НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебплатформа для телемедицини. Клієнтська частина»

Виконав:
студент IV курсу, групи КП-93
Петренко Нікіта Андрійович _____

Керівник:
Асистент кафедри ПЗКС
Жикін Юрій Сергійович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Асистент кафедри СПСКС
Щербина Богдан Олександрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Петренку Нікіті Андрійовичу

1.Тема проєкту «Вебплатформа для телемедицини. Клієнтська частина», керівник проєкту Жикін Юрій Сергійович, асистент кафедри ПЗКС, затверджені наказом по університету від «31» травня 2023 р. № 2107-с

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
- аналіз предметної області та існуючих рішень;
- аналіз технологій розроблення клієнтської частини вебзастосунків;
- структурна організація розроблених програмних засобів;
- аналіз реалізації програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
- Процес проведення відеоконференції. UML-діаграма послідовності (креслення);
- алгоритм управління записами на прийом. Блок-схема алгоритму (креслення);
- архітектура модулів клієнтської частини (плакат).

– структура вебсторінок (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2022	
2.	Розроблення та узгодження технічного завдання	25.11.2022	
3.	Розроблення структури клієнтської частини вебплатформи	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2023	
7.	Програмна реалізація клієнтської частини вебплатформи	10.03.2023	
8.	Тестування клієнтської частини вебплатформи	17.03.2023	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2023	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2023	
11.	Підготовка графічної частини дипломного проєкту	21.04.2023	
12.	Оформлення документації дипломного проєкту	26.05.2023	

Студент Нікіта ПЕТРЕНКО

Керівник проєкту Юрій ЖИКІН

АНОТАЦІЯ

Даний дипломний проєкт присвячено розробленню клієнтської частини вебплатформи для телемедицини.

Задача дипломного проєкту полягає у тому, щоб розробити клієнтську частину вебплатформи для телемедицини, що втілює у собі основну функціональність сучасної вебплатформи для телемедицини, а саме: електронний запис на прийом, відеоконференції та чат, експорт та імпорт документів. Клієнтська частина має бути легко розширюваною і масштабованою, для забезпечення можливості доповнення застосунку додатковою функціональністю.

У роботі виконано порівняльний аналіз існуючих застосунків (мобільних, веб) для телемедицини. Був проведений аналіз та обґрунтування вибору технологій, та допоміжних бібліотек клієнтської частини для реалізації даної вебплатформи. Основною функціональністю розробленої вебплатформи є надання можливості лікарям та клієнтам клініки проводити прийоми в дистанційному режимі, за допомогою відеоконференцій та чату, та мати можливість завантажувати документи, такі як, наприклад, електронні рецепти чи заключення лікаря. Клієнт клініки може самостійно записатися на прийом, використовуючи розроблену вебплатформу, може скасовувати власні записи на прийом. У системі наявні три ролі користувачів: «Лікар», «Пацієнт», «Адміністратор». Кожна роль має свою унікальну функціональність.

В даному проєкті розроблено та досліджено: архітектуру клієнтської частини вебплатформи, алгоритми керування записами на прийом, алгоритми керування доступами користувачів, алгоритми проведення відеоконференцій та чатів, а також механізми експорту та імпорту файлів.

ABSTRACT

This diploma project is dedicated to the development of the client-side web application for telemedicine.

The objective of the diploma project is to develop the client-side component of a web application for telemedicine that encompasses the essential functionality of a modern telemedicine application, including appointment scheduling, video conferencing, chat, and document export/import. The client-side component should be easily extensible and scalable to accommodate additional functionality.

The work includes a comparative analysis of existing telemedicine applications (both mobile and web-based), an analysis and justification of technology selection, and the use of auxiliary libraries for implementing the client-side of the web application. The primary functionality of the developed web application is to enable remote consultations between healthcare professionals and patients through video conferencing and chat, as well as facilitate document uploads such as electronic prescriptions or medical reports. Clinic clients can self-schedule appointments using the developed web application and have the ability to cancel their own appointments. The system includes three user roles: "Doctor," "Patient," and "Administrator," each with their unique set of functionalities.

The following components were researched and developed in the project: the architecture of the client-side web application, appointment management algorithms, user access control algorithms, video conferencing and chat algorithms, as well as file export/import mechanisms.

ДП.045440-01-90. Вебплатформа для телемедицини. Клієнтська частина
Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебплатформа для	6	
	телемедицини. Клієнтська		
	частина. Технічне		
	завдання		
ДП.045440-03-81	Вебплатформа для	72	
	телемедицини. Клієнтська		
	частина. Посяснювальна		
	записка		
ДП.045440-04-51	Вебплатформа для	4	
	телемедицини. Клієнтська		
	частина. Програма та		
	методика тестування		
ДП.045440-05-34	Вебплатформа для	21	
	телемедицини. Клієнтська		
	частина. Керівництво		
	користувача		
ДП.045440-06-99	Вебплатформа для	1	
	телемедицини. Клієнтська		
	частина. Процес		
	проведення		
	відеоконференції.		
	UML-діаграма		
	Послідовності		
ДП.045440-07-99	Вебплатформа для	1	
	телемедицини. Клієнтська		
	частина. Алгоритм		
	управління записами на		
	прийом. Блок-схема		
	алгоритму		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-08-98	Вебплатформа для телемедицини. Клієнтська частина. Компакт-диск	1	

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ВЕБПЛАТФОРМА ДЛЯ ТЕЛЕМЕДИЦИНИ. КЛІЄНТСЬКА ЧАСТИНА

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Юрій ЖИКІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Нікіта ПЕТРЕНКО

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: вебплатформа для телемедицини. Клієнтська частина.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для проведення лікарських консультацій в режимі онлайн.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Для усіх користувачів у системі повинен бути спільний доступ до наступної функціональності системи:

1. Авторизація.
2. Перегляд власної облікової інформації

Для обох ролей як «Лікар» так і «Пацієнт», повинна бути доступною наступна функціональність системи:

1. Робота з відеодзвінками (використання аудіо та відео входу під час дзвінка, підключення до дзвінка, відключення від дзвінка, ввімкнення аудіо та відео під час дзвінка).
2. Робота з чатами (перегляд історії повідомлень у чаті, відправка повідомлень, надсилання медіа та файлового контенту наступних

розширень: PDF, DOC, DOCX, JPG, JPEG, PNG, GIF).

Для користувачів із роллю «Пацієнт» повинен бути доступ до такої функціональності системи:

1. Пошук послуги (за частиною ПІБ лікаря, за спеціалізацією, наприклад, кардіологія).
2. Перегляд інформації про послугу (список лікарів, що надають).
3. Перегляд основної інформації про лікаря (стаж, категорія, список послуг, що надає, найближчий час коли можна записатися на послугу, що надає).
4. Керування власними прийомами (надсилання заявки на прийом, скасування, перегляд інформації щодо власних прийомів).
5. Оцінка якості візиту (Оцінка якості відеоконференції, оцінка якості надання послуги лікарем).
6. Керування власними виписками та історіями хвороби (перегляд, та експорт файлів).
7. Редагування власних облікових даних.

Для користувачів із роллю «Лікар» доступна наступна функціональність системи:

1. Керування прийомами у електронній черзі (перегляд існуючих прийомів у розділі електронної черги, відхилення назначених прийомів із вказанням причини).
2. Керування виписками та історіями хвороби пацієнтів (перегляд, імпорт та експорт файлів наступних розширень: PDF, DOC, DOCX, JPG, JPEG, PNG, GIF).

Для користувачів із роллю «Адміністратор» доступна наступна функціональність системи:

1. Створення облікових записів лікарів.
2. Керування зареєстрованими у системі лікарями (перегляд зареєстрованих лікарів, редагування інформації лікарів, активація та деактивація облікових записів).

3. Керування записами на прийом, що на розгляді (перегляд інформації про таки записи, редагування часу та скасування прийому).

Додаткові вимоги:

1. Кросплатформність клієнтської частини вебплатформи: підтримка Google Chrome (від 95 версії), Firefox (від 85 версії), Edge (від 100 версії) на операційній системі Windows 10.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення:
   - «Управління записами на прийом. Схема алгоритму»;
   - «Проведення відеоконференцій у вебплатформі. UML діаграма послідовності».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи..... 13.11.2022
Розроблення та узгодження технічного завдання..... 25.11.2022
Розроблення структури клієнтської частини вебплатформи..... 15.12.2022
Розроблення дизайну сторінок та графічних елементів..... 03.02.2023
Програмна реалізація клієнтської частини вебплатформи..... 10.03.2023
Тестування клієнтської частини вебплатформи 17.03.2023
Підготовка матеріалів текстової частини проєкту..... 28.04.2023
Підготовка матеріалів графічної частини проєкту 21.04.2023
Оформлення технічної документації проєкту..... 26.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Завідувач кафедри
_____ Євгенія СУЛЕМА
"___" _____ 2023 р.

ВЕБПЛАТФОРМА ДЛЯ ТЕЛЕМЕДИЦИНИ. КЛІЄНТСЬКА ЧАСТИНА.

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____Юрій ЖИКІН

Нормоконтроль: Виконавець:

_____Микола ОНАЙ _____ Нікіта ПЕТРЕНКО

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

AJAX – Asynchronous JavaScript and XML – в це технологія, яка дозволяє взаємодіяти з сервером без перезавантаження сторінки.

JSON – JavaScript Object Notation – це формат обміну даними, який забезпечує легку і зрозумілу структуру даних, що передаються між різними системами. Формат JSON є текстовим і може бути використаний з більшістю мов програмування, включаючи JavaScript, Python, PHP та багато інших.

JWT – JSON Web Token – це мінімалістичний стандарт для передачі токенів безпеки на основі JSON. Він використовується для автентифікації користувачів та забезпечення безпеки даних.

UI – User Interface (укр. користувацький інтерфейс) – це все, що стосується того, як користувач взаємодіє з вебзастосунок або мобільним застосунком.

UX – User Experience (укр. користувацький досвід) – це дизайн і розміщення елементів на сторінках сайту або в застосунку, щоб забезпечити максимальний комфорт користування.

RESTful API – це загальні принципи організації взаємодії з програмним забезпеченням за протоколом HTTP методами GET, POST, PUT і DELETE.

MVC – Model-View-Controller – це архітектурний шаблон програмування, який дозволяє розділити логіку програми на три основні компоненти: Модель (Model), Представлення (View) та Контролер (Controller). Модель відповідає за управління даними та бізнес-логікою застосунку. Представлення відповідає за відображення даних користувачу. Контролер відповідає за керування взаємодією між моделлю та представленням.

CRUD – Create, Read, Update, Delete – базові функції управління даними.

CDN – Content Delivery Network – це мережа серверів, яка дозволяє

доставляти контент швидше шляхом зберігання копій контенту на серверах, розташованих ближче до користувачів.

SSL – Secure Sockets Layer – це криптографічний протокол, що забезпечує захищену передачу даних між клієнтом та сервером в Інтернеті.

PWA – Progressive Web Application – це вебзастосунок, який може працювати на будь-якому пристрої і надає користувачам досвід, який нагадує нативні мобільні застосунки. Він поєднує в собі переваги вебсайтів та мобільних застосунків, забезпечуючи високу продуктивність, швидкість роботи, можливість встановлення на головний екран, офлайн функціональність та доступ до апаратних функцій пристрою, таких як камера, геолокація та інші.

SPA – Single Pape Application – це тип вебзастосунку, який відрізняється від традиційного підходу, де кожен запит до сервера завантажує нову сторінку. У SPA весь контент завантажується один раз при запуску застосунку, і подальша навігація відбувається без перезавантаження сторінки.

Фреймворк – це програмна платформа, яка надає загальні функціональні можливості та може бути додатково налаштована за допомогою коду, написаного користувачем. Він забезпечує стандартний спосіб створення та розгортання застосунків і є універсальним програмним середовищем для багаторазового використання.

ООП – об’єктно-орієнтоване програмування – це підхід до програмування, який використовує об'єкти як основну одиницю роботи з даними і функціями.

SOLID – це абревіатура, що означає п'ять основних принципів об'єктно-орієнтованого програмування. Ці принципи допомагають покращити якість коду, роблять його більш зрозумілим, розширюваним та підтримуваним.

ВСТУП

В останні роки розвиток технологій та інтернету надав можливість розширити межі медичної допомоги та створити новий формат – телемедицину. Телемедицина – це забезпечення медичної допомоги та діагностики на відстані за допомогою засобів телекомунікацій та інформаційних технологій. Зараз цей вид медицини активно розвивається із залученням інформаційних технологій, що дозволяє більш ефективно надавати медичну допомогу, особливо в умовах карантинних обмежень та віддалених місцевостях.

Одним із головних інструментів телемедицини є вебплатформи. Вони дають можливість швидко та зручно звернутися до медичного фахівця онлайн, пройти консультацію, отримати рекомендації та відповіді на питання. Вебплатформи для телемедицини повинні відповідати вимогам безпеки та забезпечувати надійність зберігання та передачі медичної інформації. Для цього використовуються різні технології, такі як SSL, JWT, PWA, SPA, та інші.

Таким чином, розробка вебзастосунка для телемедицини є важливим етапом у розвитку цього виду медицини. Правильно розроблений та забезпечений безпекою вебзастосунок може допомогти покращити якість та доступність медичної допомоги, зменшити витрати на організацію та проведення консультацій та діагностики, а також збільшити задоволення пацієнтів від наданої послуги.

У даній дипломній роботі буде розглянуто проєктування та розробка клієнтської частини вебплатформи для телемедицини, що націлений саме на обслуговування пацієнтів клініки, де користувачами будуть як лікарі, які працюють у відповідній клініці, так і пацієнти відповідної клініки. Саме націленість розробки, вирізняє її від аналогів, що присутні на вітчизняному ринку.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Види телемедицини залежно від типу послуг

Телемедицина не стоїть на місці і за час розвитку інформаційних технологій, встигли сформуватися не один підвид послуг, які можуть бути надані застосунком для телемедицини. Кожен з її підвидів мають свою функціональну специфіку та не дивлячись на їх різновид, усі вони мають й спільні риси такі як безпека даних, ведення електронних медичних записів та обміну ними між лікарями на відстані та інші. Слід зазначити, що технічні аспекти телемедицини мають дуже важливе значення. Для забезпечення безперебійного та якісного надання медичних послуг на відстані необхідні різноманітні технічні засоби. До таких можна віднести телекомунікаційні мережі, програмне забезпечення для збереження та обробки медичної інформації, відеокамери та інші пристрої для збору та передачі даних про пацієнтів.

1.1.1. *Телемедицина в режимі реального часу*

Даний підвид телемедицини можна описати як процес взаємодії між пацієнтами та провайдерами медичних послуг через програмне забезпечення в реальному часі для виконання різних медичних процедур [1]. Для цього використовуються спеціальні платформи відео- та аудіо телекомунікацій, які відповідають стандартам безпеки та конфіденційності HIPAA. Цей підвид телемедицини може бути використаний для надання первинної та невідкладної медичної допомоги, організації контрольних візитів та призначення ліків.

Конкретні приклади включають консультації з неврології для оцінки симптомів пацієнта, дистанційні консультації з педіатром в прямому ефірі та психічне здоров'я та психіатричну оцінку.

1.1.2. *Телемедицина зберігання та пересилання*

Телемедицина зберігання та пересилання, також відома як асинхронна телемедицина, є протилежністю телемедицини в реальному часі. Цей вид телемедицини дозволяє збирати дані про стан здоров'я та інші медичні дані пацієнтів, а потім безпечно передавати їх на захищену платформу-хмару, до якої інші авторизовані користувачі можуть отримати доступ у будь-який зручний для них час [2]. Телемедицина зберігання та пересилання є ефективним засобом співпраці між пацієнтами та спеціалістами первинної медичної допомоги, оскільки всі вони можуть переглядати інформацію, коли їм це зручно.

Для забезпечення конфіденційності пацієнтів використовуються рішення з вбудованими передовими функціями безпеки. Телемедицина зберігання та пересилання особливо корисна в дерматології, офтальмології та радіології для діагностики та лікування.

Приклади використання асинхронної телемедицини включають рентгенологічні знімки або МРТ, які перевірені дистанційними лікарями, цифрові фотографії захворювань шкіри, які переглядаються та лікуються дистанційно в дерматології, дистанційне обстеження при патології, дистанційне обстеження очей на діабетичну ретинопатію в офтальмології та портали для пацієнтів, що дозволяють спілкування між пацієнтом та його сімейним лікарем.

1.1.3. *Дистанційний моніторинг пацієнта*

Віддалений моніторинг пацієнта (RPM) є методом дистанційного моніторингу здоров'я та клінічних ознак пацієнта за допомогою різних технологічних пристроїв. Він зазвичай застосовується для лікування пацієнтів, які перебувають у високому ризику або мають хронічні захворювання, таких як цукровий діабет або гіпертонія. RPM є особливо важливим під час пандемії, оскільки він дозволяє хворим і пацієнтам, які

одужують, залишатися вдома замість того, щоб перебувати в лікарні чи клініці.

Наприклад, дистанційне лікування хронічних захворювань, імплантовані або керовані пацієнтом пристрої дистанційного моніторингу серцевої недостатності, трекери глюкози для хворих на цукровий діабет, дистанційний моніторинг для довготривалої та пост-гострої допомоги та дистанційний моніторинг на основі програм координації догляду є прикладами технологічних пристроїв, які використовуються для RPM.

1.1.4. *mHealth*

mHealth – це галузь телемедицини, що забезпечує медичні та клінічні послуги за допомогою мобільних пристроїв, таких як смартфони та планшети, а також бездротових медичних пристроїв [3]. Основна мета mHealth – забезпечити взаємодію пацієнтів та медичних працівників за допомогою новітніх технологій, які дозволяють ефективно відслідковувати стан здоров'я пацієнта та надавати своєчасну медичну допомогу.

Приклади застосування mHealth:

- Використання мобільних застосунків та індивідуальних медичних пристроїв для вимірювання пульсу, тиску, рівня глюкози в крові та інших показників здоров'я пацієнта.
- Використання бездротових пристроїв та сенсорів для моніторингу стану хворого та передачі цих даних до медичних працівників для аналізу та взаємодії з пацієнтом.
- Електронна медична консультація через безпечний застосунок для охорони здоров'я або відеозв'язок з медичним працівником для отримання консультації та лікування.
- Використання «розумних» коробок для таблеток, які нагадують пацієнтам про час та дозу ліків, та повідомляють медичним працівникам про те, які ліки були прийняті.

- Використання мобільних застосунків для віддаленого моніторингу пацієнтів з хронічними захворюваннями, що дозволяє медичним працівникам відслідковувати стан здоров'я пацієнта та надавати своєчасну допомогу.

1.2. Аналіз існуючих програмних рішень

У рамках дипломного проєкту було проведено пошук та аналіз наявних програмних рішень в галузі телемедицини на українському ринку з метою підтвердження цінності та унікальності розробки вебплатформи.

Для проведення аналізу були висунуті основні критерії, такі як: наявність відеоконференцій, чату, запису на прийом до лікаря та відправка аудіо повідомлень. У роботі розглядалися такі системи, як вебзастосунок Medikit, нативні застосунки для Android та iOS Doctor Online та Likar Online. Аналіз особливостей кожної з них дозволить покращити програмну розробку вебплатформи в рамках дипломного проєкту.

1.2.1. *Medikit*

Medikit – це український застосунок для телемедицини, який надає можливість звернутися до лікаря безпосередньо з дому за допомогою смартфону або комп'ютера через браузер [4].

Основні функції платформи Medikit:

1. Запис на прийом до лікаря – вебзастосунок Medikit надає можливість користувачам записатися на прийом до лікарів, що надають свої послуги на платформі, в зручний для них час.
2. Електронна медична картка – застосунок надає можливість вести електронну медичну картку, де можна зберігати інформацію про своє здоров'я та медичні записи.
3. Відеоконференції – платформа надає можливість проводити відеоконференції, що дозволяє лікарю отримати додаткову інформацію для постановки діагнозу.

4. Чат – дозволяє користувачам звертатися до лікарів та отримувати консультації через чат-бота, а також обговорювати питання зі своїми лікарями, які були призначені їм для консультацій або лікування.
5. Електронна медична картка – застосунок надає можливість вести електронну медичну картку, де можна зберігати інформацію про своє здоров'я та медичні записи.

З усіх застосунків, що були проаналізовані в ході написання дипломної роботи, Medikit є найпотужнішою платформою і надає функціональність сучасного вебзастосунку для телемедицини.

1.2.2. *Doctor Online*

Doctor Online – це мобільний застосунок для телемедицини, який надає можливість отримати консультацію лікаря та багато іншого. Даний застосунок надає ті ж основні можливості, що й наведений вище вебзастосунок [5].

Окрім того, так як це все ж таки нативний застосунок для смартфонів, то він може отримувати дані з пристрою користувача, наприклад, з розумних годинників про кількість кроків, дані про пульс та інші дані, що можуть бути агреговані подібними пристоями.

1.2.3. *Likar Online*

Likar Online це також нативний мобільний застосунок, як і наведений вище, він доступний для смарфтонів з операційною системою Android та iOS. Цей застосунок, надає такі ж послуги, що й Doctor Online, але Doctor Online дозволяє отримувати консультації з більш широкого спектру спеціалізацій, включаючи загальну медицину, кардіологію, психіатрію, дерматологію, неврологію та інші.

1.3. Результати аналізу

Як можна підмітити, розглянуті рішення мають дуже багато спільного, і кожен з них надає функціональність декількох з видів телемедицини, наведених у розділі 1.1. Це вказує на те, що сьогодні, для користувачів важлива не тільки консультація лікаря в реальному часі, а й інша функціональність застосунків для телемедицини.

Однак, очевидно, що якщо існував би один застосунок, в якому було б все необхідне для користувачів, то не було б стільки аналогів. Тому пропонується порівняти наведені у розділі 1.2 застосунки за наступними критеріями:

1. *Платформа*. Система розроблена під веббраузери, iOS чи Android.
2. *Чат.* Система надає можливість спілкування з лікарем в режимі реального часу. Сюди також входить можливість відправляти та отримувати файли.
3. *Відеоконференції.* Система надає можливість спілкування з лікарем в режимі реального часу, використовуючи технології відеоконференцій.
4. *Відстеження показників здоров'я.* Мається на увазі, можливість інтеграції з пристроями, такими як розумні годинники і збір даних про здоров'я з них.
5. *Медична картка користувача.* Система надає можливість як користувачу так і лікарю вести медичну карту. В ній може бути зазначена наступна інформація:
    - Інформація про пацієнта (ПІБ, рік народження, зріст, вага, номер телефону).
    - Виписані рецепти.
    - Висновки лікарів.
    - Алергічні реакції.
    - Хронічні хвороби.

6. *Сповіщення та нагадування.* Система надає можливість отримувати сповіщення та нагадування як на пристрій (нативні застосунки для iOS, Android) так і на пошту (нативні застосунки для iOS, Android, вебзастосунок).

Нижче наведено табл. 1.1 з результатами порівняння програмних рішень з телемедицини за вказаними критеріями.

Таблиця 1.1

Порівняльна характеристика особливостей існуючих рішень

Критерій	Аналоги		
	Medikit	Likar Online	Doctor Online
iOS платформа	–	+	+
Android платформа	–	+	+
WEB	+	–	–
Чат	+	+	+
Відеоконференції	+	+	+
Відстеження показників здоров’я	–	–	+
Медична картка користувача	+	+	+
Сповіщення та нагадування	–	+	+

З табл. 1.1 можна дійти висновків, що усі застосунки майже відповідають усім критеріям. Найбільш вагома різниця, це платформа, на якій доступний той чи інший розглянутий застосунок.

Medikit – застосунок лише для браузерів, тому в нього немає функціональності відстеження показників здоров’я за допомогою різних

пристроїв. *Likar Online* – нативний застосунок для Android та iOS пристоїв. *Doctor Online* – нативний застосунок, що може бути використаний як на пристроях з операційною системою iOS так і з Android.

Після аналізування наявних рішень на ринку застосунків телемедицини в Україні можна стверджувати, що жодне з них не відповідає повністю вимогам, поставленим до дипломного проєкту. Отже, було вирішено розробити клієнтську частину вебплатформи, що задовольнятиме вимоги та враховуватиме переваги та недоліки кожної з систем, описаних раніше.

2. АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКІВ

2.1. Обґрунтування вибору платформи застосунку

У сьогоденні існує багато платформ, на яких може бути використаний застосунок, і типів програмного забезпечення. Кожен тип платформи має свої індивідуальні особливості: продуктивність, розміри та інше.

Лідерами з платформ сьогодні є: персональні комп'ютери (та ноутбуки), смартфони, браузери. Для кожної платформи існують відомі підходи до розробки програмного забезпечення. Пропонується розглянути найпопулярніші платформи та типи програмного забезпечення, їх переваги та недоліки, що можуть бути використані на них.

2.1.1. *Смартфони*

Найпоширенішим видом застосунків для смартфонів є нативний застосунок.

Нативний застосунок в контексті смартфонів – це програмне забезпечення, розроблене спеціально для операційної системи певного типу мобільного пристрою (наприклад, Android або iOS) з використанням мови програмування, яка підтримується цією операційною системою. Цей тип застосунків запускається безпосередньо на пристрої, тому вони мають доступ до всіх можливостей пристрою, таких як камера, GPS, контакти, датчики, інтернет-підключення та інші.

Наведемо декілька прикладів переваг та недоліків використання смартфонів як платформу:

1. Висока продуктивність: нативний застосунок розроблений спеціально для платформи, що дозволяє отримати максимальну продуктивність.
2. Нативний доступ до апаратних ресурсів: такі як камера, геолокація, гіроскоп та інші.

3. Нативна інтеграція з операційною системою: застосунок повністю інтегрований з операційною системою пристрою, що дозволяє використовувати всі можливості платформи.
4. Краща безпека: нативні застосунки мають вищий рівень захисту від хакерських атак та вірусів.
5. Сповіщення.

Недоліки:

1. Високі витрати на розробку: розробка нативних застосунків зазвичай потребує більше часу та коштів порівняно з іншими видами застосунків.
2. Складна розробка для кількох платформ: якщо потрібно розробити застосунок для кількох платформ (iOS та Android), це може потребувати розробки двох окремих застосунків.
3. Строгі вимоги до оновлень: нативні застосункм можуть вимагати оновлення при зміні версії операційної системи, що може викликати проблеми для користувачів.
4. Складність тестування: тестування нативних застосунків може бути складнішим порівняно з іншими видами застосунків.

2.1.2. *Настільні комп'ютери та ноутбуки*

Для цих платформ лідирує нативний застосунок. Наведемо його переваги та недоліки. Взагалі, переваги нативних застосунків для комп'ютерів та смартфонів багато де є спільними.

Переваги:

1. Швидкість та продуктивність: загалом, спільна перевага для нативних застосунків (для смартфонів та ноутбуків).
2. Нативна інтеграція з операційною системою: ще одна спільна перевага з нативними застосунками для смартфонів. Сюди входить також можливість працювати з локальними файлами та базами

даних, що забезпечує більшу стійкість до відключення від Інтернету.

3. Більш високий рівень безпеки: застосунки для комп’ютерів є менш уразливими до кібератак.

Недоліки:

1. Обмежена мобільність: нативні застосунки для комп’ютерів не можуть бути використані на різних пристроях, таких як смартфони та планшети, і не є такими мобільними, як мобільні застосунки.
2. Потребують встановлення: користувачам потрібно встановлювати застосунок на свої комп'ютери, що може бути затратним часом та складним для менш досвідчених користувачів.
3. Оновлення: оновлення може призвести до некоректної роботи з іншими програмами.
4. Підтримка: підтримка може бути складною через необхідність забезпечувати сумісність з різними операційними системами та їх версіями, а також з різними типами апаратної частини. Це може призвести до складнощів у вирішенні проблем, які виникають у різних конфігураціях.

2.1.3. *Веббраузер*

Очевидно, що основним типом застосунку для веббраузерів є вебзастосунок. Вебзастосунок – це програмне забезпечення, яке працює у веббраузері та доступне через мережу Інтернет. Наведемо деякі переваги та недоліки вебзастосунків.

Переваги:

1. Доступність: користувач може легко отримати доступ до вебзастосунку з будь-якого пристрою, який підключений до Інтернету.

2. Оновлення: вебзастосунки можуть бути оновлені в реальному часі і не потребують встановлення оновлення на кожному пристрої користувача.
3. Вартість: вебзастосунки можуть бути дешевшими у розробці та підтримці порівняно з нативними застосунками.
4. Кросплатформенність: вебзастосунки можуть бути запущені на будь-якому пристрої з веббраузером.

