. OpenJS Foundation. URL: https://nodejs.org/docs/latest/api/

15. Pratama I. P. A. E., Raharja I. M. S. Node.js Performance Benchmarking and Analysis at Virtualbox, Docker, and Podman Environment Using Node-Bench Method. JOIV: International Journal on Informatics Visualization. 2023. Vol. 7, No. 4. P. 2240–2248.

16. Socket.IO Documentation. URL: https://socket.io/docs/v4/

17. MDN Web Docs: WebRTC API. Mozilla. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

18. Правила пожежної безпеки в Україні : затверджені наказом Міністерства внутрішніх справ України від 30.12.2014 № 1417. Верховна Рада України : офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z0252-15.

19. Сокуренко В. В. Безпека життєдіяльності та охорона праці : підручник / В. В. Сокуренко, О. М. Бандурка, С. М. Бортник та ін. ; за заг. ред. В. В. Сокуренка. – Харків : Харківський національний університет внутрішніх справ, 2021. – 308 с.

20. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : затверджені наказом Міністерства соціальної політики України від 14.02.2018 № 207. Верховна Рада України : офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z0508-18

21. Желібо Є. П. Основи охорони праці : підручник / Є. П. Желібо, М. П. Гандзюк, М. О. Халімовський. – Київ : Каравела, 2023. – 384 с.

22. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 45 с. URL: https://elartu.tntu.edu.ua/handle/lib/50317

23. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) https://ceur-ws.org/Vol-4057/

24. Tsupryk, H., Olianin, D. (2025). Vydobuvannia danyh z tekstu vykorystovuiuchy transformerni neironni merezhi [Data extraction from text using Transformer Neural Networks]. Information Technology: Computer Science, Software Engineering and Cyber Security, 125–130, DOI: https://doi.org/10.32782/IT/2025-2-13

25. ОЛЯНІН D., & ЦУПРИК Н. (2025). Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing devices in technological processes, 82(2), 360–364. https://doi.org/10.31891/2219-9365-2025-82-52

26. Tsupryk H. LLM-based Extraction from Resumes / D. Olianin, H. Tsupryk // Advanced Technologies in Scientific Research: collection of scientific papers with proceedings of the 1st International Scientific and Practical Conference, Rotterdam, Netherlands, 20–22 August 2025. – International Scientific Unity, 2025. – 72-76

27. Yaroslav Kotov, Evhenia Yavorska, Halyna Tsupryk, Róża Dzierżak 1 , Oleksandr Reshetnik, Viktoriia Bokovets (2025) Evaluating interoperability and data quality in FHIR-based AI assessment pipelines. Proc. SPIE 14009, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2025, 140091F (30 December 2025) https://doi.org/10.1117/12.3100561

ДОДАТКИ

ДОДАТОК А

Тези конференції

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет
Чернівецький національний університет
імені Юрія Федьковича
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

IX МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2026 р.

(збірник тез конференції)

Тернопіль 2026

УДК 004.42
Павлов Д. – ст. гр. СП-42
Тернопільський національний технічний університет імені Івана Пулюя

АРХІТЕКТУРНІ РІШЕННЯ ДЛЯ ПОБУДОВИ ТЕЛЕМЕДИЧНИХ СИСТЕМ НА БАЗІ ТЕХНОЛОГІЇ WEBRTC

Науковий керівник: к.т.н., доцент Г. Б. Цуприк

Pavlov D.
Ternopil Ivan Puluj National Technical University

ARCHITECTURAL SOLUTIONS FOR BUILDING TELEMEDICINE SYSTEMS BASED ON WEBRTC TECHNOLOGY

Supervisor: PhD, Associate Professor H. B. Tsupryk

Ключові слова: телемедицина, відеоконференцзв'язок, електронна охорона здоров'я.
Keywords: telemedicine, videoconferencing, eHealth.

У сучасних умовах розвитку цифрової охорони здоров’я надзвичайної актуальності набуває впровадження спеціалізованих телемедичних систем, здатних забезпечити оперативний та безпечний зв'язок між лікарем і пацієнтом. Зростання попиту на віддалені медичні послуги вимагає створення надійних платформ, які гарантують високу якість потокового відео та дотримання строгих стандартів конфіденційності. Відповідно, розробка сучасних вебзастосунків для відеоконсультацій без необхідності встановлення стороннього програмного забезпечення є важливим науково-практичним завданням.