Недоліки:

1. Залежність від Інтернету: без доступу до мережі Інтернет класичні вебзастосунки не можуть працювати.
2. Швидкість: вебзастосунки можуть працювати повільніше порівняно з нативними застосунками через залежність від швидкості Інтернету та процесора.
3. Безпека: вебзастосунки можуть бути менш безпечними порівняно з нативними застосунками, особливо якщо вони не мають належних заходів безпеки.
4. Сумісність з браузерами: при створенні вебзастосунків необхідно враховувати, щоб вони працювали коректно на різних браузерах та їх різних версіях.

2.1.4. *Результати аналізу різних видів застосунку*

Після аналізу переваг та недоліків, для дипломного проєкту було обрано розробку клієнтської частини для веббраузерів як типу програмного забезпечення, оскільки це найбільш відповідне рішення для його призначення.

Так як основною функціональністю застосунку для телемедицини є наступні можливості: чат, відеоконференції, тому постійний доступ до мережі Інтернет є обов’язковою для повноцінного користування, отже це призводить до того, що нативні застосунки (для смартфонів чи комп’ютерів) втрачають деякі переваги над вебзастосунками.

Оскільки застосунок має багато функцій окрім зазначених в цьому розділі, для забезпечення кращого користувацького досвіду, потрібно забезпечити підтримку широких екранів, що не є перевагою смартфонів, але й водночас, критично важливим є мобільність та незалежність від пристрою, за допомогою якого буде відбуватися користування застосунком. Не слід також забувати, що застосунок саме для надання *медичних* консультацій, тому він повинен бути доступний незалежно від платформи, слід мінімізувати ризик помилок на стороні користувачів при оновленні застосунку.

Враховуючи вимоги до розробки, аналіз платформ та типів застосунків, було обрано розробити користувацький інтерфейс саме для веббраузерів, таким чином, застосунок буде класифікований як вебплатформа.

2.2. Вибір технологій для розроблення клієнтської частини вебплатформи

Користувацький інтерфейс взаємодіє з користувачем через клієнтську частину, яка відповідає за відображення змісту та його поведінку. Клієнтська частина вебзастосунку вирішує, як саме контент буде відображатися на вебсайті, буде він статичним чи динамічним. Основними технологіями для розробки клієнтської частини є HTML, CSS та JavaScript [7], які забезпечують відповідну структуру, стилізацію та взаємодію з користувачем.

HTML – HyperText Markup Language – мова розмітки гіпертекстових документів та використовується для створення структури вебсторінок у веброзробці. В основі HTML лежать теги, які визначають різні блоки вебсторінок, такі як параграфи, заголовки, зображення, форми та інші елементи. Крім того, HTML підтримує вставку мультимедійних елементів, таких як аудіо та відео.

CSS – Cascading Style Sheets – мова опису зовнішнього вигляду документа та надає розробникам можливість впливати на зовнішній вигляд

документа, створеного з використанням HTML. За допомогою CSS можна модифікувати розмір, кольори, шрифти, межі, фон та інші властивості елементів сторінки, а також забезпечити її адаптивність до різної роздільної здатності екранів.

Зараз в розробці клієнтської частини вебзастосунків найбільшої популярності набули JavaScript-фреймворки. Вони дозволяють значно спростити процес створення динамічних вебсторінок, забезпечують зручну взаємодію з вебсерверами, роблять застосунок більш інтерактивним та швидкодійним на стороні клієнта. Сьогодні на ринку існує багато JavaScript-фреймворків, серед яких особливо виокремлюються Angular, React та Vue.js.

2.2.1. *Vue.js*

Vue.js – це прогресивний фреймворк розроблений на мові JavaScript для створення вебінтерфейсів та зручно інтегрується в проєкти, які використовують інші JavaScript-бібліотеки [8]. Наведемо переваги та недоліки даного фреймворку.

Переваги:

1. Гнучкість: Vue.js дозволяє розробникам використовувати тільки його шаблонізатор та розширювати HTML-атрибути. Це дає можливість використовувати фреймворк в різних проєктах та налаштовувати його для різних вимог.
2. Продуктивність: Vue.js має дуже швидкий Virtual DOM та ефективно оптимізує рендеринг компонентів, що дозволяє підвищити продуктивність застосунку.
3. Простота в освоєнні: Vue.js має дуже легку та зрозумілу документацію, яка дозволяє швидко освоїти фреймворк.

Недоліки:

1. Обмежена екосистема: Vue.js не має такої великої екосистеми, як React або Angular, тому може бути складно знайти готові бібліотеки та рішення для вирішення певних задач.

2. Слабка підтримка від компаній: на відміну від React, який підтримується Facebook, та Angular, який підтримується Google, Vue.js має меншу корпоративну підтримку. Це може призвести до меншої кількості оновлень, які покращують фреймворк.
3. Менша популярність: Vue.js не є таким популярним, як React або Angular, що може зробити пошук розробників та використання фреймворку у складних проєктах важчим.

2.2.2. *Angular*

Angular – це відкрита платформа для розробки вебзастосунків на мові TypeScript, розроблена компанією Google [9].

Переваги:

1. Комплексний фреймворк, що містить усі необхідні компоненти для розробки вебзастосунків, такі як директиви, модулі, сервіси та інші.
2. Angular має високу продуктивність і швидкодію, що забезпечує користувачам швидкий і зручний досвід використання вебзастосунків.
3. Angular має велику та активну спільноту розробників, яка забезпечує постійну підтримку та розвиток фреймворку.

Недоліки:

1. Великий розмір Angular може вплинути на швидкість завантаження вебзастосунку, особливо на мобільних пристроях.
2. Необхідність використання строгих правил архітектури може зробити процес розробки менш гнучким порівняно з іншими фреймворками.
3. Синтаксис Angular може бути більш об'ємним і складним порівняно з іншими фреймворками.
4. Angular оновлюється досить часто, що може вимагати постійного оновлення залежностей.

5. Навчання Angular може бути складнішим порівняно з іншими фреймворками, особливо для початківців.

2.2.3. *React*

React.js є однією з найпопулярніших бібліотек для створення користувацьких інтерфейсів у веброзробці [10].

Переваги:

1. Швидкість та ефективність: React.js дозволяє створювати швидкі та високоефективні вебзастосунки за рахунок використання віртуального DOM та мінімізації зайвих перерендерів компонентів.
2. Модульність: React.js дозволяє розбити великі проєкти на менші та більш логічні компоненти, що робить код більш зрозумілим та легким у підтримці.
3. Широкий спектр підтримки: React.js має широку спільноту розробників, що дозволяє отримати допомогу та підтримку в розвитку застосунків.
4. Оновлення: розробник забезпечує «кодові модулі», щоб автоматизувати значну частину процесу та спростити міграцію між версіями.

Недоліки:

1. Необхідність використовувати додаткові бібліотеки для певних функцій, що може призвести до збільшення розміру проєкту.
2. Складність роботи зі станом: у React.js робота зі станом може бути складною та заплутаною, що може призвести до невірних результатів та помилок.
3. Використання JSX: використання JSX потребує додаткових інструментів для компіляції до звичайного JavaScript-коду, що може збільшити складність налаштування середовища розробки.

2.3. Огляд препроцесорів CSS

Препроцесори CSS – це інструменти, які дозволяють розширювати базові можливості CSS і працювати з ним більш ефективно та зручно. Використання препроцесорів CSS може спростити процес розробки, зробити код більш структурованим та зрозумілим, а також збільшити його повторне використання [11]. Розглянемо найпоширеніші препроцесори CSS.

2.3.1. *CSS-Crush*

CSS-Crush – це один з багатьох препроцесорів CSS, який дозволяє використовувати більш зручний та економний синтаксис для написання CSS стилів [12].

Переваги:

1. Простота використання: CSS-Crush має легкий та зрозумілий синтаксис, що робить його простим у використанні.
2. Підтримка змінних: CSS-Crush дозволяє використовувати змінні, що полегшує роботу зі стилями.
3. Функції та міксіни: CSS-Crush дозволяє використовувати функції та міксіни, що спрощує написання стилів.
4. Модульність: CSS-Crush дозволяє використовувати модульну структуру для стилів, що забезпечує більшу організованість та зручність у розробці.

Недоліки:

1. Відсутність підтримки динамічних змін змінних.
2. Обмежена функціональність порівняно з іншими препроцесорами CSS, такими як Sass або Less.

2.3.2. *SCSS*

SCSS (Sassy CSS) – це препроцесор CSS, що розширює можливості звичайного CSS. Він дозволяє писати CSS код в більш структурованій та

організованій формі за допомогою введення змінних, міксінів, вкладених правил і багато іншого [13].

Переваги:

1. Надає багато нових функцій та можливостей для CSS, таких як вкладеність, змінні, міксіни, наслідування та багато іншого. Це дозволяє розробникам більш швидко та ефективно створювати CSS-стилі для своїх проєктів.
2. SCSS дозволяє розробникам використовувати вкладеність, що зроблює код більш структурованим та зрозумілим.
3. Змінні в SCSS дозволяють використовувати один і той же значення кольору, розміру або будь-якого іншого параметру на кількох сторінках, що дозволяє значно скоротити час розробки та збільшити його читабельність.

Недоліки:

1. Може бути складно знайти готові бібліотеки SCSS, які відповідають потребам проєкту.
2. Висока складність: SCSS може бути складним для вивчення, особливо для новачків. Є багато різних можливостей, функцій та синтаксису, що можуть бути важкими для засвоєння.
3. Необхідність компіляції: SCSS є препроцесором CSS, тому браузери не підтримують його без відповідного перетворення на звичайний CSS.

2.3.3. *Stylus*

Stylus – це препроцесор CSS, який дозволяє використовувати більш зручний та зрозумілий синтаксис для написання CSS-коду. Замість фігурних дужок та крупок з комою, як у звичайному CSS, Stylus використовує відступи для визначення блоків та команд [14]. Переваги:

1. Менше коду: стилізація може бути написана з меншою кількістю символів завдяки мінімалістичному синтаксису.

2. Простота використання: відсутність фігурних дужок та крапок з комами зроблює код більш зрозумілим та легким для редагування.
3. Підтримка вкладених властивостей: підтримка вкладених властивостей дозволяє більш зручно організовувати код та робити його більш зрозумілим. Динамічні змінні: змінні можуть містити будь-який вираз JavaScript, що дозволяє створювати динамічні стилі.

Недоліки:

1. Складність налаштування: у порівнянні з іншими препроцесорами, налаштування Stylus може бути складнішим, що може відлякувати початківців.
2. Можливі проблеми з відступами: відсутність фігурних дужок та крапок з комами може призвести до проблем з відступами, якщо код не буде правильно організовано.

2.3.4. *LESS*

LESS (Learner CSS) – це препроцесор CSS, який дозволяє розширити можливості звичайного CSS і зменшити кількість коду, що потрібно написати [15]. LESS пропонує додаткові функції, такі як змінні, вкладені правила, функції, операції над значеннями, міксіни і багато іншого.

Переваги:

1. Менший обсяг коду: LESS дозволяє написати менше коду завдяки використанню змінних, міксів та імпортування стилів.
2. Простіший синтаксис: LESS має більш зрозумілий та простіший синтаксис порівняно зі стандартним CSS.
3. Можливість вкладення: можна вкладати CSS-правила одне в одне, що полегшує організацію стилів.
4. Функції: LESS підтримує використання функцій, які можуть виконувати обчислення, маніпулювати значеннями та генерувати динамічні стилі. Це надає більшу гнучкість та потужність вашим

таблицям стилів, дозволяючи створювати динамічні та адаптивні дизайни.

Недоліки:

1. Швидкість компіляції: зазвичай компіляція LESS-файлів займає більше часу, ніж компіляція файлів інших популярних препроцесорів, а конкретно Sass.
2. Відсутність математичних функцій: у LESS відсутні математичні функції, що можуть зробити складнішими обчислення та розрахунки в стилях.

2.3.5. *Sass*

Sass (Syntactically Awesome Style Sheets) – це препроцесор CSS, який дозволяє розробникам написати CSS більш ефективно і елегантно, використовуючи нові можливості, які недоступні в звичайному CSS [16].

Переваги:

1. Зручний синтаксис: Sass має зручний синтаксис, який дозволяє використовувати змінні, вкладені правила та інші корисні функції, що дозволяє значно зменшити кількість коду.
2. Підтримка змінних: Sass дозволяє використовувати змінні для збереження значень, що дозволяє значно полегшити процес розробки та змінювати стилі швидше.
3. Вкладені правила: Sass дозволяє вкладати один набір правил всередину іншого, що дозволяє збільшити зрозумілість та організацію коду.
4. Умовні конструкції: Sass дозволяє використовувати умовні конструкції, що дозволяє змінювати стилі в залежності від умов.

Недоліки:

1. Великий розмір файлів: Sass-файли можуть бути більшими, ніж звичайний CSS, що може призвести до погіршення продуктивності сайту.

2. Необхідність компіляції: Sass не можна використовувати без попередньої компіляції в звичайний CSS, що може займати додатковий час.

2.4. Огляд технологій клієнтської частини вебзсастосунку для проведення відеодзвінків та чатів

Однією з ключових складових вебплтформи для телемедицини звичайно є відеодзвінки та чат. В цьому контексті можна виділити кілька технологій, які дозволяють створювати клієнтську частину вебплатформи для проведення відеодзвінків та комунікації в реальному часі використовуючи чат.

2.4.1. *WebRTC*

Однією з основних технологій для відеодзвінків в браузері є WebRTC (Web Real-Time Communication). Ця технологія дозволяє реалізувати з'єднання між браузерами користувачів без додаткового програмного забезпечення. WebRTC забезпечує широкі можливості для створення вебзастосунків для відеодзвінків, таких як відображення відео та аудіо, обмін даними між браузерами, збереження історії дзвінків, забезпечення безпеки тощо [17].

Технологія WebRTC включає в себе набір API, які дозволяють налаштовувати мережу і здійснювати з'єднання між браузерами. WebRTC складається з трьох основних компонентів: MediaStream, RTCPeerConnection і RTCDataChannel.

MediaStream дозволяє отримувати потік мультимедійного контенту з камери або мікрофону, обробляти цей потік та передавати його іншим користувачам (рис. 2.1). RTCPeerConnection використовується для з'єднання з іншими користувачами і передачі потоків даних між ними. RTCDataChannel дозволяє передавати будь-які дані, включаючи текстові повідомлення та файли, між браузерами без втручання сервера.

WebRTC є стандартом W3C і підтримується більшістю сучасних браузерів, таких як Chrome, Firefox, Safari та Opera.

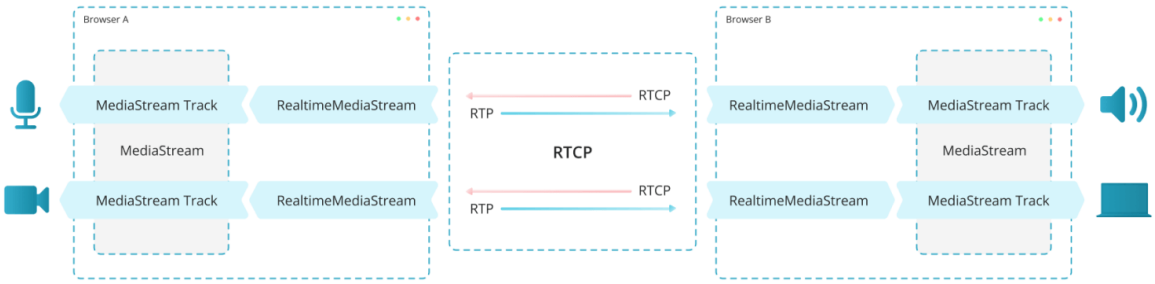


Рис. 2.1. Обробка мультимедійного контенту з камери або мікрофону за допомогою MediaStream інтерфейсу

Пропонується розглянути популярні фреймворки для використання цієї технології.

2.4.2. *SimpleWebRTC*

SimpleWebRTC – це JavaScript фреймворк, який забезпечує просте підключення і використання WebRTC для відео- та аудіодзвінків у вебплатформах. Він дозволяє легко додавати функціональність вебзастосунків для відеозв'язку без необхідності вручну писати велику кількість коду [18].

Переваги:

1. Простота використання: зручний і простий API для взаємодії з WebRTC.
2. Широкі можливості: можливість використання функціональності, такої як збереження чат-історії, зміна параметрів звуку і зображення тощо.
3. Надійність: SimpleWebRTC забезпечує надійну роботу вебзастосунку під час відеодзвінків та аудіодзвінків.
4. Кросбраузерність: підтримується більшість сучасних браузерів.

Недоліки:

1. Обмежена функціональність: SimpleWebRTC надає лише базові функції для реалізації відеодзвінків, тому якщо потрібна розширена функціональність, може знадобитися додаткова розробка.
2. Відсутність підтримки мобільних пристроїв: SimpleWebRTC не підтримує мобільні пристрої, тому якщо планується використання відеодзвінків на мобільних пристроях, може бути потрібно використовувати інші технології.

2.4.3. *PeerJS*

PeerJS – це фреймворк JavaScript, який дозволяє легко і швидко розробляти вебзастосунки, які використовують технологію WebRTC для забезпечення підключення між двома веббраузерами [19].

Переваги:

1. Різноманітність функцій: PeerJS має багату функціональність, включаючи передачу даних, чат, відеозв'язок, обмін файлами тощо.
2. Відкрите програмне забезпечення: PeerJS є відкритим програмним забезпеченням, що дозволяє розробникам змінювати та налаштовувати його відповідно до своїх потреб.
3. Легка інтеграція: PeerJS можна легко інтегрувати з будь-яким фронтенд фреймворком, таким як React, Vue або Angular.

Недоліки:

1. Залежність від сервера: PeerJS потребує центрального сервера для побудови з'єднання між клієнтами, що може створити проблеми з масштабуванням та безпекою.
2. Обмеження браузерів: PeerJS підтримується не всіма браузерами, що може обмежити кількість користувачів, з якими можна підключатися.

3. Проблеми з безпекою: PeerJS може стати джерелом проблем з безпекою, так як він працює без HTTPS-з'єднання, тому використання SSL/TLS є необхідним.
4. Кількість одночасних користувачів: PeerJS не забезпечує можливості створення конференцій, де більше двох користувачів можуть спілкуватися одночасно.

2.4.4. *EasyRTC*

EasyRTC – це JavaScript фреймворк для розробки вебзастосунків реального часу, який забезпечує комунікацію між браузерами з використанням WebRTC [20].

Переваги:

1. Автоматична маршрутизація сигналів: EasyRTC відповідає за автоматичне керування підключеннями, маршрутизацію сигналів і керування потоками даних, що дозволяє розробникам уникнути цього складного процесу.
2. Можливість розширення: EasyRTC має ряд плагінів, які дозволяють розширити його функціональність та додати нові можливості.
3. Підтримка багатьох пристроїв: EasyRTC підтримує роботу з багатьма пристроями, включаючи комп'ютери, смартфони та планшети.

Недоліки:

1. Обмежена підтримка браузерів: EasyRTC не підтримує Internet Explorer та Safari, що може бути недоліком для проєктів, які повинні підтримувати ці браузери.
2. Обмежена функціональність: порівняно з іншими фреймворками, EasyRTC має обмежену функціональність та можливості. Наприклад, він не підтримує групові відеодзвінки з більш як двох учасників.

3. Обмежена документація: EasyRTC не має достатньої документації, тому розробники можуть мати проблеми зі зрозумінням того, як він працює та як його використовувати.

2.4.5. *WebSocket*

WebSocket – це протокол двостороннього обміну повідомленнями між клієнтом та сервером у режимі реального часу [21]. Він забезпечує постійне з'єднання між браузером клієнта та сервером, що дозволяє обмінюватися повідомленнями без необхідності постійно відправляти запити на сервер.

Однією з основних переваг WebSocket є швидкість та ефективність передачі даних, оскільки з'єднання залишається відкритим і не вимагає повторних запитів на сервер для отримання нових даних. Крім того, WebSocket дозволяє побудувати масштабовані та надійні застосунки, оскільки вони можуть витримувати велику кількість одночасних підключень.

Пропонується розглянути переваги і недоліки популярних фреймворків і бібліотек, що використовуються для двостороннього з’єднання у режимі реального час між клієнтом та сервером.

2.4.6. *SignalR*

SignalR – це бібліотека для розробки вебзастосунків, що дозволяє розробникам зручніше працювати з WebSocket протоколом [22].

Переваги:

1. Підтримка багатьох типів з'єднань: SignalR підтримує різні типи з'єднань, такі як WebSockets, Server-Sent Events (SSE), Long Polling тощо. Це означає, що може використовуватися SignalR незалежно від того, який браузер або пристрій використовується для взаємодії з сервером.

2. Висока продуктивність: SignalR забезпечує швидку та ефективну передачу даних між клієнтом та сервером, що дозволяє зменшити затримки при обміні даними.
3. Багатоплечовий зв'язок: SignalR дозволяє легко встановлювати багатоплечове з'єднання між клієнтами та сервером. Це означає, що ви можете спілкуватися з багатьма клієнтами одночасно та передавати повідомлення в обидва напрямки без необхідності постійного запитування сервера.

Недоліки:

1. Обмеження на бібліотеку: SignalR – це бібліотека, а не повноцінний фреймворк, тому для складних проєктів можуть знадобитися додаткові інструменти.
2. Велика залежність від серверу: SignalR розгортається лише на сервері, тому для його використання потрібно налаштувати та підтримувати серверну інфраструктуру.
3. З попереднього пункту випливає, що SignalR не забезпечує стандартного WebSocket API для роботи з WebSocket на стороні клієнта, що може ускладнити розробку

2.4.7. *SockJs*

SockJS – це JavaScript фреймворк, який дозволяє веббраузерам взаємодіяти з сервером через відкрите з'єднання WebSocket або через імітацію WebSocket з використанням інших транспортних протоколів [23].

Переваги:

1. Підтримка відмовостійкості: якщо один транспортний протокол не може забезпечити передачу даних через мережу, SockJS автоматично переключиться на інший протокол для забезпечення безперебійної роботи системи.
2. Кросбраузерна сумісність: SockJS може працювати з будь-яким браузером, включаючи ті, що не підтримують WebSocket.

3. Активна розробка: SockJS продовжує розвиватись і отримувати підтримку від спільноти.

Недоліки:

1. Нестандартизований протокол: SockJS використовує власний протокол, який не є стандартом.
2. Відсутність підтримки SSL на сервері: SockJS не підтримує SSL на стороні сервера, що може бути проблемою для деяких проєктів, які потребують безпечного з'єднання.
3. Відсутність підтримки більшості серверних мов: SockJS має обмежену підтримку для більшості серверних мов (включно з C#), що може бути проблемою для деяких проєктів.

2.4.8. *Socket.IO*

Socket.IO – це JavaScript фреймворк, який надає можливість створювати багатокористувацькі вебзастосунки, що працюють в режимі реального часу, з використанням технологій WebSockets, Polling та ішних методів комунікації між клієнтом та сервером [24].

Переваги:

1. Розширюваність: бібліотека Socket.IO забезпечує широкий спектр можливостей для розширення функціональності за допомогою плагінів та додаткових модулів.
2. Простота використання: Socket.IO забезпечує зручний API для розробки застосунків в режимі реального часу.
3. Масштабованість: Socket.IO може працювати в режимі кластерів, що дозволяє обробляти багато запитів одночасно.
4. Розширюваність: Socket.IO дозволяє додавати нові функції та взаємодіяти з іншими системами шляхом розширення.
5. Надійність: Socket.IO забезпечує стійкість до втрати пакетів та можливість перевстановлення з'єднання в разі втрати зв'язку.

Недоліки:

1. Надмірна вага: у порівнянні з іншими бібліотеками для розробки застосунків в режимі реального часу, Socket.IO може мати надмірну вагу та складну структуру.
2. Відсутність стандартизації: Socket.IO не є офіційним стандартом, що може створювати проблеми з інтеграцією з іншими системами.
3. Відсутність стандарту: Socket.IO є відкритою бібліотекою, і відсутність стандарту може створювати деякі проблеми зі сумісністю між різними версіями бібліотеки або з іншими бібліотеками для розробки застосунків в режимі реального часу.

2.5. Результати аналізу

Провівши аналіз технологій розробки вебзастосунків, було обрано JavaScript-фреймворк React, тому й мова програмування для розробки, очевидно буде JavaScript. React є ліпше задокументованим ніж Vue.js та вимагає менше часу для налаштування ніж Angular [25].

Замість звичайного CSS, було обрано використовувати препроцесор Sass, так як Sass має більшу функціональність ніж інші розглянуті аналоги [26].

Для проведення відеодзвінків було обрано розглянуту технологію WebRTC: вона доступна для широко розповсюджених браузерів, добре задокументована, надійна в плані безпеки. Для використання цієї технології було обрано SimpleWebRTC, так як цей фреймворк надає необхідну функціональність і підтримується найпопулярнішими браузерами.

Для забезпечення комунікації в реальному часі (відправка текстових повідомлень, файлів тощо) через чат був обраний саме WebSocket протокол, щоб не ускладнювати роботу з WebRTC. Тим паче, що цей протокол використовується для надсилання текстових, аудіо повідомлень та файлів набагато частіше ніж WebRTC, таким чином, у разі виникнення проблем під час розробки, шанс, що спільнота користувачів WebSocket допоможе є вище.

Для використання цього протоколу було обрано JavaScript фреймворк Socket.IO: даний фреймворк повністю задовольняє вимоги і не має таких недоліків з безпекою як SockJs та має підтримку як на стороні клієнта так і на стороні сервера, на відміну від SignalIR.

3. СТРУКТУРНА ОРГАНІЗАЦІЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1. Загальна структура програмного забезпечення

Дипломний проєкт реалізований у вигляді клієнтської частини вебзасотсунку для телемедицини. Сервер та клієнт взаємодіють між собою за допомогою RESTful API. Таким чином, розробка клієнтської частини не залежить від прогресу розробки сервера.

Як архітектуру було обрано Feature-Sliced Design (FSD) [27]. Пропонується детальніше розглянути архітектуру та структуру розробки.

3.2. Архітектура дипломного проєкту

Feature-Sliced Design – це сучасна архітектура клієнтської частини вебзастосунків, яка дозволяє організувати розроблення проєкту в зручний та ефективний спосіб. Головною ідеєю цієї архітектури є розбиття проєкту на невеликі частини, кожна з яких відповідає за свою функціональність. Завдяки розбиттю проєкту на невеликі блоки, вона дозволяє реалізувати принципи ООП для клієнтської частини вебзастосунків. Це значно підвищує якість коду, так як клієнтська частина може бути розроблена згідно з принципами SOLID та чистого коду, що зменшує кількість дефектів та підвищує загальну якість коду.

Однак, ця архітектура має свої недоліки такі як:

1. Високий рівень входження: для розроблення клієнтської частини вебзастосунків відповідно до даної архітектури необхідно мати більше досвіду та навичок, ніж для розроблення SPA вебзастосунків.
2. Додатковий час на розробку (випливає з першого пункту): використання FSD вимагає більше часу на розробку, оскільки потрібно ретельно досліджувати і проєктувати кожен функціональний блок.

3. Ризик перевищення витрат: якщо в проєкті багато функціональних блоків, може виникнути ризик перевищення витрат на розробку та підтримку.
4. Складність в розгортанні: якщо функціональні блоки вимагають різних технологій та середовищ для розгортання, це може призвести до складнощів у розгортанні та підтримці.

Концептуально, застосунок ділиться на 3 великі частини: Layers (шари), Slices (модулі), Segments (сегменти). Шари складаються з визначеної кількості компонент, а саме 7, однак, не всі з них є обов'язковими. Так само й для сегментів, є визначена мінімальна кількість скільки їх має бути, але не всі є обов'язковими. Розглянемо кожен компонент окремо, для чого він і що в ньому може бути:

1. Layers (шари) – використовується для поділу системи на різні рівні абстракції. Кожен рівень відповідає за конкретні аспекти системи, і є залежним від нижчих рівнів:
    - App – налаштування, стилі, маршрутизатори для всієї програми.
    - Processes – складні міжсторінкові сценарії (наприклад, проведення відеоконференції).
    - Pages – композиційний рівень для створення повних сторінок із об'єктів, функцій (features) і віджетів (widgets).
    - Widgets – композиційний рівень для об'єднання сутностей (entities) і функцій (features) у значущі блоки (наприклад, панель навігації).
    - Features – взаємодія з користувачем, окремий фрагмент функціональності, що виконує певну задачу (наприклад, запис на прийом).
    - Entities – бізнес сутності (наприклад, користувач, лікар).

- Shared – багаторазова функціональність, відірвана від специфіки проєкту (наприклад, бібліотеки, API).

2. Slices (модулі) – використовуються для зберігання логічно пов’язаних модулів близько один до одного, полегшують навігацію кодовою базою. Модулі не можуть використовувати інші модулі на тому самому шарі.
3. Кожен модуль, у свою чергу, складається з сегментів (Segments). Сегменти використовуються для допомоги розділення коду у модулі за його технічним призначенням.

Продемонструємо наглядніше, на прикладі функціональності підтвердження запису прийому адміністратором (рис. 3.1), що являє собою реалізація даної архітектури.

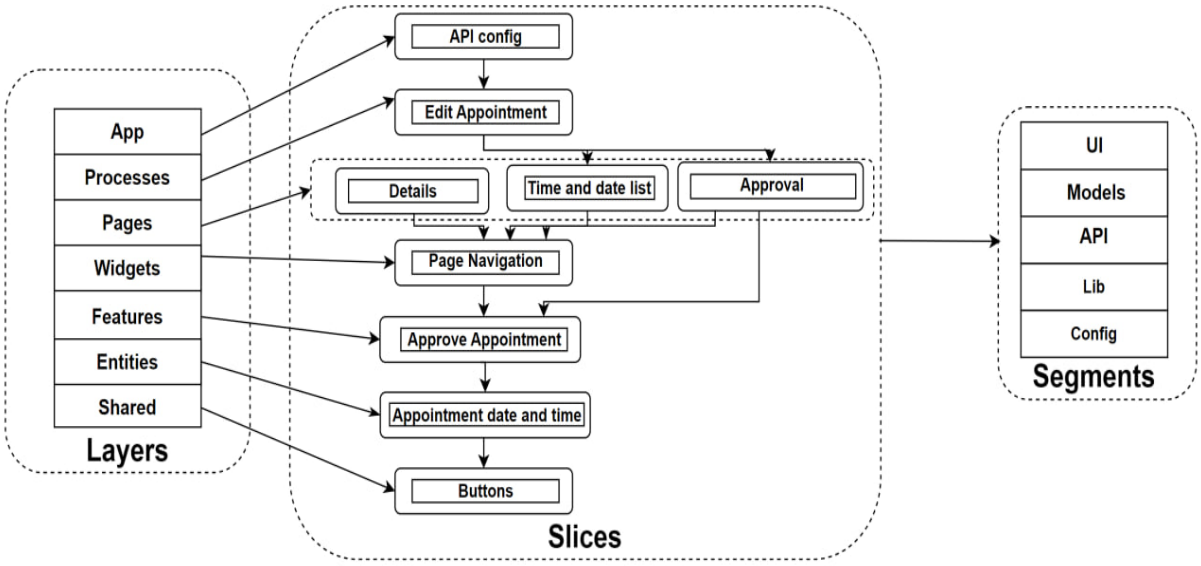


Рис. 3.1. Ілюстрація FSD архітектури вебплатформи

Як можна побачити, даний архітектурний підхід містить ступеневу жорстку ієрархію, кожен елемент з модулів може використовувати лише елементи з шарів нижче. Слід зазначити, що чим нижче шар, тим небезпечніше в нього вносити якісь зміни, адже чим нижче елемент, тим

менше в нього самостійності і тим більше він може бути використаним в шарах на рівнях вище.

3.3. Модулі проєкту

Розробка має чи малу функціональність, що можна, для спрощення структурування, виділити у окремі програмні модулі.

Для пацієнтів і лікарів, спільними є такі модулі як:

- Редагування інформації про себе.
- Приєднання та відвідування відеоконференцій.
- Чат.
- Керування записами на прийом:
  - Перегляд власних записів.
  - Створення записів відповідно до наявної черги.

Модулі для пацієнтів:

- Пошук лікаря.
- Оцінка якості візиту.
- Перегляд власних виписок та історії хвороби.

Модулі для лікарів:

- Робота з прийомами: редагування, створення та відміна.
- Робота з документами: заповнення, редагування виписок, імпорт та експорт файлів історії хвороби та виписок.
- Перегляд виписок та історії хвороби пацієнта.

Модулі адміністрації системи:

- Робота з лікарями: реєстрація лікарів та перегляд зареєстрованих лікарів, редагування інформації лікарів, активація та деактивація облікових записів лікарів.
- Керування записами на прийом (відміна візиту та редагування дати та часу візитів, що на розгляді).

Пропонується більш детально роздивитися модулі, що відповідають за керування прийомів та проведення відеоконференцій.

3.3.1. *Керування прийомами*

У розроблюваній системі, запис до лікаря має наступні стани (див. Додаток 1. Алгоритм управління записами на прийом):

1. На розгляді.
2. Підтверджено.
3. Скасовано.
4. Завершено.

Стан «На розгляді» запис отримує тоді, коли пацієнт обрав дату та час прийому, однак обрав, опцію «Передзвоніть мені». Як тільки пацієнт зробив запис на прийом з такою опцією, адміністратор системи отримує сповіщення в списку прийомів, що додався новий прийом на розгляд. Адміністратор має право підтвердити візит, редагувати дату та час візиту та скасувати візит.

Стан «Підтверджено» запис може отримати декількома способами:

- Пацієнт записався самостійно і не обирав опцію «Передзвоніть мені».
- Запис підтвердив адміністратор.

Стан «Скасовано», може отримати запис від пацієнта, лікаря та адміністратора. Скасувати візит можна лише до часу та дати настання візиту.

Стан «Завершено» запис відповідно отримує по завершенню прийому.

3.3.2. *Проведення відеоконференцій*

Архітектурно, відеоконференції проходять у форматі P2P (peer-to-peer), тобто у форматі, що означає пряме безпосереднє з'єднання між учасниками відеоконференції, при якому відео та аудіо потоки передаються без посередництва центрального сервера. Даний підхід був обраний, через наступні причини:

- В рамках дипломного проєкту не планується реалізовувати групові відеоконференції та записи відеоконференцій.
- Оскільки дані передаються прямо між учасниками, можна зменшити ризик неправомірного доступу до приватних даних через центральний сервер.
- При прямому обміні даними між учасниками без посередництва сервера передаються безпосередньо, що зменшує затримку.
- У випадку, коли сервер відмовляє, P2P-з'єднання може продовжувати працювати, що забезпечує більшу надійність та стабільність.

Однак, даний підхід має і свої недоліки:

- Відсутність централізованого керування: у P2P-архітектурі складніше керувати контролем доступу, якістю обслуговування та іншими аспектами, які можуть бути централізовано керовані в традиційній моделі.
- Завантаження ресурсів: при прямому обміні даними між учасниками може збільшуватись завантаження ресурсів (трафік, процесор, пам'ять) на їх пристроях.
- Обов’язкове використання STUN серверів [28].
- Обмеження функціональності: використання P2P архітектури значно ускладнює групові відеоконференції адже вся обробка відбувається на пристроях учасників, що вводить обмеження на здатність оброблювати велику кількість відео та аудіо потоків, що отримує користувач одночасно.

Найбільша архітектурна особливість реалізації відеоконференцій у форматі P2P є саме використання STUN серверів, адже без них P2P відеоконференція є неможливою, STUN сервери повинні бути зазначені у конфігураційному файлі для проведення відеоконференції (лістинг 1).

STUN (Session Traversal Utilities for NAT) – визначає зовнішню IP-адресу та порт, через які клієнт підключений до Інтернету. Так як відеоконференція між двома клієнтами можлива лише коли відома зовнішня IP-адреса та порт один одного, то за допомогою STUN серверу, визначаються ці дані шляхом виконання спеціальних запитів і відповідей. Власний STUN сервер не потрібно розгортати, адже деякі компанії, наприклад як Google, безкоштовно надають свої STUN сервери. Дані, що були отримані з STUN серверу використовуються при обмінні ICE (Interactive Connectivity Establishment) кандидатів. ICE – протокол, який використовується в WebRTC для встановлення з'єднання між браузерами у різних мережевих умовах.

ICE кандидати – це мережеві адреси (IP-адреси та порти), які можуть бути використані для встановлення з'єднання. Саме для знаходження у мережі NAT IP-адрес та портів, потрібні STUN сервери.

Лістинг 1. Частина конфігураційного файлу для проведення відеоконференцій

```
function createPeerConnection(lasticecandidate) {
  configuration = {
    iceServers: [{
      urls: "stun:stun.stunprotocol.org"}]};
     try {
                peerConnection = new
RTCPeerConnection(configuration);
              } catch(err) {
                chatlog('error: ' + err);
              }
  peerConnection.onicecandidate =
handleicecandidate(lasticecandidate);
  peerConnection.onconnectionstatechange =
handleconnectionstatechange;
  peerConnection.oniceconnectionstatechange =
handleiceconnectionstatechange;
  return peerConnection;
}

function handleicecandidate(lasticecandidate) {
  return function(event) {
```

```
        if (event.candidate != null) {
          console.log('new ice candidate');
        } else {
          console.log('all ice candidates');
          lasticecandidate();
        }
      }
    }

    function handleconnectionstatechange(event) {
      console.log('handleconnectionstatechange');
      console.log(event);
    }

    function handleiceconnectionstatechange(event) {
     console.log('ice connection state: ' +
event.target.iceConnectionState);
```

Сам процес початку та проведення відеоконференції має наступні кроки:

- SDP (Session Description Protocol) [29] обміни та відповіді.
- Обмін ICE кандидатами.
- Встановлення P2P з’єднання.
- Обмін відео та аудіо потоками.

Слід зазначити, що усі кроки, окрім P2P з’єднання та обміном відео та аудіо потокам, реалізуються за допомогою сервера (див. Додаток 1. Процес проведення відеоконференції).

4. АНАЛІЗ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості програмної реалізації програмного забезпечення

В рамках дипломного проєкту була реалізована наступна функціональність клієнтської частини вебзастосунку:

- Вхід у застосунок (реєстрація, авторизація та автентифікація).
- CRUD операції над сутностями (лікар, пацієнт).
- Відеодзвінки.
- Чати.
- Робота з файлами.
- Керування прийомами.

Усі компоненти програмного забезпечення мають свої особливості, реєстрація пацієнтів з двофакторною аутентифікацією [30], керування станом лікаря (реєстрація лікаря адміністратором системи, активування та деактивування облікового запису лікаря), керування прийомами (запис, підтвердження, редагування часу, скасування) і компоненти, що є головними особливостями клієнтської частини вебплатформи, а саме: відеодзвінки, чати та робота з файлами. Однак, найбільш особливим є процес проведення прийомів у форматі відеоконференції. Пропонується розглянути наведені особливості більш детально.

4.1.1. *Реєстрація нових користувачів (пацієнтів)*

Новий користувач, під час реєстрації, проходить через декілька етапів, кожен з яких має свою сторінку:

1. Збір персональної інформації про користувача (рис. 4.1):
    - Ім’я, прізвище та по батькові (не обов’язково).
    - Дата народження.
    - Номер телефону.
    - Стать.

2. Збір інформації для облікового запису (рис. 4.2):
    - Електронна адреса.
    - Пароль, відповідно до вимог.
3. Підтвердження реєстрації (використовуючи механізми двофакторної аутентифікації) (рис. 4.3)

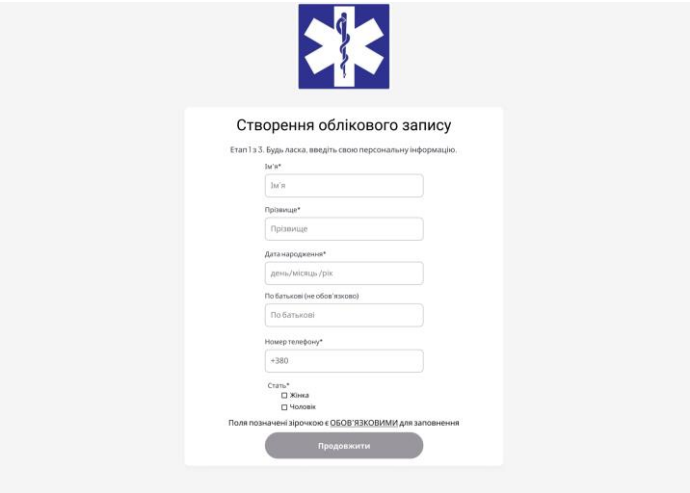


Рис. 4.1. Перший етап реєстрації користувача

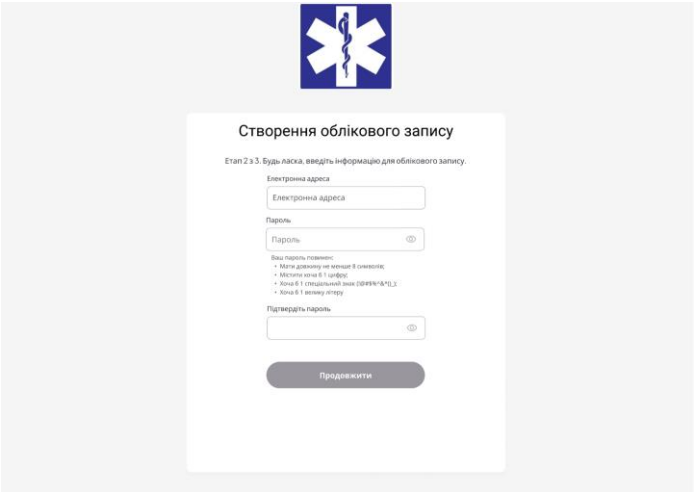


Рис. 4.2. Другий етап реєстрації користувача

Після вводу коду відповідної довжини (6 символів), кнопка «Продовжити» змінює свої стилі (рис. 4.4). Після натискання на неї, домашня сторінка пацієнта відображається (рис. 4.5).

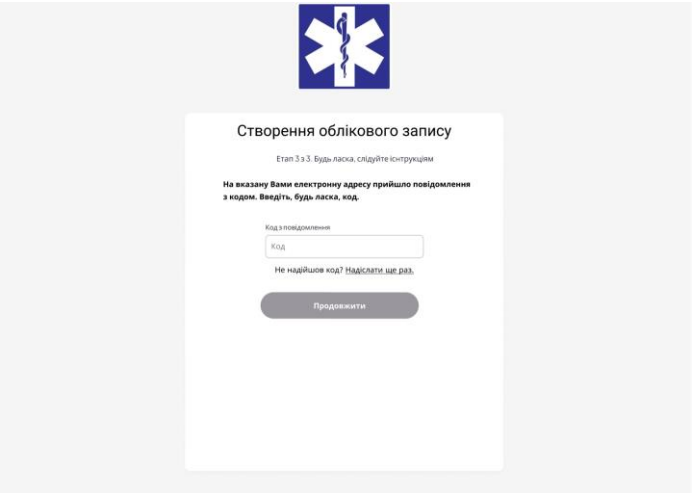


Рис. 4.3. Третій етап реєстрації користувача

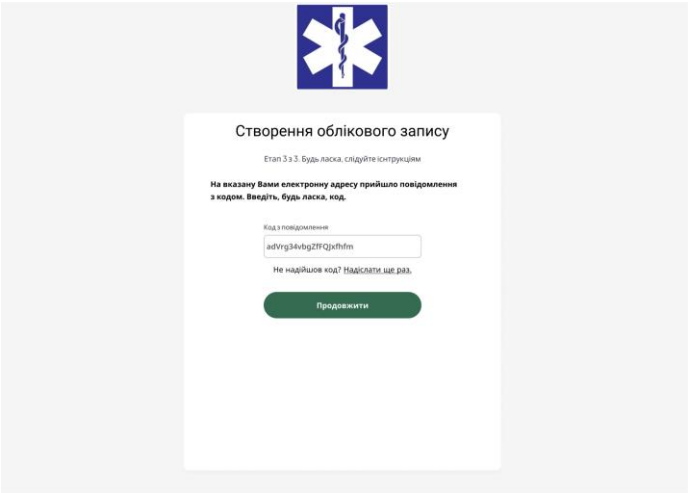


Рис. 4.4. Зміна стилів кнопки «Продовжити» після вводу коду коректної довжини

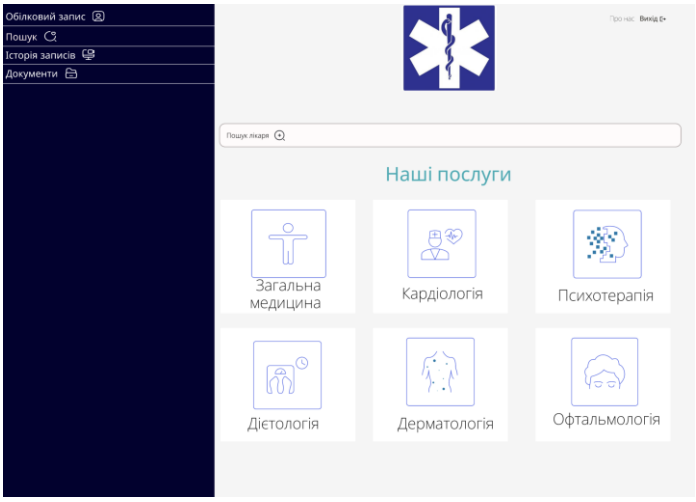


Рис. 4.5. Домашня сторінка користувача

4.1.2. *Проведення прийомів у форматі відеоконференцій*

Для того, щоб під’єднатися до прийому у форматі відеоконференції, необхідно перейти за посиланням у описі прийому. Після переходу, у веббраузері відкривається нова вебсторінка (рис. 4.6) на якій відображається:.

- Налаштування дзвінка:
  - Вимкнути або увімкнути аудіопотік.
  - Вимкнути або увімкнути відеопотік.
  - Завершити дзвінок.
- Відеопотік лікаря (якщо увімкнутий).
- Налаштування дзвінка лікаря:
  - Статус аудіопотоку (вимкнутий чи увімкнутий).
  - Статус відеопотоку (вимкнутий чи увімкнутий).

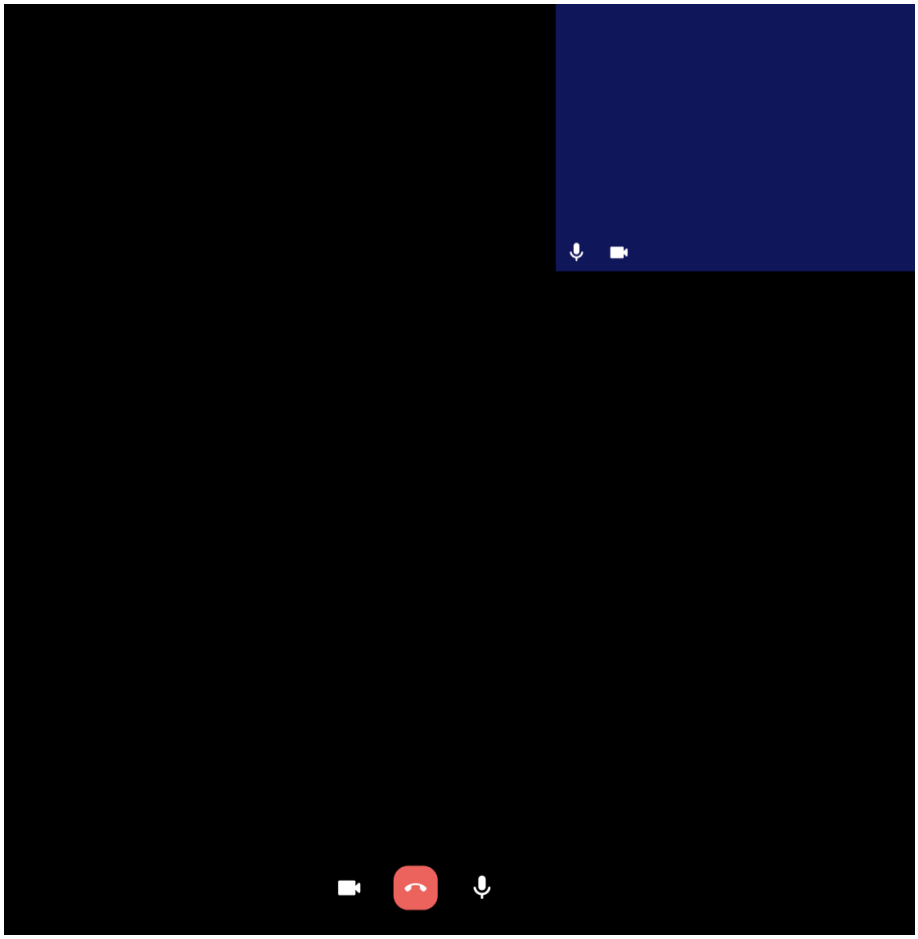


Рис. 4.6. Вікно відеоконференції

З початку відеоконференції, в більшому масштабі відображається відеопотік лікаря, однак, після натискання курсором миші два рази на відображення власного відеопотоку, власний відеопотік (синій квадрат зверху справа на рис. 4.6) міняється місцями з відображенням відеопотоку лікаря.

Після завершення часу відеоконференції, відеоконференція автоматично завершується і відображається нова вебсторінка у веббраузері, де наявне повідомлення про завершення відеоконференції і функціональність оцінку якості обслуговування (рис. 4.7).

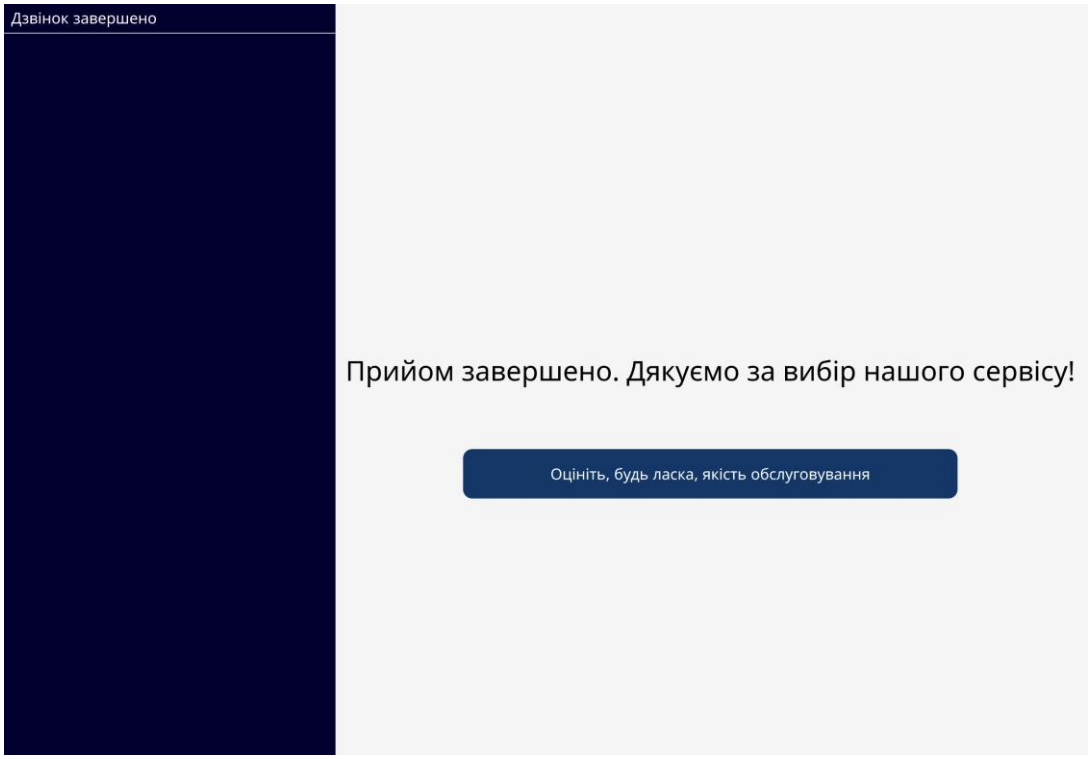


Рис. 4.7. Вебсторінка після завершення відеоконференції

Натиснувши на кнопку «Оцініть, будь ласка, якість обслуговування», відкривається модальне вікно (рис. 4.8), в якому користувачеві пропонується оцінити обслуговування за наступними параметрами:

- Якість відеопотоку під час прийому.
- Загальне враження від візиту

Користувач оцінює візит, за вище зазначеними критеріями, від одного до п’яти балів. Кількість балів, для кращого розуміння, відображається у зірках. Також, користувач може залишити свій коментар у довільній формі.

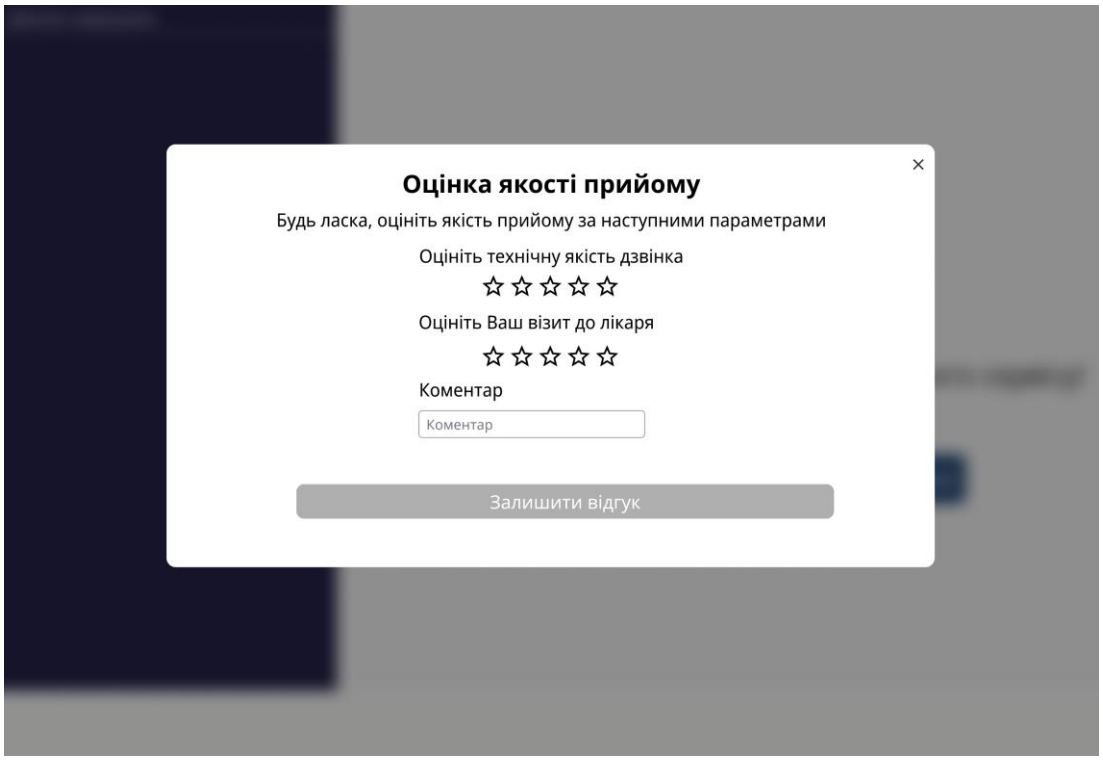


Рис. 4.8. Модальне вікно оцінки якості прийому

Кнопка «Залишити відгук» стає активною лише тоді, коли проставленні обидві оцінки або поле з коментарем не є пустим (рис. 4.9).

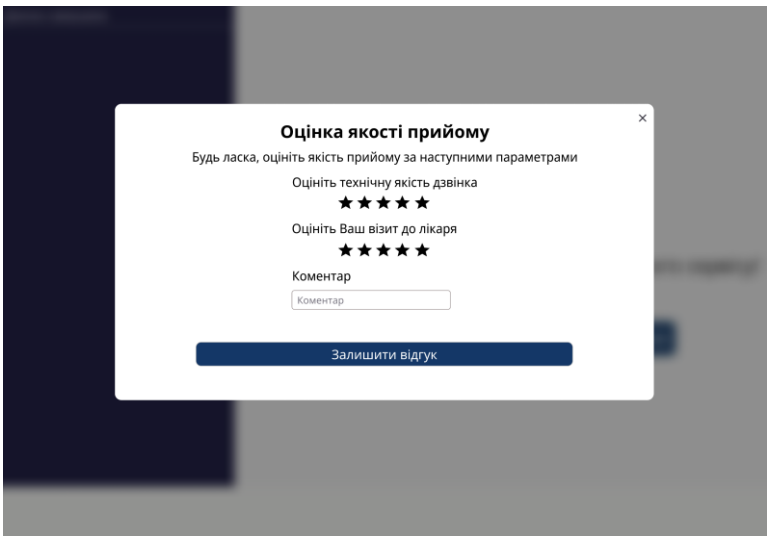


Рис. 4.9. Проставленні оцінки у модальному вікні «Оцінка якості прийому»

4.2. Опис інтерфейсу користувача

Так як розроблене програмне забезпечення реалізовує функціональність клієнтської частини вебплатформи для телемедицини, має три ролі користувачів, де кожна роль має унікальну функціональність, пропонується більш детально розглянути кожну функціональність кожної ролі.