Для забезпечення наукової обґрунтованості розробки було проведено системний аналіз вимог до телемедичних сервісів, що дозволило виокремити пріоритетні завдання проєктування. Зокрема, встановлено необхідність поєднання високої якості відеозв'язку із суворим дотриманням стандартів захисту персональних даних пацієнтів. Сформований перелік функціональних вимог охоплює забезпечення стабільності з'єднання при низькій пропускній здатності мережі, повну кросбраузерність та можливість інтерактивного обміну медичною документацією безпосередньо під час активного відеосеансу за допомогою механізму DataChannel.

Архітектура запропонованого програмного рішення базується на використанні технології WebRTC, що дозволяє організувати пряму трансляцію медіапотоків між клієнтами безпосередньо у браузері. Клієнт-серверна модель включає фронтенд, побудований за допомогою React.js, та сервіс сигналізації на базі Node.js. Для надійного збереження профілів користувачів та розкладів прийомів задіяна реляційна модель даних. Враховуючи високі стандарти медичної таємниці, відео- та аудіотрафік маршрутизується виключно через peer-to-peer канали із застосуванням наскрізного шифрування DTLS та SRTP, а на сервері фіксуються лише технічні метадані.

У процесі розробки здійснено комплексне тестування створених модулів. Для підвищення точності та надійності роботи систем у критичних умовах доцільним є використання алгоритмів порівняльної статистичної верифікації реакцій на навантаження. Важливою перевагою розробленої платформи є її масштабованість, що створює основу для інтеграції з медичними інформаційними системами (MIC). При розробці таких інтеграцій критично важливо забезпечувати високу якість даних та

інтероперабельність, зокрема на базі медичного стандарту FHIR, що дозволяє автоматизувати обмін медичною інформацією.

Перспективи подальшого розвитку проєкту полягають у впровадженні інтелектуальних аналітичних модулів. Зокрема, розглядається можливість застосування сучасних архітектур нейронних мереж, таких як трансформери, для автоматичного транскрибування консультацій та генерування попередніх лікарських висновків. Таке розширення функціонала суттєво знизить рутинне навантаження на медичний персонал та підвищить технологічну цінність розробленої платформи.

Література:

1. Methods of constructing algorithms for comparative test statistical verification of mathematical models of bioobject responses to low-intensity stimuli / Bohdan Yavorskyy, Evhenia Yavorska, Halyna Tsupryk, Roman Kinash // Scientific Journal of TNTU. — Tern.: TNTU, 2023. — Vol 112. — No 4. — P. 82–90.
2. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) https://ceur-ws.org/Vol-4057/
3. Yaroslav Kotov, Evhenia Yavorska, Halyna Tsupryk, Róża Dzierżak 1 , Oleksandr Reshetnik, Viktoriia Bokovets (2025) Evaluating interoperability and data quality in FHIR-based AI assessment pipelines. Proc. SPIE 14009, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2025, 140091F (30 December 2025) https://doi.org/10.1117/12.3100561
4. Johnston A. B. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web / A. B. Johnston, D. C. Burnett. – 2014. – 312 p.