4.2.1. *Авторизація зареєстрованих користувачів*

Вебсторінка авторизації (рис. 4.9) відкривається у веббраузері коли кожен новий користувач, або користувач, у якого вийшов час доступу до застосунку, переходить за URL [31] посиланням вебзастосунку.

Для того щоб увійти у систему, користувачу необхідно ввести електронну адресу та пароль, що були вказані при реєстрації (рис. 4.10). Поле для вводу паролю має доволі типову функціональність: показати чи скрити символи, що вводяться користувачем. Також, у разі відсутності облікового запису, є опція зареєструватись. При натисканні на «Зареєструватись», у веббраузері відкривається вебсторінка реєстрації.

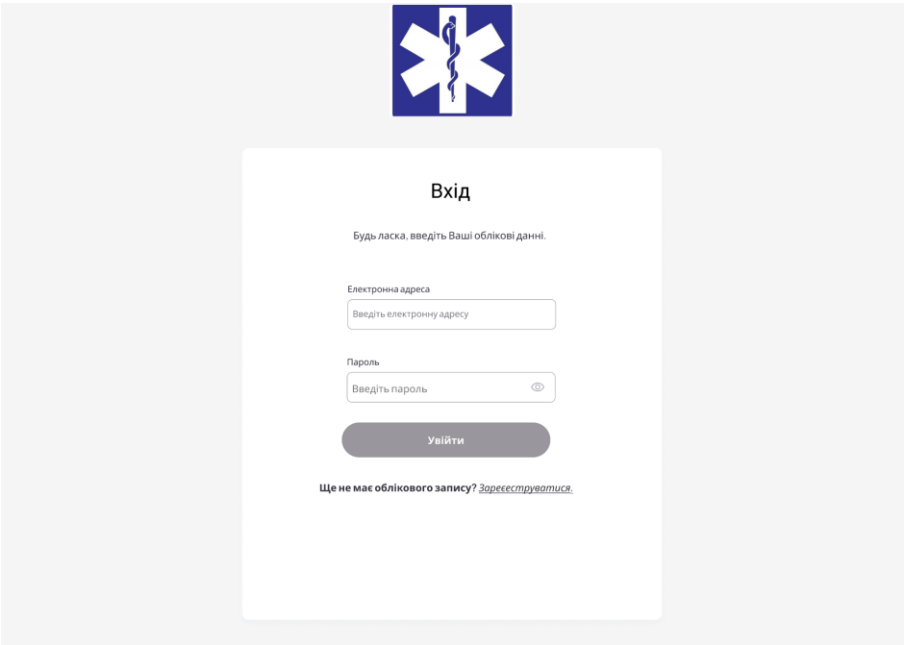


Рис. 4.10. Сторінка авторизації користувачів

4.2.2. *Навігаційна панель користувачів*

Відповідно до доступної функціональності, пацієнти, лікарі та адміністратори мають різну навігаційну панель, однак, вони мають й спільні елементи навігаційної панелі такий як «Обліковий запис» та «Історія записів» (тільки для пацієнтів та лікарів). Сама навігаційна панель знаходиться зліва і займає приблизно третину екрану користувача.

Пацієнт у свої навігаційній панелі має наступні елементи:

- «Обліковий запис».
- «Пошук».
- «Історія записів».
- «Документи».

Лікар у свої навігаційній панелі має наступні елементи:

- «Обліковий запис».
- «Історія записів».

Для адміністратора у навігаційній панелі наявні наступні елементи:

- «Обліковий запис».
- «Записи».
- «Лікарі».

4.2.3. *Обліковий запис пацієнта*

В розділі «Обліковий запис» пацієнт може переглядати та змінювати свої персональні дані, логін та пароль. Також пацієнт має можливість видалити свій обліковий запис.

Натиснувши на «Персональна інформація», у веббраузері відкривається нова вебсторінка, де користувач має змогу переглянути та редагувати надану їм персональну інформацію (рис. 4.11).

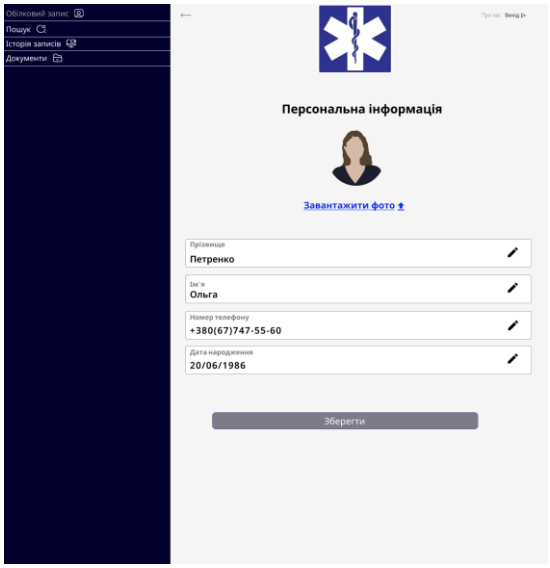


Рис. 4.11. Персональна інформація користувача

Також у розділі «Обліковий запис» є наявні налаштування логіну та паролю (рис. 4.12). В «Логін і пароль» пацієнт може змінити логін, та встановити новий пароль. Для зміни паролю, користувач повинен ввести старий пароль, новий пароль, відповідно до вимог, та повторити новий пароль.

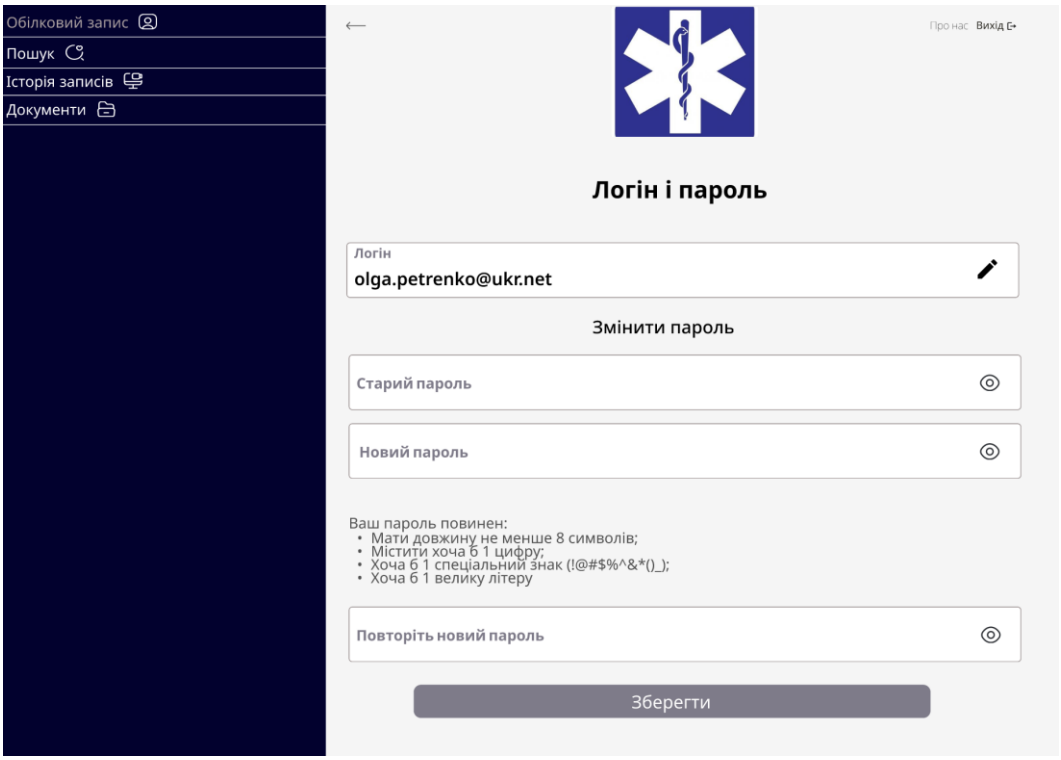


Рис. 4.12. Сторінка «Логін і пароль»

4.2.4. *Пошук лікаря пацієнтом*

Пацієнт може почати пошук лікаря одразу після логіну: на домашній сторінці. Загалом пошук лікаря можливий за наступними параметрами: за спеціалізацією чи за ПІБ. Щоб перейти до пошуку лікаря користувач, знаходячись на домашній сторінці, може:

- Вибрати на домашній сторінці спеціалізацію.
- Натиснути на поле «Пошук лікаря».
- Натиснути на «Пошук» в меню навігації.

Якщо користувач обрав пошук відразу за спеціалізацією, то для нього будуть відображатися лікарі лише зазначеної користувачем спеціалізації. Користувач, також може фільтрувати за категорією та стажем роботи наявних лікарів. У короткій інформації про лікаря, на сторінці пошуку, наявна наступна інформація:

- Фотокартка лікаря.
- ПІБ.
- Спеціалізація.
- Найближчий вільний час (слот).

Якщо користувач на домашній сторінці натиснув на «Пошук лікаря» або на «Пошук», пошук лікарів буде можливий як за спеціалізацією так і за ПІБ лікаря (рис. 4.13).

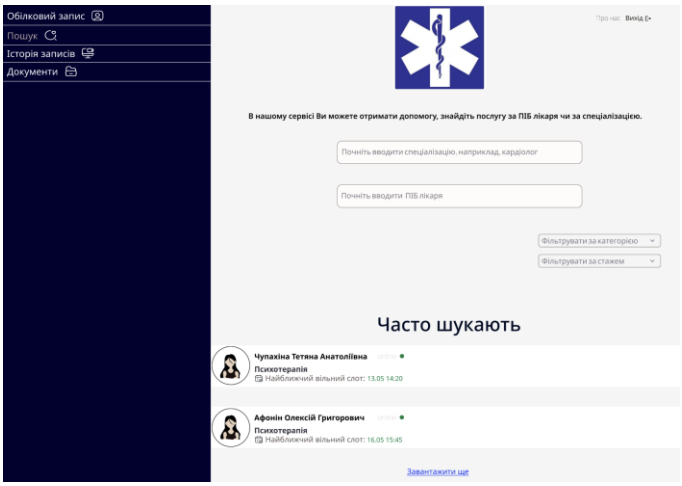


Рис. 4.13. Сторінка пошуку лікарів за спеціалізацією або ПІБ

Пошук як за спеціалізацією так і за ПІБ виконується під час написання пацієнтом значень у відповідні поля, тобто пацієнту не потрібно натискати на будь-які кнопки, щоб здійснити пошук. Однак, пошук можливий *лише* або за ПІБ або за спеціалізацією, поєднувати два параметри не є можливим.

Більш того, як було вказано вище, лікарів можна сортувати за категорією. Фільтри за категорією та стажем роботи можна використовувати одночасно.

Для того щоб перейти на запис на прийом необхідно лише обрати в меню пошуку лікаря, натиснувши на поле з його короткою інформацією.

4.2.5. *Запис та скасування на прийомів*

Як було вказано вище, щоб записатись на прийом, потрібно обрати лікаря в меню пошуку. Запис до обраного лікаря складається з декількох етапів: вибір послуги, вибір часу, підтвердження запису.

Після того, як користувач обрав лікаря, наступна інформація відображається на вебсторінці:

- Фотокартка лікаря.
- ПІБ.
- Спеціалізація.
- найближчий вільний час (слот).
- Послуги, що надає лікар.
- Кнопка «Запис на прийом».
- Розділ «Про лікаря», в цьому розділі наявна наступна інформація:
  - Стаж роботи.
  - Освіта.
  - Категорія.
  - Мови, якими володіє лікар.
  - Порада від лікаря.
  - Девіз.

Після обрання типу послуги, і натисканні на кнопку «Запис на прийом» у веббраузері відкривається нова вебсторінка з вибором часу прийому (рис. 4.14).

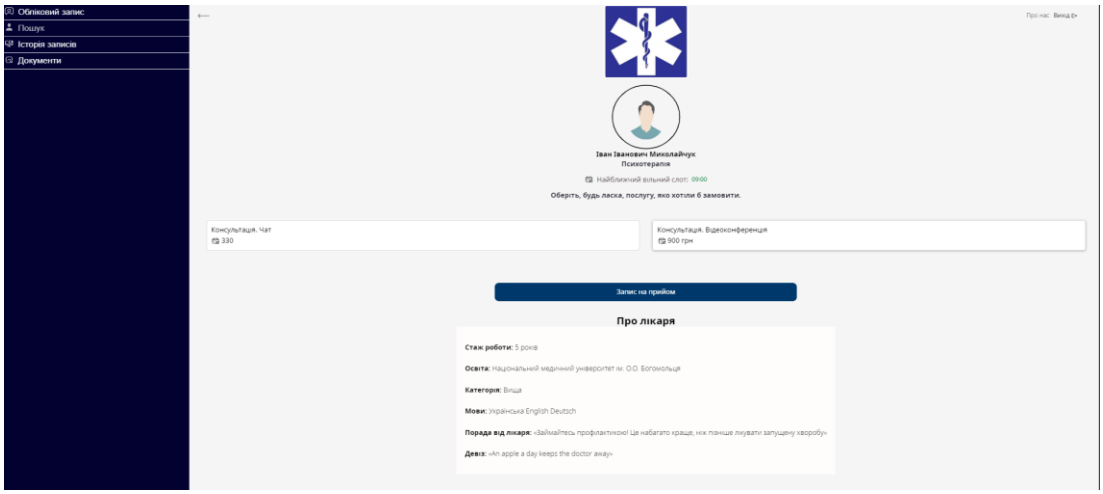


Рис. 4.14. Перший етап запису до лікаря

Другий етап, як було вже зазначено, це вибір часу для запису. Користувач, може обрати вільний час лікаря в робочі дні лікаря (рис. 4.15).

Рис. 4.15. Вибір часу та дати візиту

Після вибору часу та дати візиту, пацієнту слід підтвердити свій запис. На третьому етапі, пацієнт повинен підтвердити, що він згоден з умовами використання сервісу, може коротко описати проблему, стосовно якої потребується в допомозі (не обов’язково). Також, на сторінці відображаються деталі візиту:

- ПІБ та фотокартка користувача.
- ПІБ та фотокартка лікаря.
- Послуга.
- Дата.

Також на сторінці наявна функціональність «Передзвоніть мені», що призначена для зв’язку за межами вебплатформи користувача та адміністратора для уточнення деталей запланованого візиту.

Щоб записатися на послугу, необхідно підтвердити згоду з умовами використання.

Після всіх етапів запису на прийом, пацієнту відображаються деталі запису (такі ж самі як і на останньому етапі запису). Також у користувача є можливість скасувати прийом.

Щоб скасувати візит, пацієнту потрібно натиснути на кнопку «Скасувати». Після натискання на цю кнопку, відкривається модальне вікно «Скасування запису» (рис. 4.16). У відкритому модальному вікні слід вказати причину скасування: вибрати одну з запропонованих або написати власну. Після вказання причини, кнопка «Підтвердити» стає активною і після натискання на неї, запис скасовується.

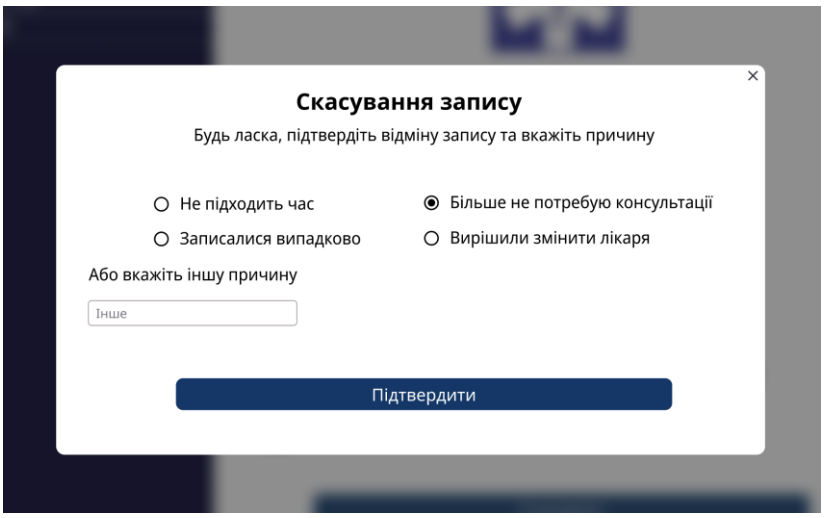


Рис. 4.16. Модальне вікно «Скасування запису» з вказаною причиною скасування

4.2.6. *Історія записів пацієнта*

Щоб перейти до історії записів, пацієнту потрібно натиснути на «Історія записів» в меню навігації.

У цій вкладці користувачу доступна навігація по минулим та запланованим записам. У разі, якщо були візити у минулому, користувачу буде відображений список візитів з датою, назвою послуги та спеціалізацією, за якою проводився візит. Пацієнт може побачити деталі візиту, натиснувши на інформацію про нього.

Список запланованих візитів виглядає так само як і список минулих, однак є деякі відмінності в деталях візиту. Так, для запланованого візиту, що ще не почався, деталі будуть відображені так само як деталі запланованого але будуть містити інформацію про URL посилання на відповідний візит (рис. 4.17).

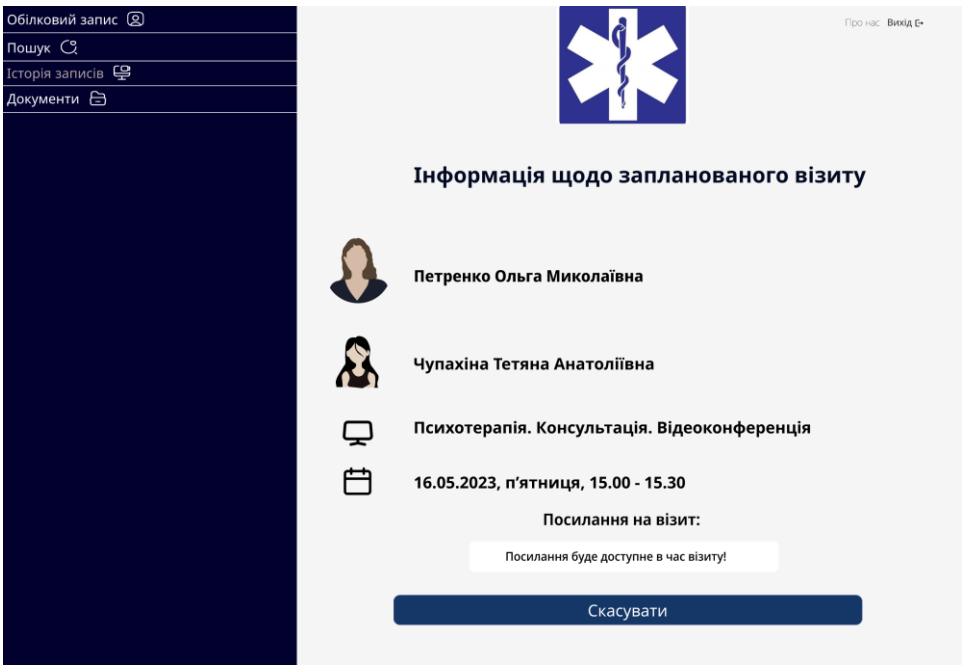


Рис. 4.17. Інформація щодо запланованого візиту, час та дата якого не настали

У випадку, коли час візиту настав, повідомлення «Розпочався», відображається в блоці короткої інформації в списку запланованих записів.

Для візиту, що вже настав інформація в запланованих візитах буде дещо іншою (рис. 4.18), а саме: не буде кнопки «Скасувати» та замість тексту «Посилання буде доступне в час візиту!» буде посилання на візит з можливостю скопіювати посилання, натиснувши на іконку праворуч від поля з посиланням.

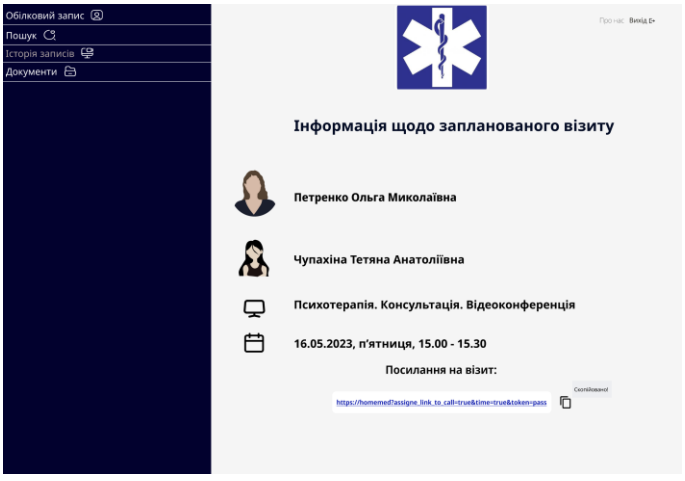


Рис. 4.18. Інформація щодо запланованого візиту, час та дата якого настали

4.2.7. *Документи пацієнта*

В меню навігації, для пацієнта наявна функціональність «Документи». Цей розділ зроблений для того, щоб надати змогу пацієнтам переглядати та завантажувати документи на свій персональний комп’ютер, що були прикріпленні лікарем під час або після візиту.

Натиснувши на «Документи» в меню навігації, у веббраузері відкривається нова вебсторінка сторінка з короткою інформацією про усі минулі прийоми, для яких лікар завантажив документи. Для користувача доступний пошук документів за ПІБ лікаря чи за спеціалізацією лікаря.

У разі якщо прийомів ще не було, відображається повідомлення про відсутність документів.

Натиснувши на коротку інформацію про візит у веббраузері відкриваєтся нова вебсторінка зі змістом документу. Як було вище зазначено, у користувача є можливість завантажити документи, він може це зробити, натиснувши на кнопку «Завантажити».

Коли лікар завантажує нові документи, відповідно до минулого візиту користувача, в меню навігації, у значка документів, відображається кількість документів, що були додані.

4.2.8. *Підключення до прийому в форматі чату*

Користувач під‘єднується до прийому у форматі чату так само як і до прийому форматі відеоконференції: переходячи за отриманим URL посиланням.

Перейшовши за посиланням, окрім активного вікна чату, користувачу відображається кількість часу, що залишився до кінця прийому. Використовуючи чат, пацієнт може відправляти текстові повідомлення та файли (тільки наступних розширень: PDF, DOC, DOCX, JPG, JPEG, PNG, GIF), а також, очевидно, бачити вхідні повідомлення від лікаря. Також, в блоці з текстом повідомлення відображається ким воно було відправлене, дата та час відправлення. Біля самого блоку повідомлення відображається

фотокартка відправника (рис. 4.19). У разі, якщо попередньо вже був прийом використовуючи чат, все листування з попереднього прийому відображається під час поточного прийому.

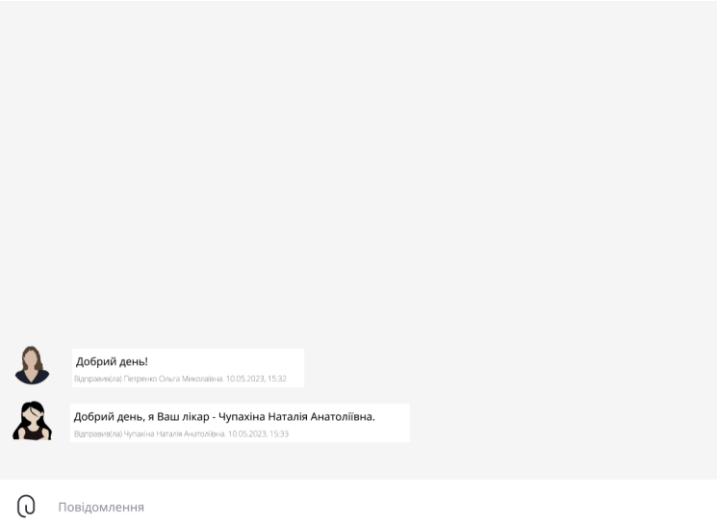


Рис. 4.19. Активний прийом у форматі чату

Після завершення прийому у форматі чату, у веббраузері відображається сторінка з повідомленням про закінчення прийому з кнопкою «Оцініть, будь ласка, якість обслуговування».

Натиснувши на кнопку «Оцініть, будь ласка, якість обслуговування», відкривається модальне вікно де пацієнту пропонується оцінити візит до лікаря та залишити коментар. Логіка в даному модальному вікні схожа з логікою модального вікна оцінки якості прийому у форматі відеоконференції. Користувачу потрібно або оцінити візит від одного до п’яти балів, або залишити коментар.

4.2.9. *Управління прийомами лікарем*

В контексті управління прийомами, лікар має наступну функціональність:

- Перегляд записів (минулих та запланованих).
- Скасування прийому з вказанням причини.

- Завантаження документів до візиту, що вже минув.

Список минулих та запланованих візитів лікар може побачивши перейшовши у «Історія записів» в меню навігації.

Логіка для відображення записів лікаря така сама як з відображенням записів пацієнта: лікар може обрати чи побачити заплановані, чи минулі записи, та, натиснувши на коротку інформацію про візит, перейти до більш детальної інформації.

Натиснувши на коротку інформацію візиту, що ще не відбувся, лікар побачить коротку інформацію про пацієнта (фотокартку, ПІБ, рік народження) та інформацію про візит (вид послуги, дату та час). Також відображається кнопка «Скасувати» та повідомлення, що URL посилання на консультацію буде доступне коли час та дата візиту настане.

Натиснувши на кнопку «Скасувати», відкривається модальне вікно, де лікарю, щоб скасувати візит, необхідно вказати причину та підтвердити скасування.

У разі, коли прийом вже розпочався, лікар, як і пацієнт, бачить повідомлення «Розпочався» в блоці короткої інформації про візит. Також бачить значок «!» у значка історії записів. Лікар не може скасувати прийом, що вже почався, так і не може завантажити документи, поки прийом не завершиться.

Як і для пацієнта, коли дата та час візиту настали, з’являється URL посилання на візит з можливістю скопіювати посилання. (рис. 4.18).

Щоб завантажити будь-який документ (що відповідає допустимому розширенню) лікарю слід перейти до минулих прийомів, знайти в списку прийом, до якого потрібно прикріпити документ, та натиснути на коротку інформацію про візит. Після натискання на коротку інформацію про візит, відображається сторінка з інформацією про візит, що минув, та кнопка «Завантажити документ».

Після натискання на кнопку «Завантажити документ» відкривається типове системне вікно для вибору документу (можна вибрати лише один документ за один раз). Лікар може завантажувати необмежену кількість документів до візиту.

4.2.10. *Перегляд інформації облікового запису лікарем*

Щоб побачити інформацію власного облікового запису, лікарю слід натиснути на «Обліковий запис» в меню навігації. Після натискання даний пункт меню, у веббраузері відкривається нова сторінка з можливістю вибору яку саме інформацію подивитися: персональну чи логін і пароль. Загалом, сторінка така ж саме як для пацієнта, однак у лікаря немає можливості видалити обліковий запис, на відміну від пацієнта.

Натиснувши на «Персональна інформація» відображається інформація про лікаря, що містить наступні поля:

- Фотокартка.
- ПІБ.
- Номер телефону.
- Дата народження.
- Спеціалізація.
- Категорія.
- Освіта.
- Стаж.
- Порада.
- Девіз.
- Мови.

Лікар свою облікову інформацію змінювати не може. Це може робити лише користувач з роллю «Адміністратор». Перейшовши у «Логін і пароль», лікар може лише побачити власний логін і пароль. Як і персональну інформацію, логін і пароль лікар змінювати не може.

4.2.11. *Створення лікаря адміністратором*

Для того, щоб створити обліковий запис лікаря, адміністратору необхідно обрати опцію «Додати» (рис. 4.20) в вкладці «Лікарі» в панелі навігації. Створення облікового запису лікаря складається з трьох етапів:

1. Заповнення необхідної персональної інформації.
2. Заповнення інформації про спеціалізацію.
3. Заповнення логіну та паролю.
4. Перевірка та підтвердження створення облікового запису.

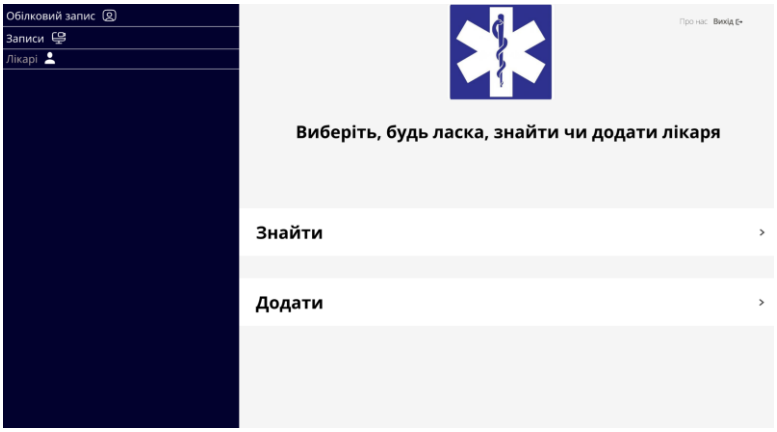


Рис. 4.20. Відкрита адміністратором вкладка «Лікарі»

Після обрання опції «Додати», адміністратору потрібно заповнити персональну інформацію лікаря, тобто завершити перший етап створення облікового запису лікаря. Фотокартку можна завантажити як на першому так і на другому етапі створення облікового запису лікаря.

Як було вище зазначено, після заповнення персональної інформації, адміністратору необхідно вказати інформацію про спеціалізацію лікаря.

Після заповнення усієї необхідної інформації, адміністратору необхідно завершити третій етап створення облікового запису лікаря, а саме ввести логін і пароль.

Після заповнення усієї необхідної інформації, адміністратор має натиснути на кнопку «Перевірити», щоб перевірити правильність усіх введених даних. Під час перевірки адміністратор має зручне меню навігації

між персональною інформацією, інформацією про спеціалізацією та логіном і паролем. Тільки у вкладці «Логін і пароль» адміністратор має можливість створити обліковий запис лікаря.

Після створення облікового запису лікаря, відображається повідомлення про успішне створення облікового запису лікаря.

4.2.12. *Керування обліковим записом лікаря адміністратором*

Під керуванням мається на увазі редагування, активація та деактивація облікового запису лікаря.

Для того щоб здійснювати маніпуляції над обліковим записом лікаря, адміністратору в першу чергу слід знайти обліковий запис лікаря, що йому потрібний. Для того щоб знайти лікаря, необхідно обрати «Знайти» в вкладці «Лікарі». Процес пошуку лікаря адміністратора не відрізняється від пошуку лікаря пацієнтом. Тобто адміністратор може шукати лікаря як за ПІБ та і за спеціалізацією, а також має можливість фільтрувати лікарів за стажем та категорією. Однак, коротка інформація відрізняється від того, що бачить користувач. Адміністратору відображається статус облікового запису лікаря: активовано чи деактивовано.

Після процесу пошуку лікаря, адміністратор має можливість змінювати усю персональну інформацію, інформацію про спеціалізацію, логін і пароль а також має можливість активувати чи деактивувати лікаря, залежно від статусу облікового запису лікаря (активований чи деактивований) (рис. 4.21). *Дані деактивованого облікового запису редагувати неможливо.*

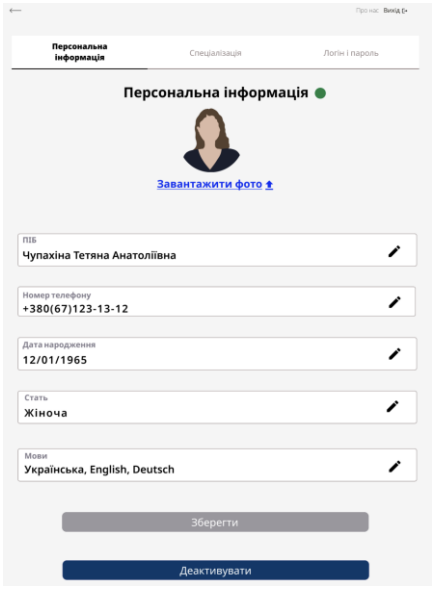


Рис. 4.21. Сторінка редагування персональної інформації облікового запису лікаря

Кнопка «Зберегти» є активною тоді і тільки тоді, якщо хоча б одне значення у будь-якому полі для будь-якої вкладки («Персональна інформація», «Спеціалізація», «Логін і пароль») було змінено.

Для деактивації облікового запису лікаря адміністратору слід натиснути на кнопку «Деактивувати» та підтвердити деактивацію у модальному вікні деактивації облікового запису.

Для активації деактивованого користувача, адміністратору необхідно натиснути на кнопку «Активувати» на сторінці з інформацією про лікаря.

Після того як адміністратор натиснув на кнопку «Активувати», як і з активацією, відкривається модальне вікно з підтвердженням операції. Після підтвердження, відображається повідомлення про успішну активацію облікового запису.

4.2.13. *Керування адміністратором записами на прийом*

Коли пацієнт обирає опцію «Передзвоніть мені», значок «!» відображається на іконці елементу панелі навігації адміністратора «Записи». При переході на вкладку «Записи», адміністратор бачить усі записи, що на розгляді, тобто такі, для яких пацієнти обрали опцію «Передзвоніть мені».

В якості керування записами адміністратор має можливість змінити дату та час прийому, підтвердити та скасувати прийом. Після кліку на прийом зі списку прийомів що на розгляді, у веббраузері відкривається нова вебсторінка з інформацією про прийом, контактною інформацією пацієнта та кнопками «Редагувати», «Підтвердити» та «Скасувати».

Щоб змінити дату та час прийому, адміністратору необхідно натиснути на кнопку «Редагувати». Після натискання на цю кнопку, відображається список вільних дат та часу для лікаря.

Після зміни дати та часу прийому, відображається сторінка з оновленою інформацією про візит, на якій адміністратор має можливість редагувати ще раз чи підтвердити прийом.

Після натискання на кнопку «Підтвердити», відображається повідомлення про успішне редагування прийому на сторінці зі списком записів, що на розгляді. Така ж сама логіка буде, якщо на сторінці з інформацією про візит на розгляді натиснути на кнопку «Підтвердити».

Відповідно, щоб скасувати візит, адміністратору необхідно натиснути на кнопку скасувати. Після натискання на кнопку «Скасувати», відображається повідомлення про успішне скасування прийому на сторінці зі списком записів, що на розгляді.

4.2.14. *Перегляд облікового запису адміністратора*

Натиснувши на «Обліковий запис» на панелі навігації, відкривається сторінка з навігацією між персональною інформацією та логіном і паролем. На сторінці наявна зручна навігація між персональною інформацією та інформацією про логін та пароль. В якості персональної інформації відображається ПІБ адміністратора, а перейшовши на вкладку «Логін і пароль», відображається відповідно логін і пароль адміністратора. Свої дані про обліковий запис адміністратор змінювати не може.

4.3. Рекомендації щодо подальшого вдосконалення

Провівши аналіз розробленого програмного забезпечення можна визначити можливості та перспективи подальшого його розширення та оптимізації.

Так як застосунок зв’язаний з медициною і користуватися ним можуть люди з різноманітними фізіологічними обмеженнями, пріоритетним є додати реалізацію WCAG стандарту [32]. WCAG є найважливішим стандартом в сфері вебдоступності. Він містить рекомендації як зробити контент вебзасосунку більш зручним та доступним для людей з особливими потребами.

Пропонується також додати підтримку проведення групових відеоконференцій та перегляд запису відеоконференції, з можливим завантаженням запису на пристрій користувача. Ця функціональність значно покращить рівень обслуговування клініки, якій був наданий вебзастосунок.

Іншою не менш важливою функціональністю є відправка на сервер як технічної інформації про стан вебзастосунку так і інформації про роботу лікарів. Адміністративна інформація про роботу лікарів буде доступна лише адміністраторам клініки, технічна інформація про стан вебзастосунку буде доступна лише для розробників.

Також, важливим для покращення користувацького досвіду є додавання функціональності «Відновити пароль». За допомогою цієї функціональності, використовуючи вже наявний механізм двофакторної аутентифікації, користувач легко зможе відновити доступ до свого облікового запису у разі втрати паролю.

Разом з тим, не менш важливим є підтримка різної локалізації вебплатформи. Ця функціональність може значно підвищити популярність розробленого продукту, адже графічний інтерфейс вебплатформи буде зрозумілим і для іноземних користувачів.

ВИСНОВКИ

Метою проєкту є створення клієнтської частини вебзастосунку для телемедицини, що поєднує у собі основну функціональність застосунків для телемедицини такі як запис на прийом, прийоми у форматі відеоконференції та чату, завантаження та перегляд документації.

Під час розроблення програмного забезпечення та роботи над продуктом були проаналізовані аналоги на ринку України та виявлено, що існуючі аналоги або доступні лише як мобільні застосунки, мають не достатньо зручний та зрозумілий UI/UX або не задовольняють у повній мірі поставленим вимогам, чим обґрунтовується необхідність розроблення клієнтської частини вебплатформи для телемедицини.

У рамках дипломного проєкту були розглянуті найбільш розповсюдженні технології розроблення клієнтської частини вебзастосунків. З розглянутих технологій були обрані найбільш доцільні інструменти, а саме JavaScript фреймворк React, препроцесор CSS SASS, бібліотеку SocketIO та фреймворк для роботи з відеоконференціями SimpleWebRTC.

Розроблена клієнтська частина надає користувацький інтерфейс для основного функціоналу сучасного вебзастосунку для телемедицини, а саме:

- Реєстрація та авторизація користувачів.
- Пошук спеціалістів та запис на прийом.
- Проведення прийомів у форматі відеоконференції.
- Проведення прийомів у форматі чату.
- Перегляд історії прийомів.
- Перегляд та завантаження документів для пацієнта стосовно прийому.

В списку був перерахований лише основний функціонал, що надає клієнтська частина. Також клієнтська частина забезпечує підтримку

функціональності керування обліковими записами лікарів, керування прийомами, редагування облікового запису.

Реалізована клієнтська частина в поєднанні з серверною частиною дозволить мати вебплатформу для телемедицини з зручним інтерфейсом користувача і функціональністю сучасної вебплатформи для телемедицини.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Hassan, X. Telemedicine [Text] / Hassan Aziz; Hiba Abuchar. // American Society for Clinical Laboratory Science. — 2015. — Vol. 28, № 4. — P. 256-259 — ISSN: 1945-3574.
2. Розробка телемедичних веб- і мобільних додатків: вартість і процес [Електронний ресурс]. — Режим доступу: https://sdh.com.ua/blog/mobile/Rozrobka_Telemedicine_Web/
3. What is mHealth, what does it do & how does it benefit Countries? [Електронний ресурс]. — Режим доступу: https://www.itu.int/en/ITU-D/ICT-Applications/eHEALTH/Be_healthy/Pages/guide-01.aspx
4. ПРО НАС [Електронний ресурс]. — Режим доступу: https://medikit.ua/about-us
5. Doctor Online – Health Assistant [Електронний ресурс]. — Режим доступу: https://doctoronline.care/en/
6. Likar Online [Електронний ресурс]. — Режим доступу: https://likaronline.com.ua/
7. JavaScript [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
8. The Progressive JavaScript Framework [Електронний ресурс]. — Режим доступу: https://vuejs.org/
9. Deliver web apps with confidence [Електронний ресурс]. — Режим доступу: https://angular.io/
10. React. The library for web and native user interfaces [Електронний ресурс]. — Режим доступу: https://react.dev/
11. CSS preprocessor [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor
12. Core Features [Електронний ресурс]. — Режим доступу: https://the-echoplex.net/csscrush/
13. SASS Vs SCSS: What’s The Difference? [Електронний ресурс]. — Режим

доступу: https://www.interviewbit.com/blog/sass-vs-scss/
14. Expressive, dynamic, robust CSS [Електронний ресурс]. — Режим доступу: https://stylus-lang.com/docs/
15. It's CSS, with just a little more. [Електронний ресурс]. — Режим доступу: https://lesscss.org/
16. CSS with superpowers [Електронний ресурс]. — Режим доступу: https://sass-lang.com/
17. WebRTC API [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
18. SimpleWebRTC Documentation [Електронний ресурс]. — Режим доступу: https://docs.simplewebrtc.com/#/
19. The PeerJS library [Електронний ресурс]. — Режим доступу: https://peerjs.com/
20. EasyRTC Framework Tutorial [Електронний ресурс]. — Режим доступу: https://gitee.com/_ok/easyrtc/blob/master/docs/easyrtc_client_tutorial.md
21. The WebSocket API (WebSockets) [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/enUS/docs/Web/API/WebSockets_API
22. Introduction to SignalR [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr
23. SockJS [Електронний ресурс]. — Режим доступу: https://ably.com/periodic-table-of-realtime/sockjs
24. Introduction to SocketIO [Електронний ресурс]. — Режим доступу: https://socket.io/docs/v4/
25. Angular vs React vs Vue: Which Framework to Choose [Електронний ресурс]. — Режим доступу: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#gref
26. Sass, Less, and Others: Which CSS Preprocessor Should You Choose?

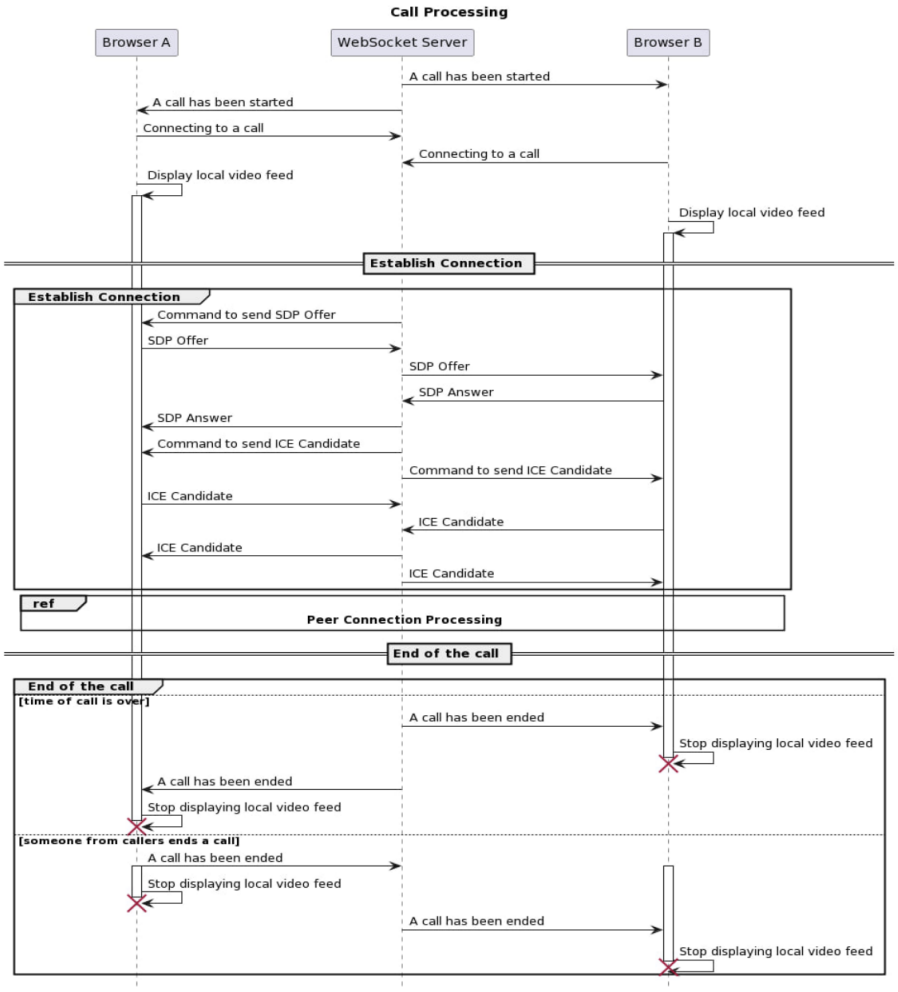
[Електронний ресурс]. — Режим доступу: https://www.bitdegree.org/tutorials/css-preprocessor/

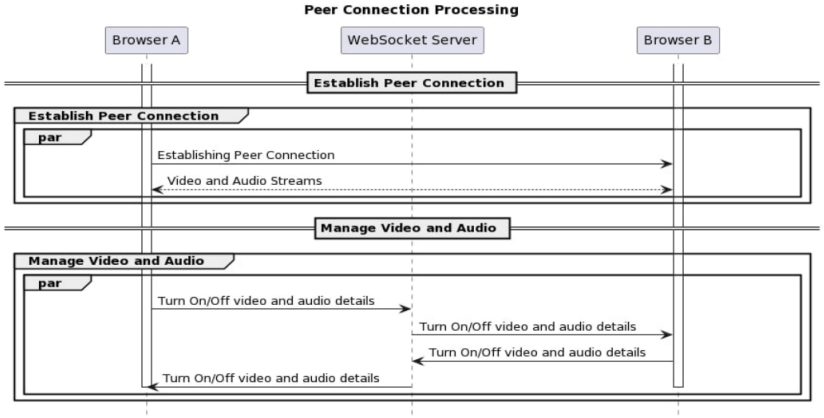
27. Feature Sliced Design Overview [Електронний ресурс]. — Режим доступу: https://feature-sliced.design/docs/get-started/overview
28. Introduction to WebRTC protocols [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols
29. SDP: Session Description Protocol [Електронний ресурс]. — Режим доступу: https://www.ietf.org/rfc/rfc2327.txt
30. What is Two-Factor Authentication (2FA) and How Does It Work? [Електронний ресурс]. — Режим доступу: https://feature-sliced.design/docs/get-started/overview
31. What is a URL (Uniform Resource Locator) [Електронний ресурс]. — Режим доступу: https://www.techtarget.com/searchnetworking/definition/URL
32. W3C Accessibility Guidelines (WCAG) 3.0 [Електронний ресурс]. — Режим доступу: https://www.w3.org/TR/wcag-3.0/

ДОДАТКИ

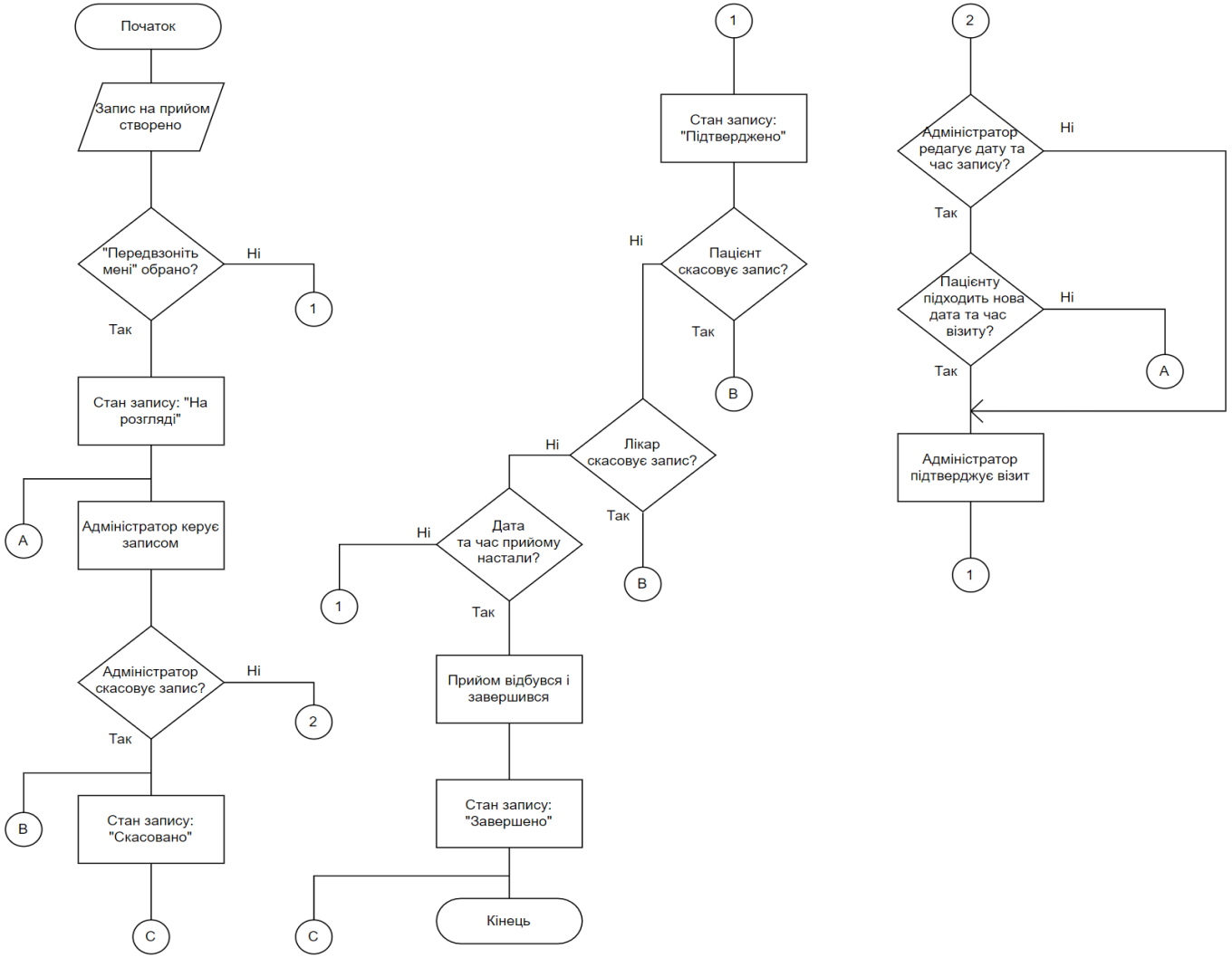
Додаток 1

Копії графічних матеріалів



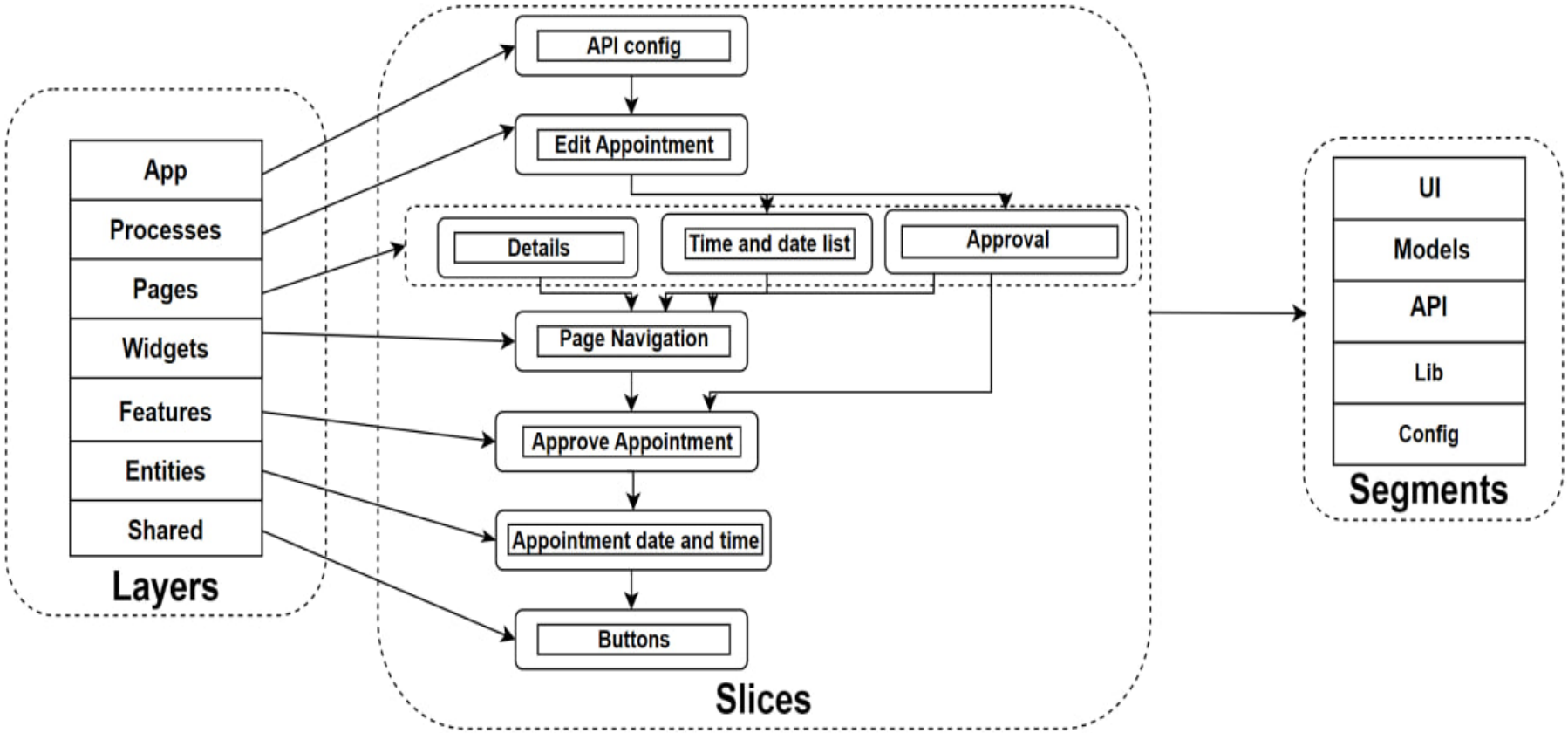


ДП.045440-06-99
Вебплатформа для телемедицини. Клієнтська частина. Процес проведення відеоконференції. UML-діаграма послідовності



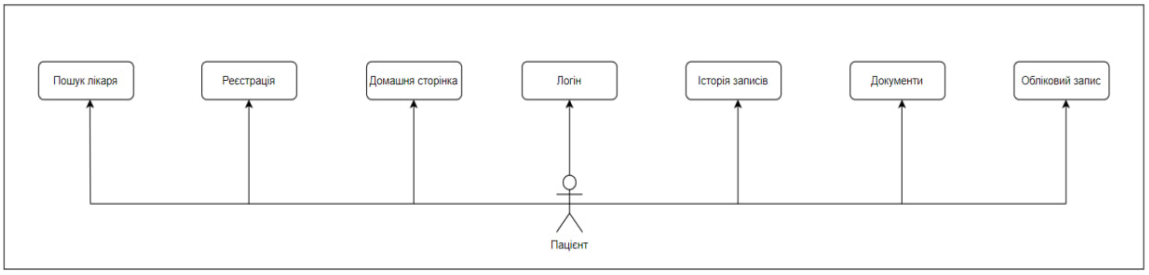
ДП. 045440-07-99
Вебплатформа для телемедицини. Клієнтська частина. Алгоритм управління записами на прийом. Блок-схема алгоритму