ДОДАТОК Б

Програмний код компонента відеокімнати VideoRoom

```
import React, { useEffect, useRef, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import io from 'socket.io-client';
import ChatBox from './ChatBox';
import './VideoRoom.css';

const SOCKET_SERVER_URL = '/';

const VideoRoom = ({ roomId }) => {
    const localVideoRef = useRef(null);
    const remoteVideoRef = useRef(null);
    const peerConnectionRef = useRef(null);
    const socketRef = useRef(null);
    const dataChannelRef = useRef(null);
    const fileChunksRef = useRef({}); // Сховище для шматочків файлів
    const navigate = useNavigate();

    const [error, setError] = useState('');
    const [messages, setMessages] = useState([]);
    const [isAudioMuted, setIsAudioMuted] = useState(false);
    const [isVideoOff, setIsVideoOff] = useState(false);

    useEffect(() => {
        socketRef.current = io(SOCKET_SERVER_URL);

        navigator.mediaDevices.getUserMedia({ video: true, audio:
true })
            .then((stream) => {
```

```
                if (localVideoRef.current) localVideoRef.current.srcObject = stream;


                const peerConnection = new RTCPeerConnection({
                    iceServers: [{ urls: 'stun:stun.l.google.com:19302' }]
                });
                peerConnectionRef.current = peerConnection;


                const dataChannel = peerConnection.createDataChannel('chat');
                dataChannelRef.current = dataChannel;


                // ОБРОБКА ОТРИМАНИХ ДАНИХ (Текст + Файли)
                const handleReceiveMessage = (event) => {
                    try {
                        const data = JSON.parse(event.data);
                        if (data.type === 'text') {
                            setMessages(prev => [...prev, { sender: 'other', text: data.text }]);
                        } else if (data.type === 'file-start') {
                            // Початок прийому файлу
                            fileChunksRef.current[data.fileId] = { chunks: [], total: data.total, name: data.fileName };
                        } else if (data.type === 'file-chunk') {
                            // Збираємо шматочки
                            const fileData = fileChunksRef.current[data.fileId];
                            if (fileData) {
                                fileData.chunks.push(data.chunk);
                                // Якщо зібрали всі шматочки - показуємо файл
                                if (fileData.chunks.length === fileData.total) {
```

```
                                    const fullDataUrl = fileData.chunks.join('');
                                    setMessages(prev => [...prev, { sender: 'other', isFile: true, fileName: fileData.name, fileData: fullDataUrl }]);
                                    delete fileChunksRef.current[data.fileId]; // Очищаємо пам'ять
                                }
                            }
                        }
                    } catch (e) {
                        setMessages(prev => [...prev, { sender: 'other', text: event.data }]);
                    }
                };

                dataChannel.onmessage = handleReceiveMessage;

                peerConnection.ondatachannel = (event) => {
                    const receiveChannel = event.channel;
                    dataChannelRef.current = receiveChannel;
                    receiveChannel.onmessage = handleReceiveMessage;
                };

                stream.getTracks().forEach(track => peerConnection.addTrack(track, stream));

                peerConnection.ontrack = (event) => {
                    if (remoteVideoRef.current) remoteVideoRef.current.srcObject = event.streams[0];
                };

                peerConnection.onicecandidate = (event) => {
```

```
                    if (event.candidate) socketRef.current.emit('ice-
candidate', event.candidate, roomId);
                };

                socketRef.current.emit('join-room',          roomId,
socketRef.current.id);

                socketRef.current.on('user-connected', async () => {
                    const offer = await peerConnection.createOffer();
                    await peerConnection.setLocalDescription(offer);
                    socketRef.current.emit('offer', offer, roomId);
                });

                socketRef.current.on('offer', async (offer) => {
                    await    peerConnection.setRemoteDescription(new
RTCSessionDescription(offer));
                    const          answer          =          await
peerConnection.createAnswer();
                    await peerConnection.setLocalDescription(answer);
                    socketRef.current.emit('answer', answer, roomId);
                });

                socketRef.current.on('answer', async (answer) => {
                    await    peerConnection.setRemoteDescription(new
RTCSessionDescription(answer));
                });

                socketRef.current.on('ice-candidate',          async
(candidate) => {
                    try  {  await  peerConnection.addIceCandidate(new
RTCIceCandidate(candidate)); }
                    catch (e) { console.error(e); }
                });
            })
```

```
            .catch(err => {
                setError('Не вдалося отримати доступ до камери або мікрофона.');
            });

        return () => {
            if (localVideoRef.current && localVideoRef.current.srcObject) {
                localVideoRef.current.srcObject.getTracks().forEach(track => track.stop());