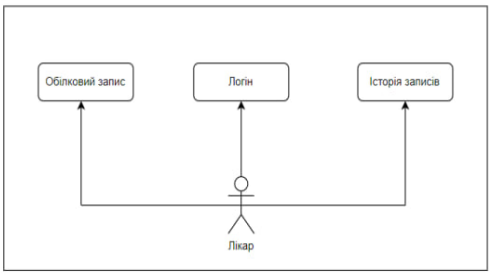
Архітектура модулів клієнтської частин

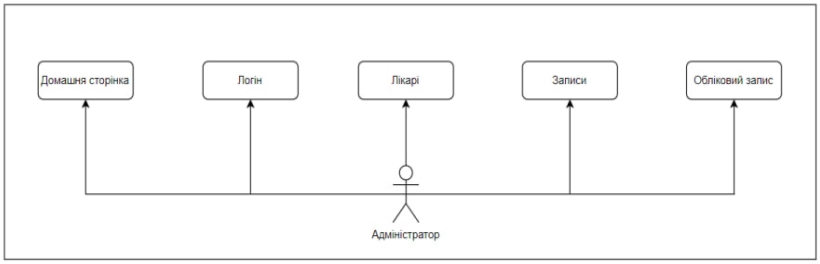


Петренко Нікіта, група КП-93

Структура вебсторінок







Петренко Нікіта, група КП-93

Додаток 2

Лістинг програми

index.js

```
import React from 'react'; import ReactDOM from 'react-dom/client'; import {BrowserRouter} from "react-router-dom"; import {Provider} from "react-redux"; import {App} from "./containers/App"; import {configureStore} from "./state/store"; const root = ReactDOM.createRoot(document.getElementById('root')); const store = configureStore(); root.render( <Provider store={store}> <BrowserRouter> <React.StrictMode> <App/> </React.StrictMode> </BrowserRouter> </Provider> );
```

doctorService.js

```
import server from "./server"; import objectUtil from "../utils/objectUtil"; const SLOTS = [9, 10, 11, 12, 13, 14, 15, 16, 17, 18] export const getDoctorAvailableSlots = (email, date=false) => { const zeroPad = (num, places) => String(num).padStart(places, '0') const year = new Date().getFullYear(); const month = new Date().getMonth() + 1; const day = new Date().getDate(); const hour = new Date().getHours(); const meetings = objectUtil.getDangerousProperty(server.getMeetings(), email); const _meeting = objectUtil.getDangerousProperty(meetings, date ? date : `${zeroPad(day, 2)}/${zeroPad(month, 2)}/${year}`) || []; const meeting = Object.keys(_meeting).map(item => parseInt(item)); return SLOTS.filter(item => (!date ? item > hour : true) && !meeting?.includes(item)); }; export const getDoctorFirstAvailableSlot = (email, date=false) => { const slots = getDoctorAvailableSlots(email, date); if (slots?.length === 0) { const nextDate = !date ? new Date() : new Date(date); nextDate.setDate(nextDate.getDate()+1); return getDoctorAvailableSlots(email, nextDate) } return slots.length > 0 ? slots[0] : false; }; export const slotToString = (slot) => { let a; try { a = parseInt(slot); } catch (e){} return a === 9 ? '09:00' : `${a}:00`; } export const zeroPad = (num, places) => String(num).padStart(places, '0')
```

server.js

```
import storage from "../config/storage";
import notificationService from "./notificationService";
import {dispatch, getState} from "../state/store";
import {setToken} from "../state/ducks/global";
import {selectToken} from "../state/selectors/global";
import {groups} from "../consts/groups";
import objectUtil from "../utils/objectUtil";
const register = (userData) => {
    try {
        delete userData.step;
        delete userData.password_confirmation;
        delete userData.mfa;
        userData.groups=[groups.patient];

        const users =
JSON.parse(localStorage.getItem(storage.users)) || [];
        localStorage.setItem(storage.users,
JSON.stringify([...users, userData]));
        notificationService.showSuccess(`Користувач з ім'ям
${userData?.name} успішно зареєстрований.`)
    } catch (e) {
        notificationService.showError('Не вдалось зареєструвати
користувача, спробуйте будь ласка ще.')
    }
}

const login = (username, password) => {
    // todo: request to backend
    try {
        const users =
JSON.parse(localStorage.getItem(storage.users));
        const userAccount = users?.find(object => object.email
=== username);
        const isLogin = userAccount?.password === password;
        if (!isLogin) {
            // eslint-disable-next-line
            throw 'NotFoundUser'
        }
        dispatch(setToken(userAccount))
    } catch (e) {
        notificationService.showError('Не вдалось увійти.')
    }
}

const deleteAccount = () => {
    // todo: request to backend
    const user = selectToken()(getState());
```

```
        let users = JSON.parse(localStorage.getItem(storage.users))
|| [];

        users = users.filter(item => item?.name !== user?.name)

        localStorage.setItem(storage.users,
JSON.stringify([...users]));
        notificationService.showSuccess(`Акаунт користувача успішно
видалений`);
    }

    const changeInfo = (newUserInfo) => {
        // todo: request to backend
        delete newUserInfo?.password_confirmation;

        try {
            const user = selectToken()(getState());
            let users =
JSON.parse(localStorage.getItem(storage.users)) || [];

            const old_user = users.filter(item => item?.name ===
user?.name)
            users = users.filter(item => item?.name !== user?.name)

            const new_user = {...old_user, ...newUserInfo}
            localStorage.setItem(storage.users,
JSON.stringify([...users, ...[new_user]]));
            notificationService.showSuccess(`Дані користувача
успішно змінено`);
        } catch (e) {
            notificationService.showError(`Невдалось змінити дані
користувача`);
        }
    }

    const setMeeting = (data) => {
        // todo: add validation
        try {
            const meetings =
JSON.parse(localStorage.getItem(storage.meetings)) || [];
            const oldRecords =
objectUtil.getDangerousProperty(objectUtil.getDangerousProperty(meet
ings, data?.doctor) || {}, data?.date) || []
            const newDate = {
                ...meetings,
                [data?.doctor]: {
                    ...(objectUtil.getDangerousProperty(meetings,
data?.doctor) || {}),
                    [data?.date]: {...oldRecords, [data?.time]:
data}
```

```
                }
            }
            localStorage.setItem(storage.meetings,
JSON.stringify(newDate));
            // notificationService.showSuccess(`Запис до
${data?.doctor}, ${data?.date} успішно створений`)
        } catch (e) {
            notificationService.showError('Не вдалось створити
запис')
        }
    }

    const getMeetings = () => {
        try {
            return
JSON.parse(localStorage.getItem(storage.meetings)) || {};
        } catch (e) {}
    }

    const getDoctors = (filter=false, filterName=false) => {
        try {
            const users =
JSON.parse(localStorage.getItem(storage.users)) || [];
            return users.filter(item =>
item.groups.includes(groups.doctor) && (filter ?
item?.specialization?.includes(filter) : true) && (filterName ?
item?.name?.includes(filterName) ||
item?.surname?.includes(filterName) ||
item?.lastname?.includes(filterName): true));
        } catch (e) {}
    }

    const registerDoctor = (data) => {
        try {
            data.groups=[groups.doctor];

            const users =
JSON.parse(localStorage.getItem(storage.users)) || [];
            localStorage.setItem(storage.users,
JSON.stringify([...users, data]));
            notificationService.showSuccess(`Лікар з ім'ям
${data?.name} успішно зареєстрований.`)
        } catch (e) {
            notificationService.showError('Не вдалось зареєструвати
лікаря, спробуйте будь ласка ще.')
        }
    }

    const getMeetingId = (user, doctor, data) => {
        try {
```

```
            const meeting =
JSON.parse(localStorage.getItem(storage.meeting)) || {};
            const findMeeting = Object.keys(meeting).filter(id =>
meeting[id]?.user === user && meeting[id]?.doctor === doctor);

            if (findMeeting.length > 0) {
                return findMeeting[0];
            }
            const id = Object.keys(meeting).length;
            const newDate = {
                ...meeting,
                [id]: {
                    user: user,
                    doctor: doctor,
                    date: data?.date,
                    time: data?.time,
                    type: data?.type,
                    specialization: data?.specialization,
                    chat: []
                }
            }
            localStorage.setItem(storage.meeting,
JSON.stringify(newDate));
            return id;
        } catch (e) {
            notificationService.showError('Не вдалось створити
запис')
        }
    };

    const getMeeting = (id) => {
        try {
            const meeting =
JSON.parse(localStorage.getItem(storage.meeting)) || {};
            return meeting[id];
        } catch (e) {
            notificationService.showError('Не вдалось з`єднатись');
        }
    }

    const sendMessage = (text, from, to) => {
        try {
            const meetings =
JSON.parse(localStorage.getItem(storage.meeting)) || {};
            const findMeeting = Object.keys(meetings).filter(id =>
(meetings[id]?.user === from && meetings[id]?.doctor === to) ||
(meetings[id]?.user === to && meetings[id]?.doctor === from));
            const meetingId = findMeeting[0];

            const newDate = {
```

```
                ...meetings,
                [meetingId]: {
                    ...meetings[meetingId],
                    chat: [
                        ...meetings[meetingId]?.chat,
                        {
                            id: meetings[meetingId]?.chat?.length,
                            message: text,
                            from: from,
                            time: new Date()
                        }
                    ]
                }
            }

            localStorage.setItem(storage.meeting,
JSON.stringify(newDate));
        } catch (e) {
            notificationService.showError('Не вдалось надіслати');
        }
    }

    const getMessages = (from, to) => {
        try {
            const meetings =
JSON.parse(localStorage.getItem(storage.meeting)) || {};
            const findMeeting = Object.keys(meetings).filter(id =>
(meetings[id]?.user === from && meetings[id]?.doctor === to) ||
(meetings[id]?.user === to && meetings[id]?.doctor === from));
            const meetingId = findMeeting[0];
            return meetings[meetingId]?.chat;
        } catch (e) {
            notificationService.showError('Не вдалось отримати
діалог');
        }
    }

    const server = {
        register,
        login,
        deleteAccount,
        changeInfo,
        setMeeting,
        getMeetings,
        getDoctors,
        registerDoctor,
        getMeetingId,
        getMeeting,
        sendMessage,
        getMessages
```

```
}

export default server;
```

search.js

```
import { useState } from "react";
import { useSelector } from "react-redux";
import {
ApproveStyle,
DoctorInfo,
DoctorInformation,
DoctorsWrapper,
Options,
SearchStyle,
SearchWrapper,
SlotsWrapper
} from "./styled";
import { selectToken } from "../../../state/selectors/global";
import { groups } from "../../../consts/groups";
import { Input } from "../../../components/Input";
import { CalendarSvg, ManSvg, WomanSvg } from
"../../../components/Svg";
import server from "../../../services/server";
import { getDoctorAvailableSlots, getDoctorFirstAvailableSlot,
slotToString } from "../../../services/doctorService";
import routePaths from "../../../consts/routePaths";
import { useHistory, useLocation } from "react-router-dom";
import { MeetingDetail } from "../components/MeetingDetail";
import Toggle from "../../../components/Toggle";

export const Search = () => {
const history = useHistory();
const location = useLocation();
const [mode, setMode] = useState('');
const [openDoctor, setOpenDoctor] = useState(false);
const [option, setOption] = useState(false);
const [date, setDate] = useState({});
// const [interaction, setInteraction] = useState({});
const user = useSelector(store => selectToken()(store));
const isAdmin = user?.groups?.includes(groups.admin);
const isPatient = user?.groups?.includes(groups.patient);

const searchParam = new
URLSearchParams(location.search).get('search');

const [search, setSearch] = useState('');
const [specialization, setSpecialization] =
useState(searchParam);
```

```
      const [toggle, setToggle] = useState(true);
      const [toggle2, setToggle2] = useState(true);

      const doctors = server.getDoctors(specialization, search);

      const today = new Date();
      const tomorrow = new Date(today);
      const tomorrow2 = new Date(today);
      tomorrow.setDate(tomorrow.getDate() + 1);
      tomorrow2.setDate(tomorrow.getDate() + 1);

      const onRecord = (dayOffset, time) => {
      const d = new Date(today);
      d.setDate(d.getDate() + dayOffset);
      setDate({
      date: d.toLocaleDateString(), time
      })
      setMode('approveTime');
      };

      const onWriteRecord = () => {
      server.setMeeting({ ...date, doctor: openDoctor?.email, user:
user?.email, type: !option ? 'Відеоконференція' : 'Чат',
specialization: openDoctor?.specialization })
      history.push(routePaths.USER_HISTORY);
      };

      return (
      <SearchStyle>
      {isAdmin &&
      <>
      <h2 className={'title'}>Виберіть, будь ласка, знайти чи додати
лікаря</h2>
      <div className="mode" onClick={() =>
setMode('search')}>Пошук</div>
      <div className="mode" onClick={() =>
setMode('add')}>Додати</div>
      </>
      }
      {(isPatient || mode === 'add') && !openDoctor &&
      <SearchWrapper>
      {isPatient && <h3 className="title-for-patient">В нашому
сервісі Ви можете отримати допомогу, знайдіть послугу за ПІБ лікаря
чи за спеціалізацією.</h3>}

      <Input
      value={search}
      placeholder={'Почніть вводити  ПІБ лікаря  '}
      onChange={(e) => setSearch(e.target.value)}
      />
```

```
      <Input
      value={specialization}
      placeholder={'Почніть вводити назву послуги, наприклад:
кардіолог'}
      onChange={(e) => setSpecialization(e.target.value)}
      />

      <DoctorsWrapper>
      {doctors?.length > 0 ? doctors.map(item =>
      <div className="item" onClick={() => setOpenDoctor(item)}>
      {item.male ? <ManSvg /> : <WomanSvg />}

      <div>
      <h4 className="name">{item.name} {item?.surname}
{item?.lastname}</h4>
      <h4 className="doctor">{item?.specialization}</h4>
      <div className="info">
      <CalendarSvg />
      <h5>Найближчий вільний слот:</h5>
      <h6>{slotToString(getDoctorFirstAvailableSlot(item?.email))}</h
6>
      </div>
      </div>
      </div>
      ) :
      <h3 className="title-for-patient">Не знайдено жодного лікаря,
спробуйте ще.</h3>
      }
      </DoctorsWrapper>
      </SearchWrapper>
      }

      {openDoctor && mode === '' &&
      <DoctorInfo>
      {openDoctor?.male ? <ManSvg /> : <WomanSvg />}
      <h3 className="name">{openDoctor.name} {openDoctor?.surname}
{openDoctor?.lastname}</h3>
      <h4 className="doctor">{openDoctor?.specialization}</h4>

      <div className="info">
      <CalendarSvg />
      <h5>Найближчий вільний слот:</h5>
      <h6>{slotToString(getDoctorFirstAvailableSlot(openDoctor?.email
))}</h6>
      </div>

      <h4 style={{ marginTop: '15px' }} className={'doctor'}>Оберіть,
будь ласка, послугу, яко хотіли б замовити.</h4>

      <Options>
```

```
      <div className={`option ${option ? 'active' : ''}`} onClick={()
=> setOption(true)}>
      <h5>Консультація. Чат</h5>
      <h5><CalendarSvg />{openDoctor?.price?.chat}</h5>
      </div>

      <div className={`option ${!option ? 'active' : ''}`}
onClick={() => setOption(false)}>
      <h5>Консультація. Відеоконференція</h5>
      <h5><CalendarSvg />{openDoctor?.price?.video} грн</h5>
      </div>
      </Options>

      <button onClick={() => setMode('selectTime')}>Запис на
прийом</button>

      <h4 className="about">Про лікаря</h4>

      <DoctorInformation>
      <h5><strong>Стаж роботи:</strong> 5 років</h5>
      <h5><strong>Освіта:</strong> Національний медичний університет
ім. О.О. Богомольця</h5>
      <h5><strong>Категорія:</strong> Вища</h5>
      <h5><strong>Мови:</strong> Українська English Deutsch</h5>
      <h5><strong>Порада від лікаря:</strong> «Займайтесь
профілактикою! Це набагато краще, ніж пізніше лікувати запущену
хворобу»</h5>
      <h5><strong>Девіз:</strong> «An apple a day keeps the doctor
away»</h5>
      </DoctorInformation>
      </DoctorInfo>
      }

      {mode === 'selectTime' &&
      <SlotsWrapper>
      {/* <div className="item">
      <h5>{today.toLocaleDateString()}</h5>
      {getDoctorAvailableSlots(openDoctor?.email).map(item => <div
className="time" onClick={() => onRecord(0,
item)}>{slotToString(item)}</div>)}
      </div> */}

      <div className="item">
      <h5>{tomorrow.toLocaleDateString()}</h5>
      {getDoctorAvailableSlots(openDoctor?.email,
tomorrow.toLocaleDateString()).map(item => <div className="time"
onClick={() => onRecord(1, item)}>{slotToString(item)}</div>)}
      </div>

      <div className="item">
```

```
      <h5>{tomorrow2.toLocaleDateString()}</h5>
      {getDoctorAvailableSlots(openDoctor?.email,
tomorrow2.toLocaleDateString()).map(item => <div className="time"
onClick={() => onRecord(2, item)}>{slotToString(item)}</div>)}
      </div>
      </SlotsWrapper>
      }

      {mode === 'approveTime' &&
      <ApproveStyle>
      <h3>Підтвердження запису</h3>

      <MeetingDetail date={date} user={user} doctor={openDoctor}
option={option} />

      <h5 className={'comment'}>Поділіться, з чим Ви до нас прийшли
:)</h5>
      <Input
      placeholder={'Короткий опис проблеми (не обов’язково)'}
      />

      <div className="toggle">
      <h6>Передзвоніть мені</h6>
      <Toggle
      value={toggle}
      setValue={setToggle}
      />
      </div>

      <div className="toggle agreement">
      <h6>Я прочитав(ла) і згоден(дна) з умовами використання
сервісу</h6>
      <Toggle
      value={toggle2}
      setValue={setToggle2}
      />
      </div>

      <button onClick={onWriteRecord}>Записатися</button>
      </ApproveStyle>
      }
      </SearchStyle>
      )
      };
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБПЛАТФОРМА ДЛЯ ТЕЛЕМЕДИЦИНИ. КЛІЄНТСЬКА ЧАСТИНА

Виконав: студент групи КП-93 Петренко Нікіта Андрійович

Керівник: асистент Жикін Юрій Сергійович

Київ – 2023

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити клієнтську частину вебплатформи для телемедицини

Завдання:

1. Проаналізувати застосунки для телемедицини на ринку України
2. Обґрунтувати вибір платформи для користувацького інтерфейсу застосунку
3. Розробити клієнтську частину вебплатформи для телемедицини
4. Описати напрямки подальшого вдосконалення ПЗ

АКТУАЛЬНІСТЬ

1. Розвиток технологій розроблення вебзастосунків
2. Розвиток телемедицини в Україні і світі

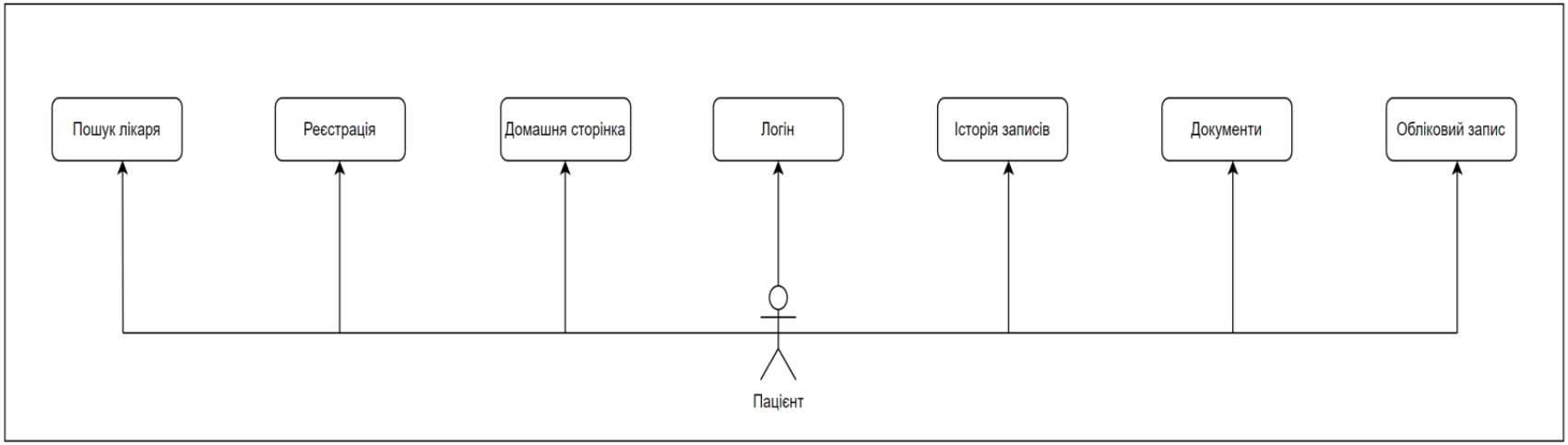
ІСНУЮЧІ АНАЛОГИ

Послуги	Medikit	Likar Online	Doctor Online
Android платформа	–	–	+
iOS платформа	–	–	+
Відеоконференції	+	+	+
Чат	+	+	+
Відстеження показників здоров’я	–	–	+
Сповіщення та нагадування	–	+	+

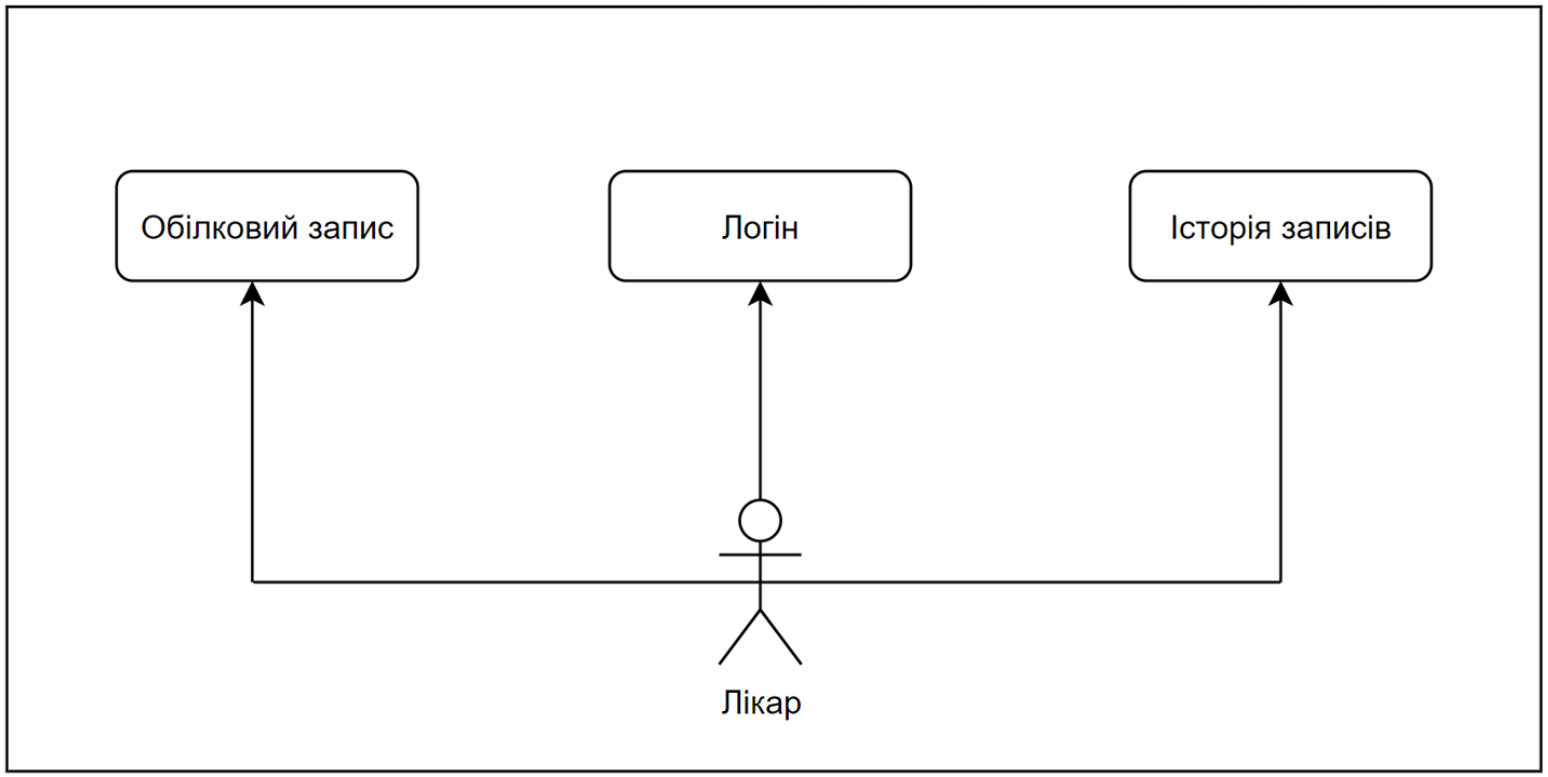
ЗАСОБИ РЕАЛІЗАЦІЇ: ЗАГАЛЬНІ ВІДОМОСТІ

- JavaScript
- React
- Sass
- SocketIO
- SimpleWebRTC

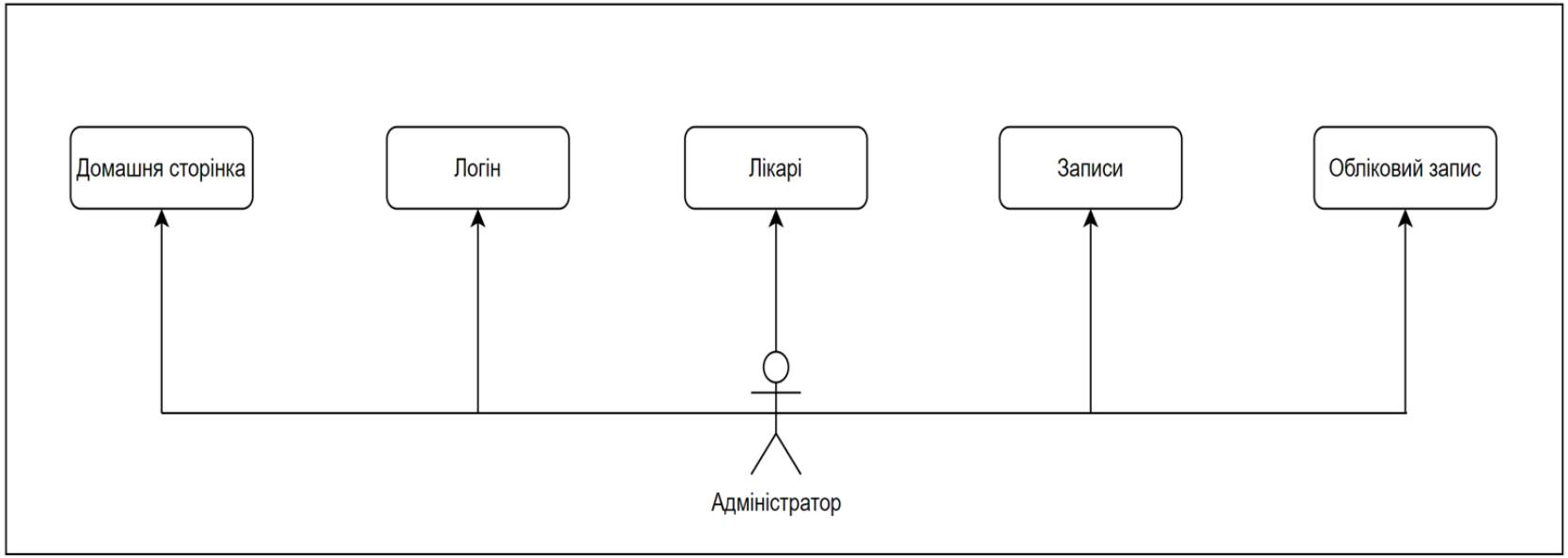
ДОСТУПНІ СТОРІНКИ ПАЦІЄНТ



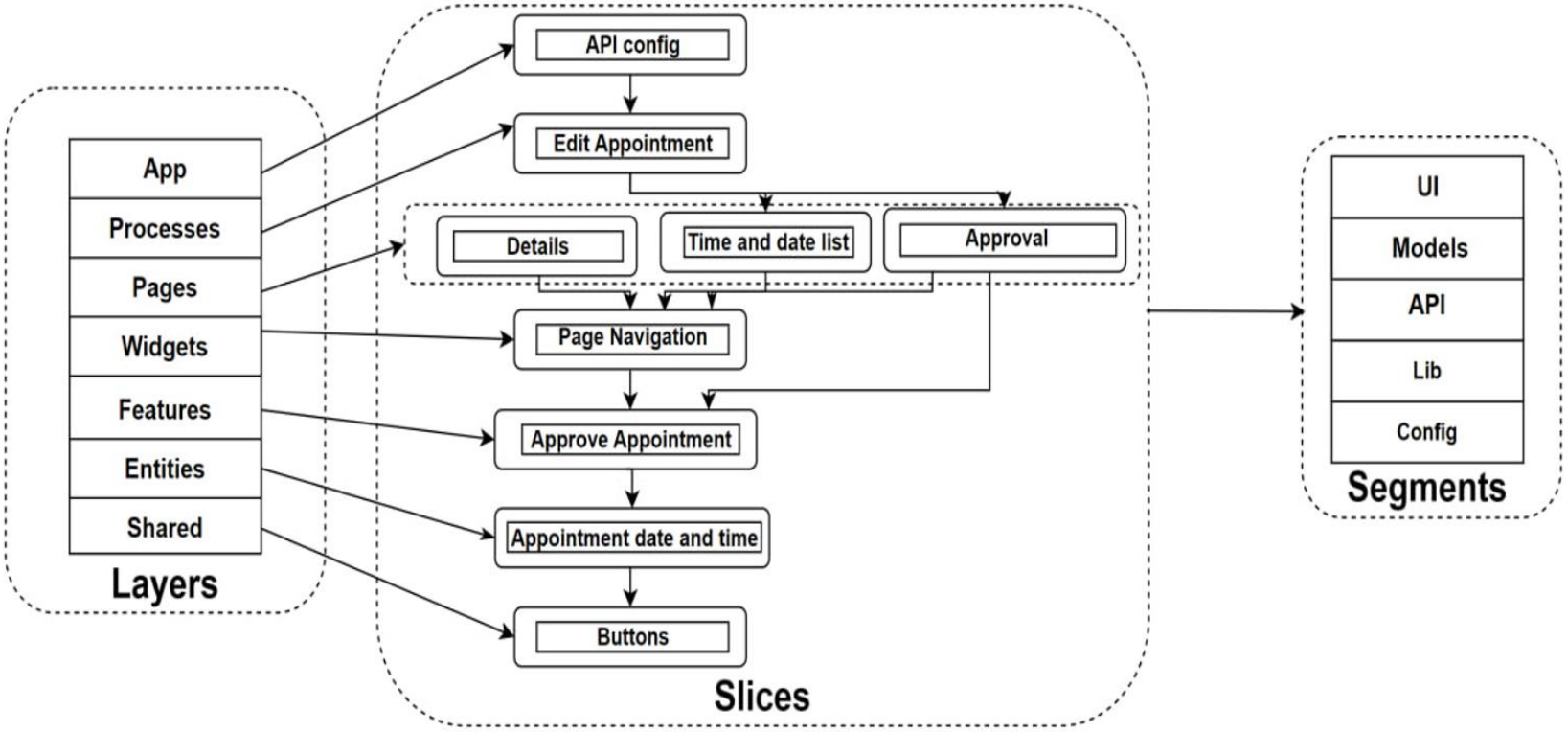
ДОСТУПНІ СТОРІНКИ
ЛІКАР



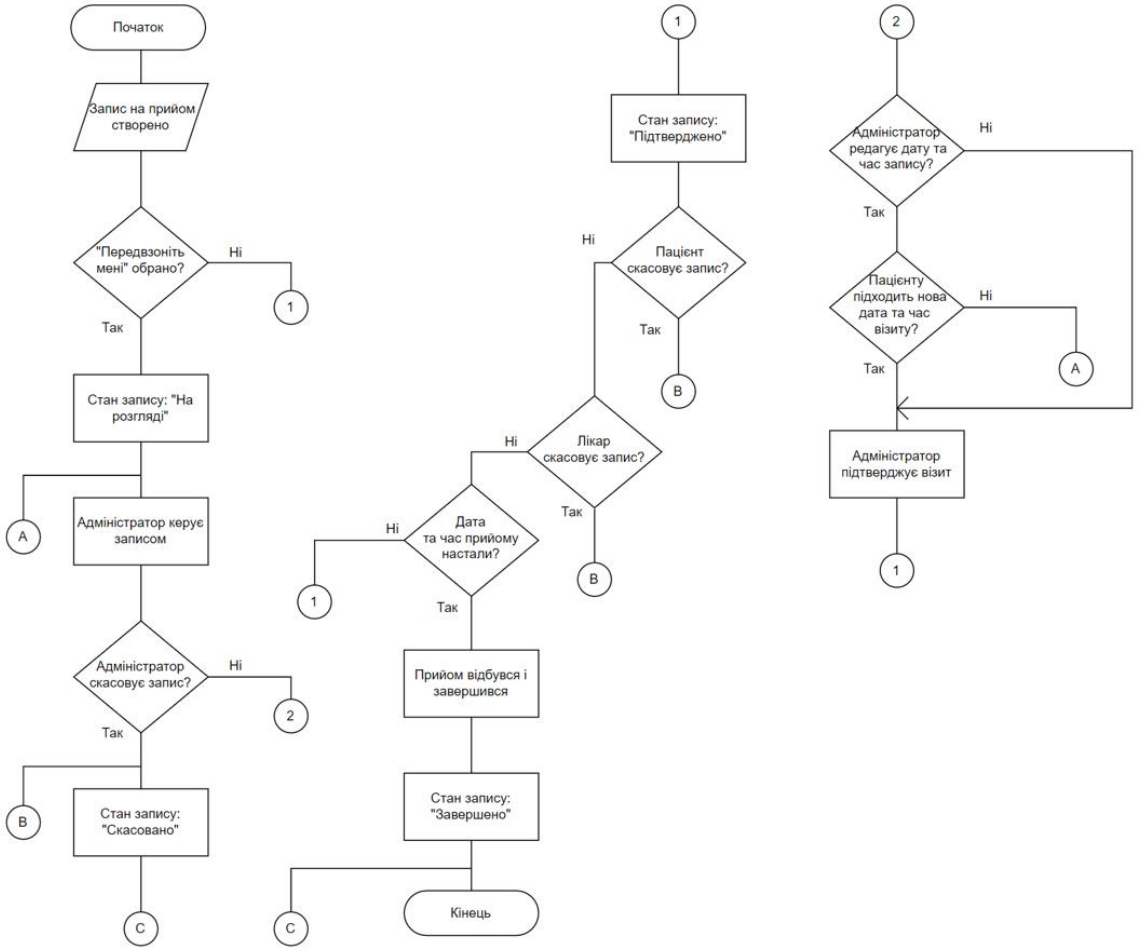
ДОСТУПНІ СТОРІНКИ АДМІНІСТРАТОР



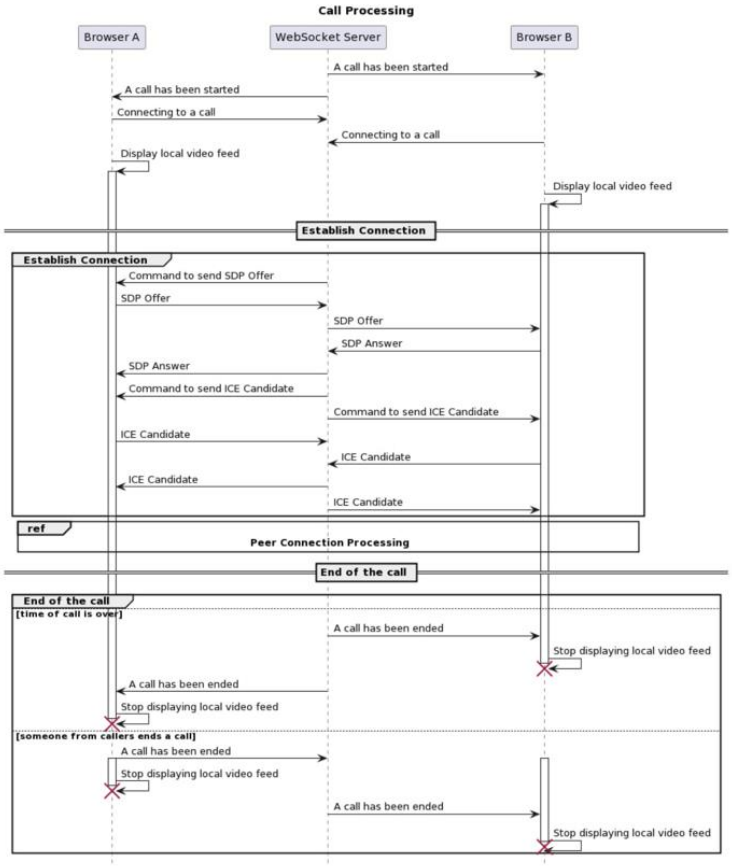
АРХІТЕКТУРА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБПЛАТФОРМИ

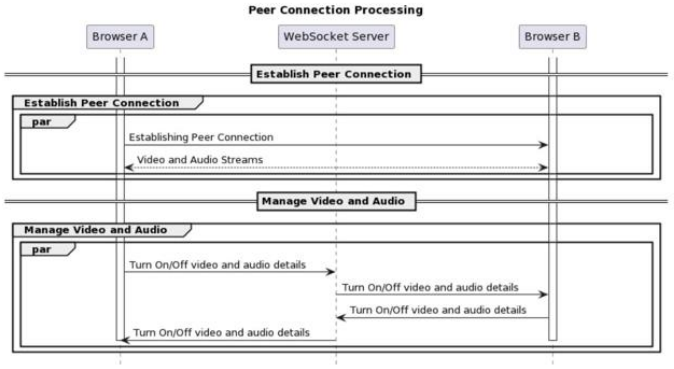


АЛГОРИТМ КЕРУВАННЯ ЗАПИСАМИ НА ПРИЙОМ



ПРОЦЕС ПРОВЕДЕННЯ ВІДЕОКОНФЕРЕНЦІЙ





НАПРЯМИ РОЗВИТКУ

1. Впровадження стандарту вебдоступності WCAG
2. Впровадження функціональності «Відновити пароль» використовуючи двофакторну аутентифікацію
3. Додавання локалізаційних файлів

ВИСНОВКИ

В процесі роботи над даним проєктом було зроблено:

1. Знайдено та проаналізовано існуючі програмні рішення на ринку України .
2. Сформовано і визначено вимоги до розроблюваного ПЗ.
3. Розроблено клієнтську частину вебплатформи для телемедицини, що реалізує основну функціональність ПЗ для телемедицини.
4. Проведено тестування отриманого ПЗ та проаналізовано результати.
5. Знайдено шляхи подальшого розвитку проєкту.

РЕЗУЛЬТАТИ ПЕРЕВІРКИ НА УНІКАЛЬНІСТЬ

UNICHECK
by Turnitin

User name:
Жикін Юрій Сергійович

Check ID:
1015619189

Check date:
15.06.2023 22:21:10 EEST

Check type:
Doc vs Internet + Library

Report date:
18.06.2023 22:54:12 EEST

User ID:
100012489

File name: **petrenko-antiplagiarism**

Page count: **48** Word count: **9397** Character count: **72461** File size: **523.45 KB** File ID: **1015262103**

2.32% Matches

Highest match: **1.33%** with Internet source (**https://sdh.com.ua/blog/mobile/Rozrobka_Telemedicine_Web**)

1.82% Internet sources	**9**	Page 50
0.89% Library sources	**51**	Page 50

0% Quotes

No quotes found

Exclusion of references is off

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ВЕБПЛАТФОРМА ДЛЯ ТЕЛЕМЕДИЦИНИ. КЛІЄНТСЬКА ЧАСТИНА

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Юрій ЖИКІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Нікіта ПЕТРЕНКО

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Клієнтська частина вебплатформи для телемедицини. Розроблене програмне забезпечення написане мовою програмування JavaScript із використанням фреймворку React.

2. МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка наступних елементів:

1. Працездатність функціональності реєстрації та автентифікації.
2. Функціональна працездатність елементів графічного інтерфейсу користувача вебплатформи.
3. Коректна робота функціональності запису на прийом.
4. Коректна робота функціональності пошуку лікарів.
5. Коректна робота функціональності перегляду історії записів.
6. Коректна робота функціональності приєднання до прийому у форматі чату.
7. Коректна робота функціональності приєднання до прийому у форматі відеоконференції.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом сірого ящику або Gray Box Testing. Детальніше розглянемо особливості застосування даної техніки та зазначимо ключові компоненти інших підходів, що стали її основою:

1. White Box Testing – основною особливістю даного підходу є проведення тестування на основі логіки роботи елементів коду програми. Для точного визначення частини коду, котра має помилку, тестувальник повинен мати комплексні знання, щодо внутрішнього функціонування системи.
2. Black Box Testing – даний підхід реалізує процес тестування за рахунок використання функцій системи через графічні

інтерфейси користувача, без використання будь-яких знань щодо внутрішньої структури системи.

Застосування методу сірого ящику дозволяє поєднати попередньо розглянуті підходи та надає можливість проводити тестування програмного забезпечення, маючи обмежені знання про внутрішні особливості функціонування системи. Даний підхід виконує перевірку як вихідного коду, так і окремих компонентів програмного продукту на відповідність існуючим функціональним та нефункціональним вимогам.