            }
            if (socketRef.current) socketRef.current.disconnect();
            if (peerConnectionRef.current) peerConnectionRef.current.close();
        };
    }, [roomId]);

    // ВІДПРАВКА ТЕКСТУ
    const sendMessage = (text) => {
        if (dataChannelRef.current && dataChannelRef.current.readyState === 'open') {
            dataChannelRef.current.send(JSON.stringify({ type: 'text', text }));
            setMessages(prev => [...prev, { sender: 'me', text }]);
        } else {
            alert('З\'єднання зі співрозмовником ще не встановлено!');
        }
    };

    // ВІДПРАВКА ФАЙЛУ (Нарізка та P2P передача)
    const sendFile = (file) => {
        if (!dataChannelRef.current || dataChannelRef.current.readyState !== 'open') {
```

```
            alert('З\'єднання ще не встановлено!');
            return;
        }
        if (file.size > 2 * 1024 * 1024) { // Ліміт 2 МБ, щоб браузер не завис
            alert('Файл занадто великий! Максимальний розмір для прямої передачі: 2 МБ.');
            return;
        }

        const reader = new FileReader();
        reader.onload = (e) => {
            const dataUrl = e.target.result;
            const chunkSize = 16384; // 16 КВ на шматок
            const totalChunks = Math.ceil(dataUrl.length / chunkSize);
            const fileId = Math.random().toString(36).substring(7);

            // 1. Сповіщаємо про початок відправки
            dataChannelRef.current.send(JSON.stringify({ type: 'file-start', fileId, fileName: file.name, total: totalChunks }));

            // 2. Відправляємо шматочки з мікро-затримкою (щоб не перевантажити канал)
            let offset = 0;
            const sendNextChunk = () => {
                if (offset < dataUrl.length) {
                    const chunk = dataUrl.slice(offset, offset + chunkSize);
                    dataChannelRef.current.send(JSON.stringify({ type: 'file-chunk', fileId, chunk }));
                    offset += chunkSize;
                    setTimeout(sendNextChunk, 10);
                } else {
```

```
                    // Файл успішно відправлено
                    setMessages(prev => [...prev, { sender: 'me', isFile: true, fileName: file.name, fileData: dataUrl }]);
                }
            };
            sendNextChunk();
        };
        reader.readAsDataURL(file); // Конвертуємо файл у Base64 рядок
    };

    const toggleAudio = () => {
        const stream = localVideoRef.current?.srcObject;
        if (stream) {
            const audioTrack = stream.getAudioTracks()[0];
            if (audioTrack) {
                audioTrack.enabled = !audioTrack.enabled;
                setIsAudioMuted(!audioTrack.enabled);
            }
        }
    };

    const toggleVideo = () => {
        const stream = localVideoRef.current?.srcObject;
        if (stream) {
            const videoTrack = stream.getVideoTracks()[0];
            if (videoTrack) {
                videoTrack.enabled = !videoTrack.enabled;
                setIsVideoOff(!videoTrack.enabled);
            }
        }
    };
```

```
    const handleEndCall = () => {
        if (localVideoRef.current && localVideoRef.current.srcObject)
{
localVideoRef.current.srcObject.getTracks().forEach(track =>
track.stop());
        }
        if (peerConnectionRef.current)
peerConnectionRef.current.close();
        if (socketRef.current) socketRef.current.disconnect();
        navigate('/');
    };

    return (
        <div style={{ width: '100%' }}>
            {error && <p style={{ color: 'red', textAlign: 'center',
fontWeight: 'bold' }}>{error}</p>}

            <div className="room-container">
                <div className="video-section">
                    <div className="video-grid">
                        <div className="video-wrapper">
                            <p className="video-label">Ви</p>
                            <video  className="local-video"
ref={localVideoRef} autoPlay muted playsInline />
                        </div>
                        <div className="video-wrapper">
                            <p  className="video-
label">Співрозмовник</p>
                            <video  ref={remoteVideoRef}  autoPlay
playsInline />
                        </div>
                    </div>
```

```
                    <div className="controls-bar">
                        <button title={isAudioMuted ? "Увімкнути мікрофон" : "Вимкнути мікрофон"} onClick={toggleAudio} className={`control-btn btn-media ${isAudioMuted ? 'off' : ''}`}>
                            {isAudioMuted ? '🔇' : '🎤'}
                        </button>
                        <button title={isVideoOff ? "Увімкнути камеру" : "Вимкнути камеру"} onClick={toggleVideo} className={`control-btn btn-media ${isVideoOff ? 'off' : ''}`}>
                            {isVideoOff ? '📵' : '📹'}
                        </button>
                        <button title="Завершити дзвінок" onClick={handleEndCall} className="control-btn btn-end">
                            📞
                        </button>
                    </div>
                </div>

                {/* Передаємо функцію sendFile у чат */}
                <ChatBox messages={messages} sendMessage={sendMessage} sendFile={sendFile} />
            </div>
        </div>
    );
};

export default VideoRoom;
```

ДОДАТОК В

Посилання на репозиторій GitHub

Повний програмний код знаходиться в репозиторії GitHub. Репозиторій проєкту: URL: https://github.com/TNTU-121-Software-Engineering/2025-2026-KRB-SE-42-Dmytro-PAVLOV