Використовуються такі методи тестування:

1. Функціональне тестування.
2. Автоматизоване тестування графічного інтерфейсу користувача.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Повний процес тестування проходить у такому порядку:

1. Статичне тестування коду.
2. Динамічне автоматизоване тестування полів вводу на граничні та неможливі значення.
3. Динамічне автоматизоване тестування на відповідність функціональним вимогам.
4. Тестування інтерфейсу при різній роздільній здатності екрану.
5. Тестування зручності використання.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ВЕБПЛАТФОРМА ДЛЯ ТЕЛЕМЕДИЦИНИ. КЛІЄНТСЬКА ЧАСТИНА.

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Юрій ЖИКІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Нікіта ПЕТРЕНКО

2023

ЗМІСТ

1. СТРУКТУРА КЛІЄНТСЬКОЇ ЧАСТИНИ

Клієнтська частина вебплатформи для телемедицини складається з динамічних сторінок. Українська мова є головною і єдиною мовою інтерфейсу. Графічний інтерфейс сторінок формується відповідно до ролей користувача, а саме пацієнт, лікар, адміністратор. Однак, є сторінки, що є спільними для усіх користувачів. Детальніше розглянемо переліки доступних сторінок для кожної з зазначених ролей.

Спільні сторінки: логін, облікова інформація

Графічна частина розбита на функціональності, де кожна функціональність має від двох до 10 сторінок. Для навігації у застосунку, розміщена панель з відповідними графічними елементами керування. Слід зазначити, що так як доступна функціональність залежить від ролі користувача, тому й елементи на панелі навігації будуть різні для кожної групи. (рис. 1.1-1.3). Навігаційна панель відображається на кожній сторінці клієнтської частини, для побудови найоптимальнішого рішення навігації.

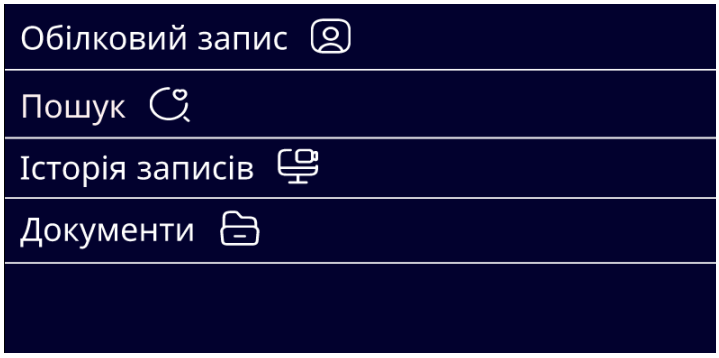


Рис. 1.1. Елементи панелі навігації користувача

Рис. 1.2. Елементи панелі навігації лікаря

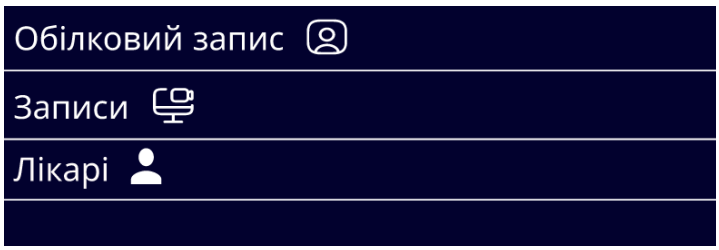


Рис. 1.3. Елементи панелі навігації адміністратора

2. ПРОЦЕДУРА РЕЄСТРАЦІЇ ТА АВТОРИЗАЦІЇ

Для того, щоб користувач зміг користуватися усім доступним функціоналом пацієнта, користувачу необхідно пройти процедуру реєстрації та авторизації. Реєстрація користувача здійснюється у три кроки (рис. 2.1-2.3): ввід персональної інформації (ПІБ, номер телефону, стать), ввід електронної пошти, логіну, паролю та підтвердження паролю і останній етап за ввести код підтвердження, що був надісланий на електронну пошту, що була вказана у попередньому кроці. Також на останньому кроці є можливість отримати новий код. Слід зазначити, що у випадку коли усі обов’язкові поля не є заповненні, користувач не зможе перейти до наступного кроку.

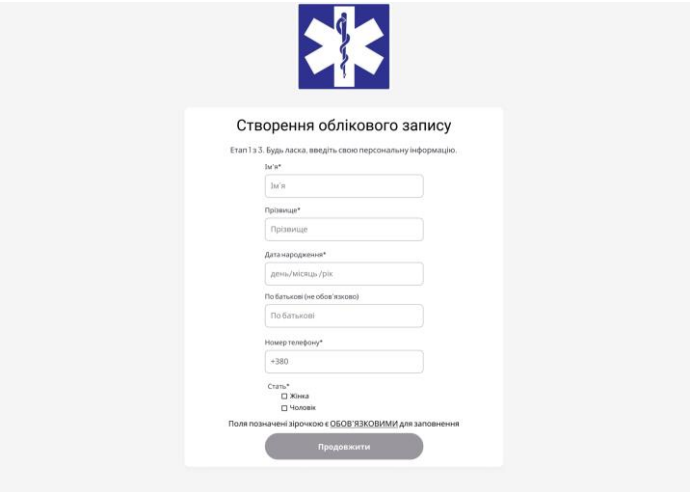


Рис. 2.1. Перший етап реєстрації користувача

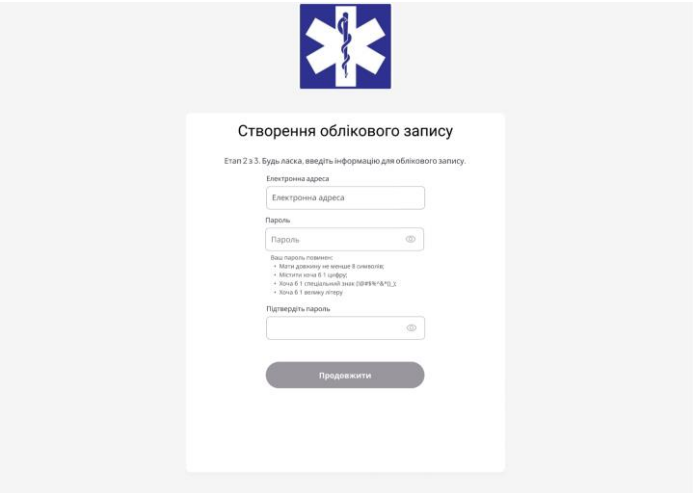


Рис. 2.2. Другий етап реєстрації користувача

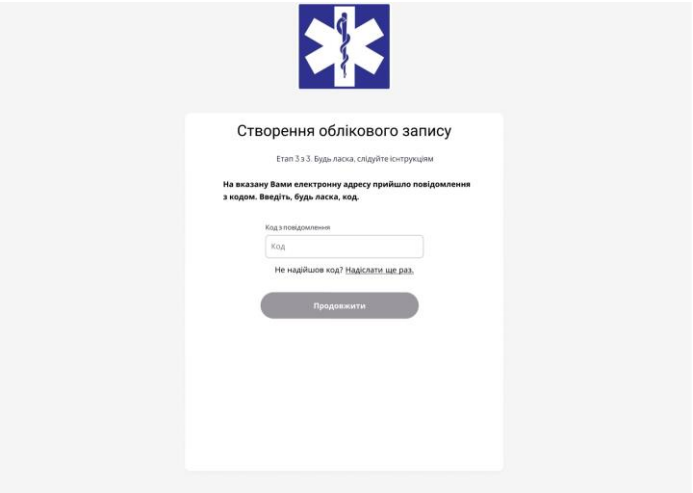


Рис. 2.3. Третій етап реєстрації користувача

Щоб авторизуватися, користувачу необхідно ввести електронну адресу, на яку був зареєстрований обліковий запис, та пароль (рис. 2.4). Також на сторінці логіну, є пропозиція зареєструватися, якщо користувач не має облікового запису.

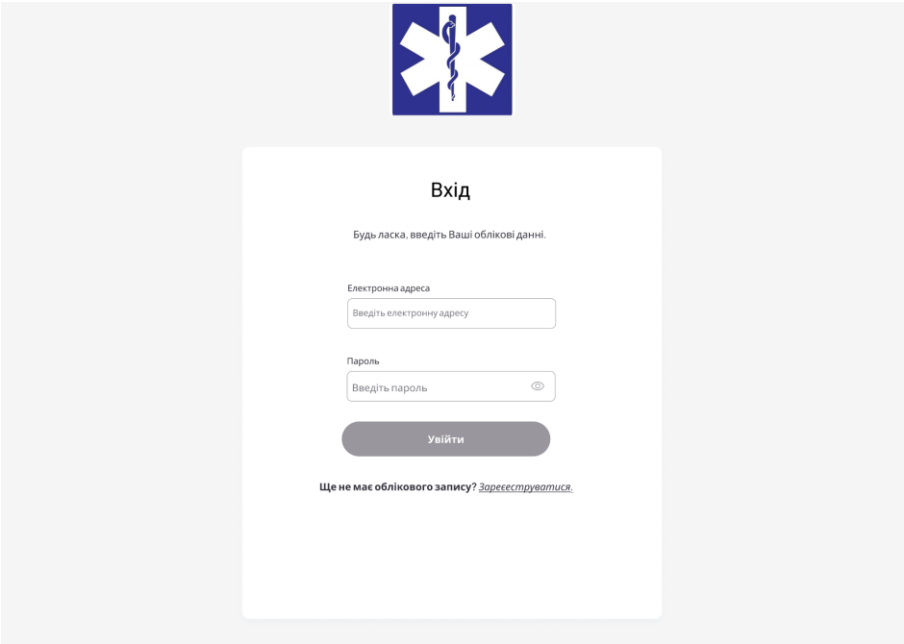


Рис. 2.4. Сторінка логіну

Після реєстрації та логіну, домашня сторінка пацієнта відображається у веббраузері (рис. 2.5).

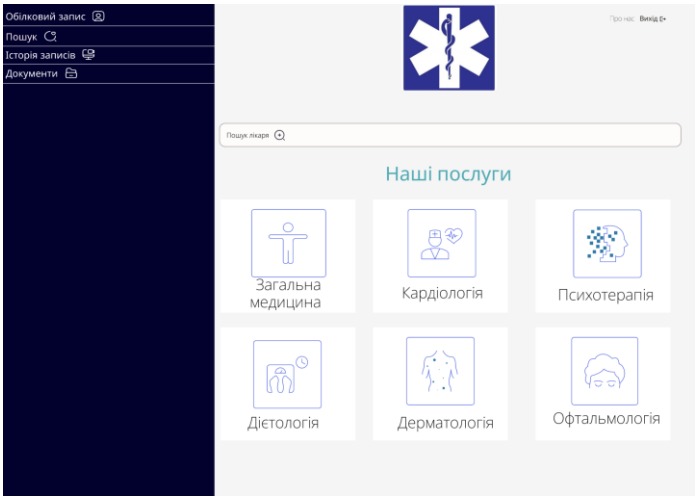


Рис. 2.5. Домашня сторінка користувача

3. ПРОЦЕДУРА ЗАПИСУ НА ПРИЙОМ

Для запису на прийом до лікаря, користувачу необхідно спочатку знайти лікаря, це можна зробити або обравши спеціалізацію на домашній сторінці, або натиснувши на «Пошук» на панелі навігації. Яка б опція не

була обрана, користувацький інтерфейс надає можливість шукати лікаря за ПІБ (рис. 3.1). Якщо користувач не обрав спеціалізацію на домашній сторінці, тоді є можливість знайти усіх лікарів, що мають шукану спеціалізацію (рис. 3.2). Після пошуку лікаря, слід обрати одну послугу з тих, що надає лікар (рис. 3.3). Після обрання послуги, необхідно обрати дату та час прийому (рис. 3.4). Коли дата та час обрані, користувач підтверджує запис на прийом (рис. 3.5). Також наявна опція «Передзвоніть мені». Обравши її, прийом не буде доданий до списку запланованих, поки адміністратор системи не підтвердить його. Після підтвердження запису, користувачу відображається список запланованих візитів.

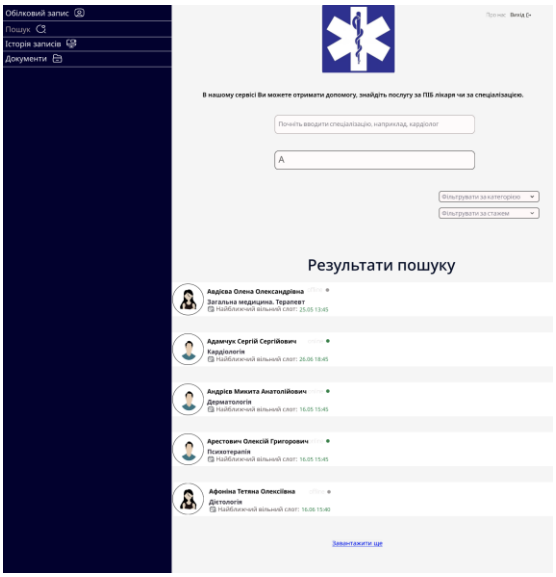


Рис. 3.1. Пошук лікаря за ПІБ.

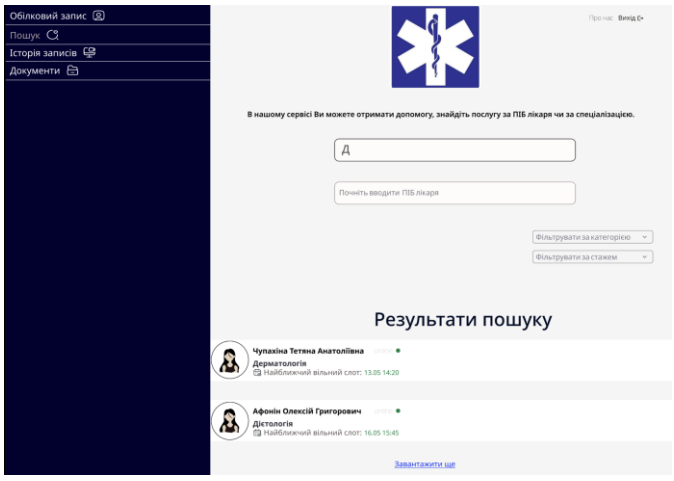


Рис. 3.2. Пошук лікаря за спеціалізацією

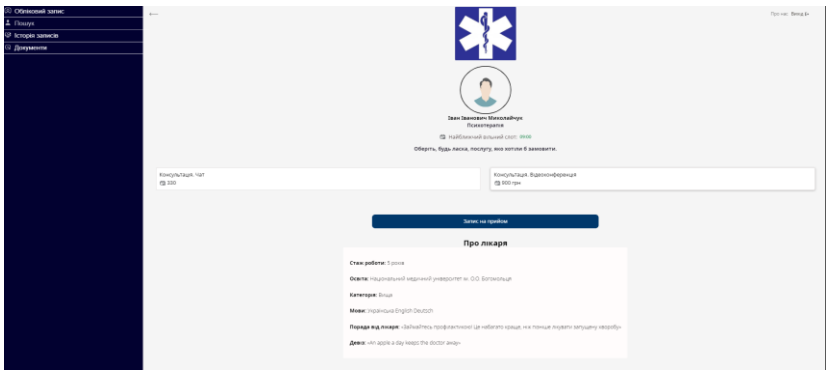


Рис. 3.3. Інформація про лікаря

Рис 3.4. Таблиця з доступним часу для запису

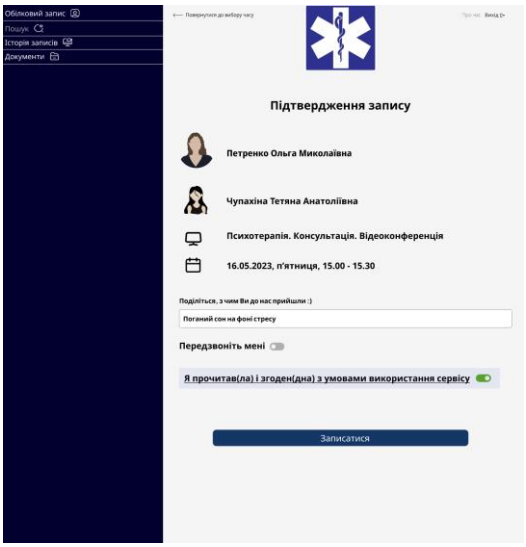


Рис. 3.5. Підтвердження запису

4. ІСТОРІЯ ЗАПИСІВ НА ПРИЙОМ

Користувач має можливість переглядати історію записів, як запланованих так і усіх минулих Для цього слід натиснути на «Історія записів» в панелі навігації. Обрати, які саме візити, заплановані чи минулі дивитися, можна за допомогою меню на самій сторінці. У разі, якщо немає жодних запланованих прийомів, відповідне повідомлення відображається у вкладці «Заплановані» (рис. 4.1). У разі, якщо немає жодних минулих прийомів, відповідне повідомлення відображається у вкладці «Минулі» (рис. 4.2).

Після натискання на коротку інформацію про запланований запис, відображається інформація про запис, та посилання на прийом, якщо час прийому настав (рис. 4.3). Якщо час прийому не настав, то користувачу відображається повідомлення, що посилання буде доступним у час прийому (рис. 4.4).

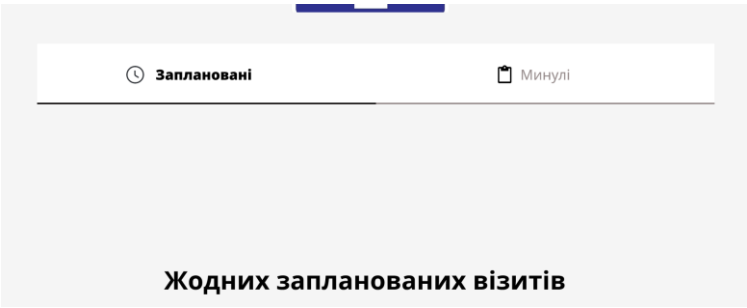


Рис. 4.1. Повідомлення у разі жодних запланованих візитів

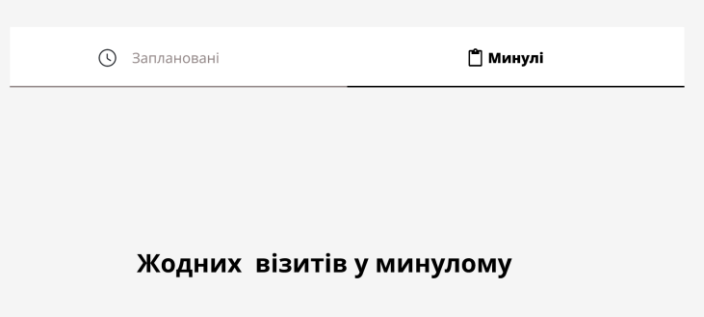


Рис. 4.2. Повідомлення у разі жодних минулих візитів

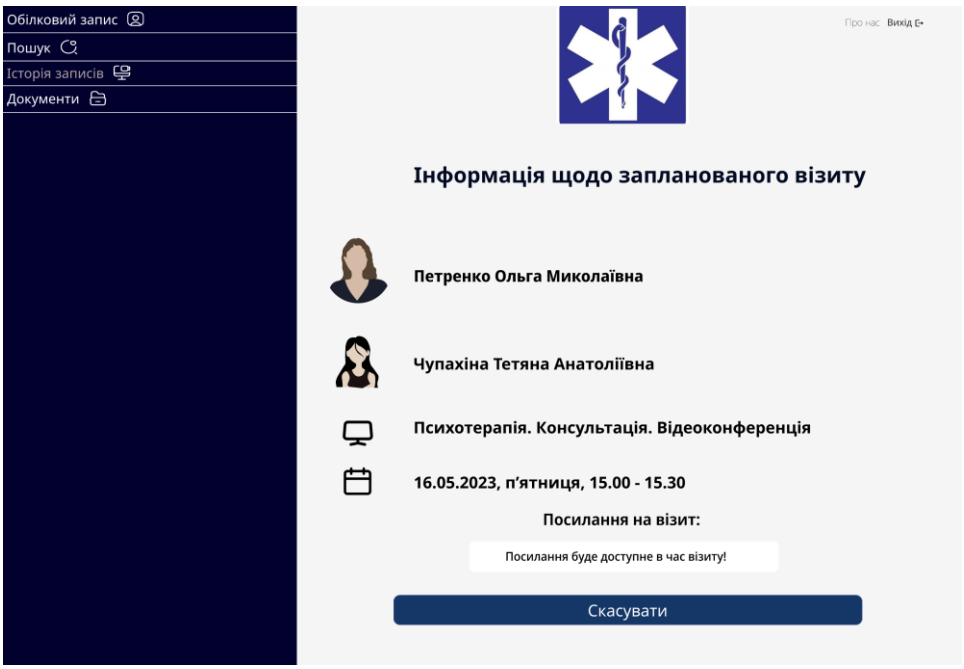


Рис. 4.3. Інформація щодо запланованого візиту, час та дата якого не настали

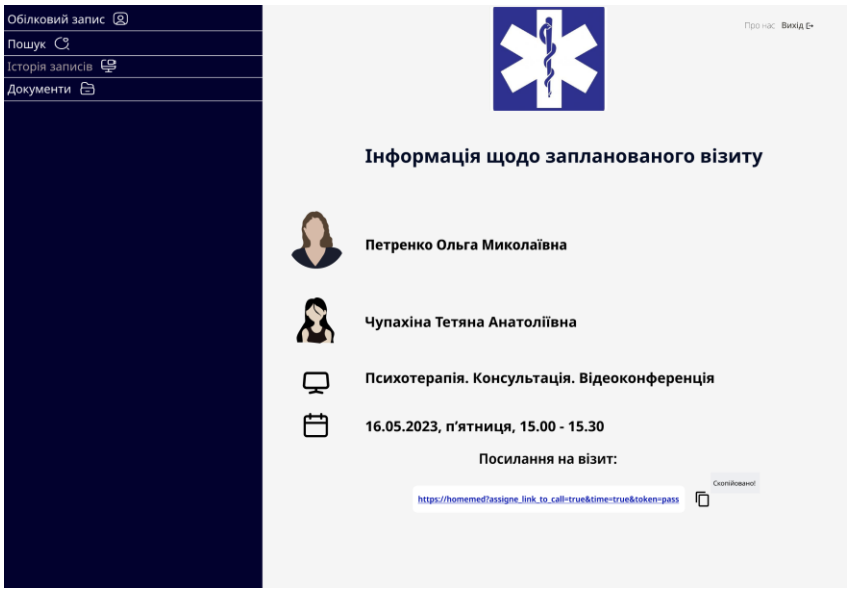


Рис. 4.4. Інформація щодо запланованого візиту, час та дата якого настали

5. ПІДКЛЮЧЕННЯ ДО ПРИЙОМУ

Щоб підключитися до прийому, необхідно обрати прийом зі списку запланованих, що вже настав, і перейти по посиланню на візит. Якщо прийом у форматі чату, користувачу відображається сторінка чату з лікарем

(рис. 5.1). Користувач має змогу надсилати як текстові повідомлення, так і файли (розширень: PDF, DOC, DOCX, JPG, JPEG, PNG, GIF). Якщо візит у форматі чату, після переходу за посиланням, користувачеві відображається сторінка з активним дзвінком (рис. 5.2). Користувач має змогу увімкнути та вимкнути свій аудіо та відео потік, натиснувши на відповідні іконки. Також користувач бачить чи увімкнутий або вимкнутий відео та аудіо потік лікаря.

Після завершення прийому користувачеві пропонується залишити відгук про якість прийому. В залежності від типу прийому, відеоконференція чи чат, різні критерії оцінки прийому доступні для користувача (рис. 5.3, 5.4).

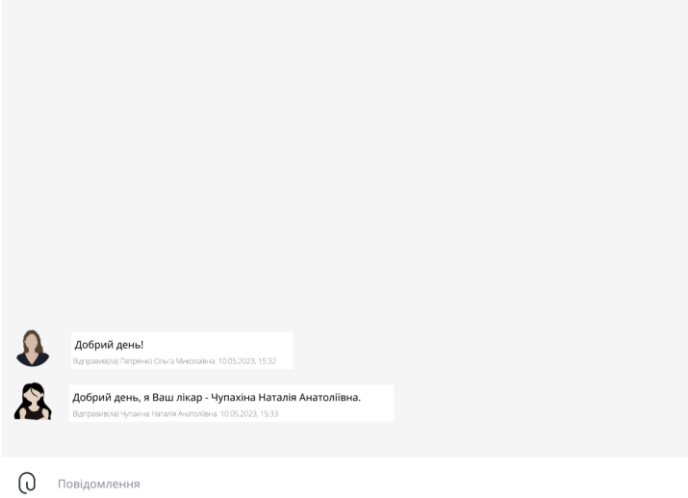


Рис. 5.1. Прийом у форматі чату

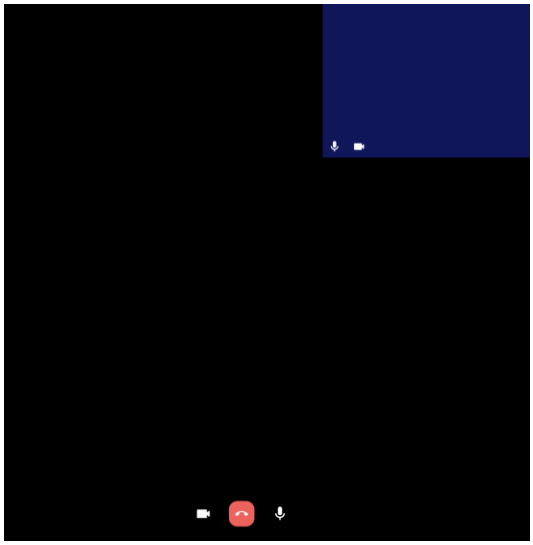


Рис. 5.2. Вікно відеоконференції

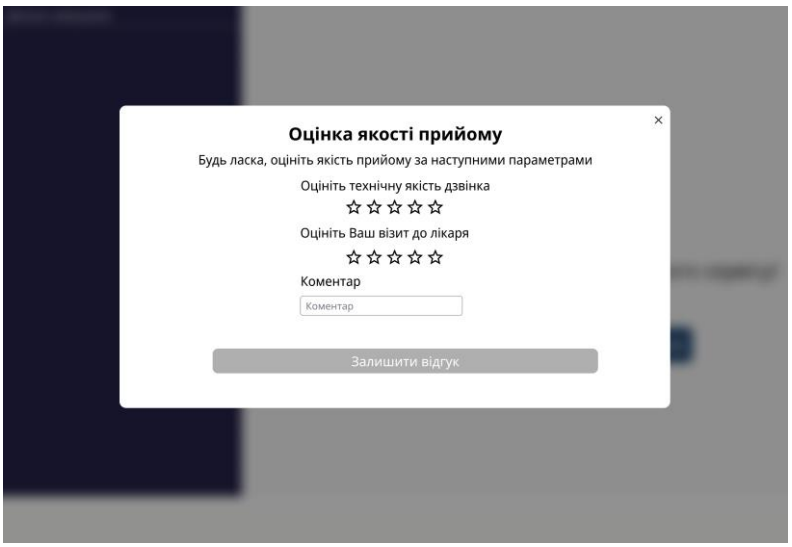


Рис. 5.3. Модальне вікно оцінки якості прийому у форматі відеоконференції

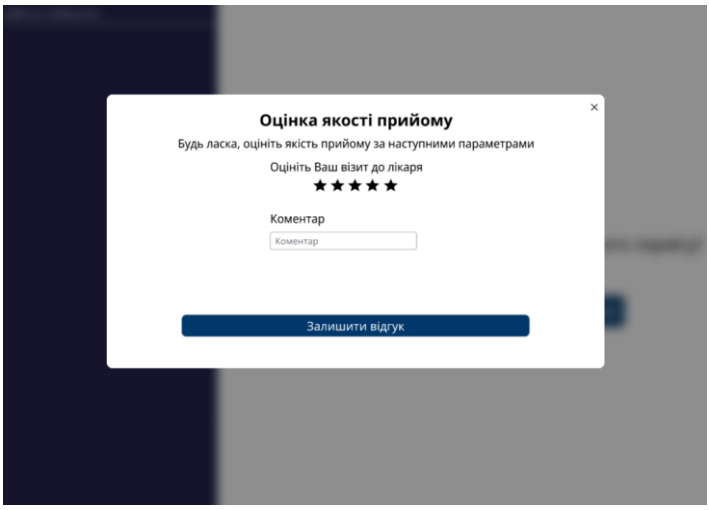


Рис. 5.4. Модальне вікно оцінки якості прийому у форматі чату

З початку відеоконференції, в більшому масштабі відображається відеопотік лікаря, однак, після натискання курсором миші два рази на відображення власного відеопотоку (синій квадрат зверху справа на рис. 5.2), власний відеопотік міняється місцями з відображенням відеопотоку лікаря.

6. РОБОТА З ДОКУМЕНТАМИ

В меню навігації, для пацієнта наявна функціональність «Документи». Цей розділ зроблений для того, щоб надати змогу пацієнтам переглядати та

завантажувати документи на свій персональний комп’ютер, що були прикріпленні лікарем під час або після візиту.

Натиснувши на «Документи» в меню навігації, у веббраузері відкривається нова вебсторінка сторінка з короткою інформацією про усі минулі прийоми, для яких лікар завантажив документи (рис. 6.1). Для користувача доступний пошук документів за ПІБ лікаря чи за спеціалізацією лікаря.

У разі якщо прийомів ще не було, відображається повідомлення про відсутність документів (рис. 6.2).

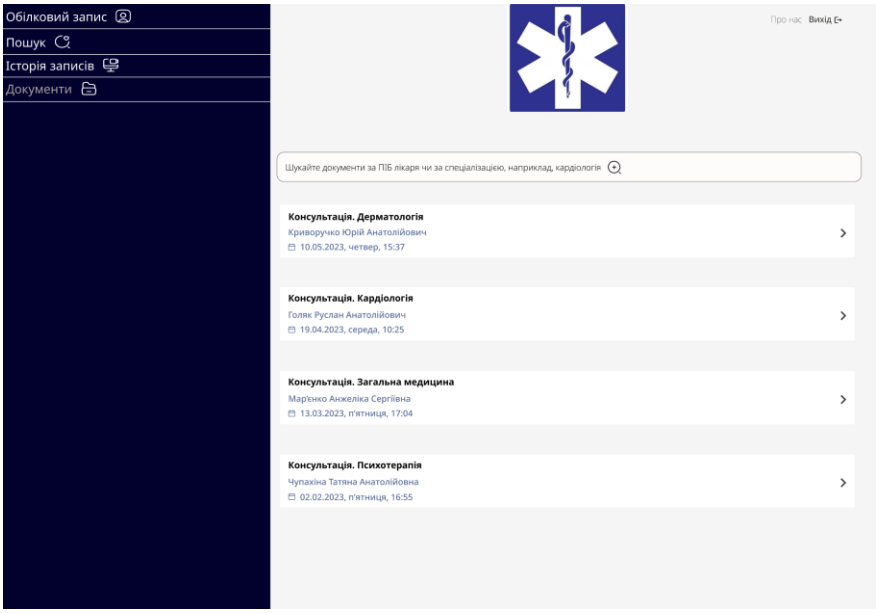


Рис. 6.1. Список візитів для яких є завантажені документи

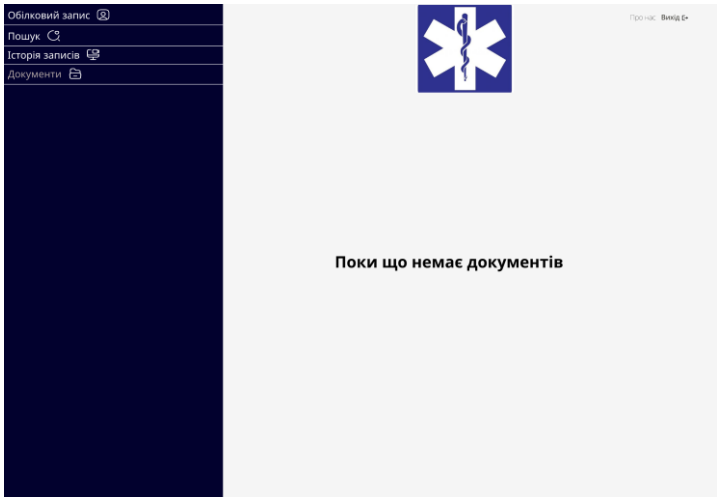


Рис. 6.2. Повідомлення про відсутність завантажених документів

Натиснувши на коротку інформацію про візит (рис. 6.1), у веббраузері відкривається нова вебсторінка зі змістом документу. Як було вище зазначено, у лікаря є можливість завантажити документи, він може це зробити, натиснувши на кнопку «Завантажити».

7. ОБЛІКОВИЙ ЗАПИС

Для того щоб переглядати та змінювати свої персональні дані, логін та пароль користувачу необхідно перейти в розділ «Обліковий запис. Також пацієнт має можливість видалити свій обліковий запис (рис. 7.1).

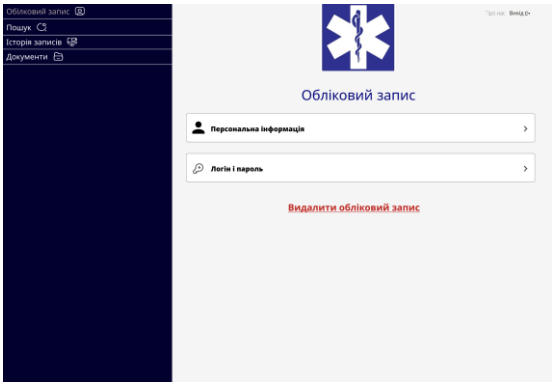


Рис. 7.1. Розділ «Обліковий запис»

Пацієнт має змогу переглянути та редагувати надану їм персональну інформацію, натиснувши на «Персональна інформація» (рис. 7.2). Після збереження редагованих даних, повідомлення про успішне оновлення інформації буде відображатися (рис. 7.3). Щоб змінити логін, та встановити новий пароль, користувачеві необхідно перейти в «Логін і пароль» (рис. 7.4). Для зміни паролю, користувач повинен ввести старий пароль, новий пароль, відповідно до вимог, та повторити новий пароль.

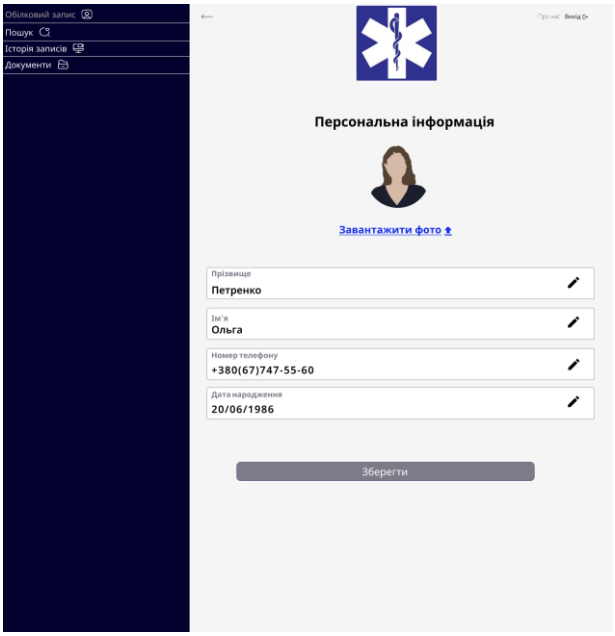


Рис. 7.2. Персональна інформація користувача

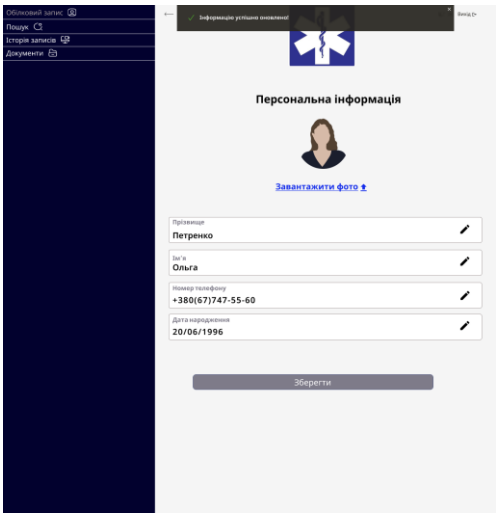


Рис. 7.3. Повідомлення про успішне оновлення персональної інформації

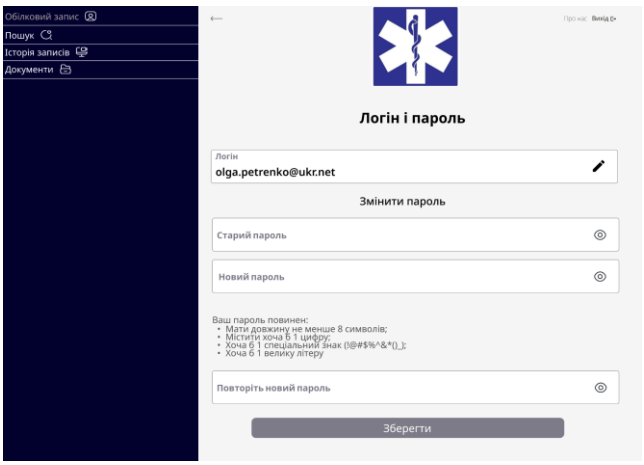


Рис. 7.4. Сторінка «Логін і пароль»

8. КЕРУВАННЯ ПРИЙОМАМИ ЛІКАРЕМ

Лікар має можливість скасувати прийом, дата та час якого ще не настали. Для цього, лікарю необхідно перейти в «Історія записів» в панелі навігації і обрати один з візитів у списку запланованих візитів (рис. 8.1).

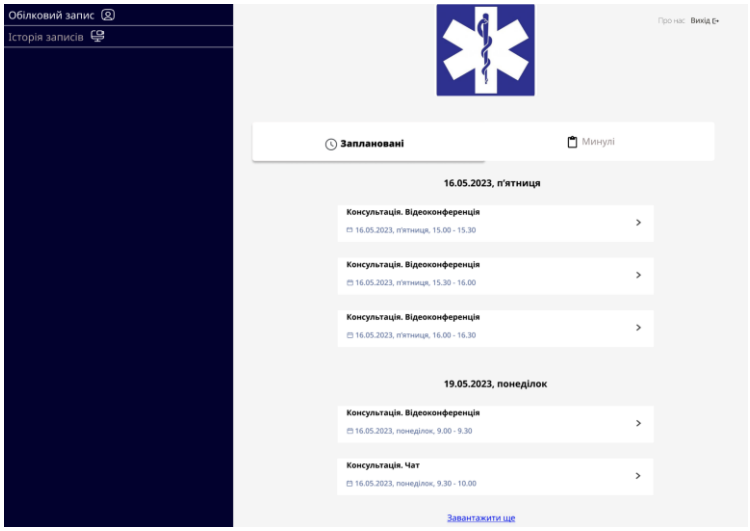


Рис 8.1. Список запланованих записів лікаря

Знизу під короткою інформацією про прийом знаходиться кнопка «Скасувати» (рис. 8.2). Після натисканні на яку відображається спливаюче вікно з полем для коментарю. Скасувати прийом без пояснення (коментарю) неможливо.

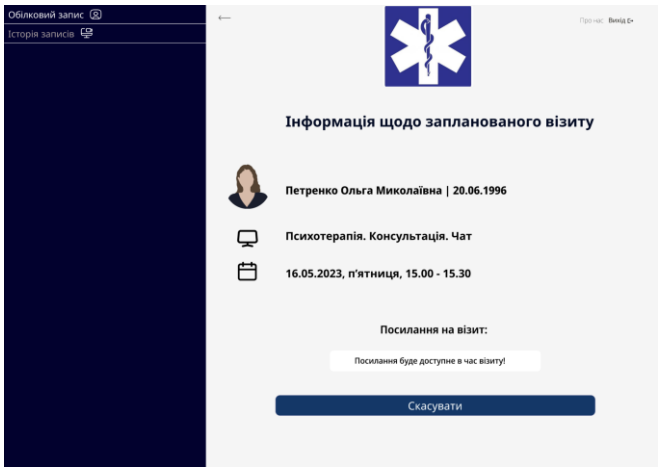


Рис. 8.2. Інформація про прийом

9. КЕРУВАННЯ ПРИЙОМАМИ АДМІНІСТРАТОРОМ

Адміністратор має змогу редагувати прийоми лише у тому випадку, коли користувач обрав функціональність «Передзвоніть мені». У адміністратора наявний список прийомів, що на розгляді. Обравши запис зі списку, адміністратору відображаються наступні кнопки. (рис. 9.1).

- Редагувати.
- Підтвердити
- Скасувати.

Після натискання на кнопку «Редагувати», відображається таблиця з з доступними датами та часом, що є доступними для лікаря (рис. 3.4).

Після натискання на кнопку «Підтвердити», сторінка з запланованими прийомами відображається, де прийом вже не у списку прийомів на розгляді.

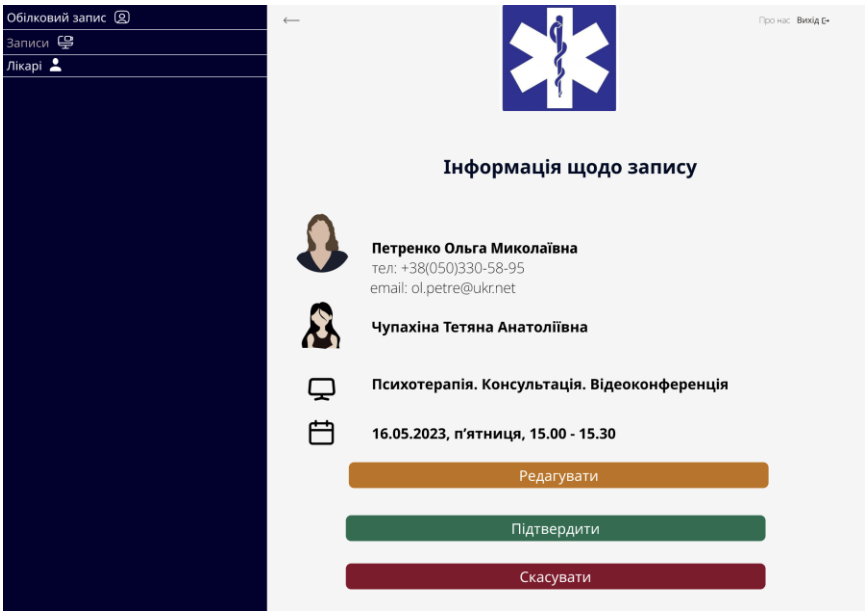


Рис. 9.1. Інформація щодо запису на розгляді

Після редагування дати та часу прийому, адміністратору слід підтвердити нову дату та час, для цього, після обрання да та часу, адміністратору слід натиснути на кнопку «Підтвердити» (рис. 9.2).

Після натискання на кнопку «Скасувати», повідомлення про успішне скасування прийому відображається на сторінці з списком прийомів.

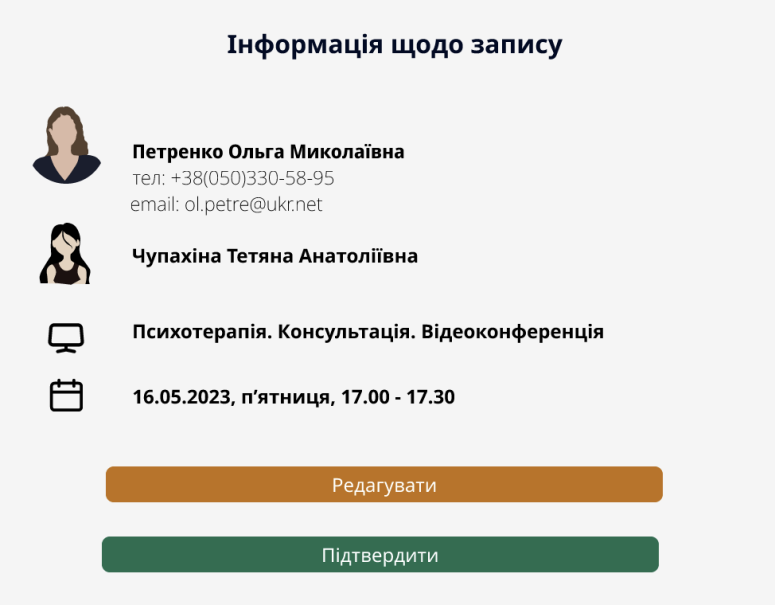


Рис. 9.2. Сторінка з оновленої інформацію прийому що слід підтвердити або редагувати

10. КЕРУВАННЯ ОБЛІКОВИМИ ЗАПИСАМИ ЛІКАРІВ

Щоб створити лікаря, адміністратору слід перейти в «Лікарі» у панелі навігації і обрати там «Додати». Після натискання на «Додати», відкривається сторінка де адміністратору необхідно крок за кроком заповнити усі необхідні данні (рис 10.1-10.3), перевірити їх та підтвердити.

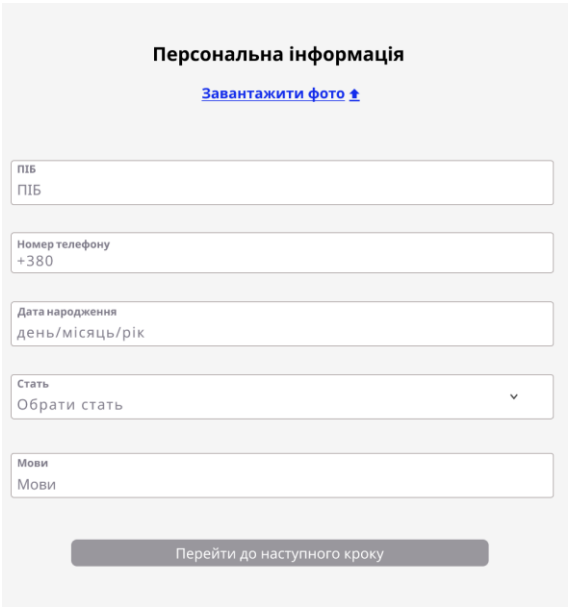


Рис. 10.1. Перший етап створення лікаря

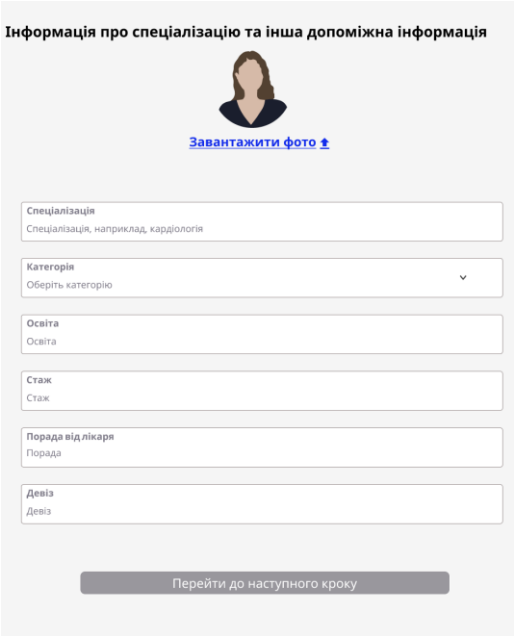


Рис. 10.2. Другий етап створення лікаря

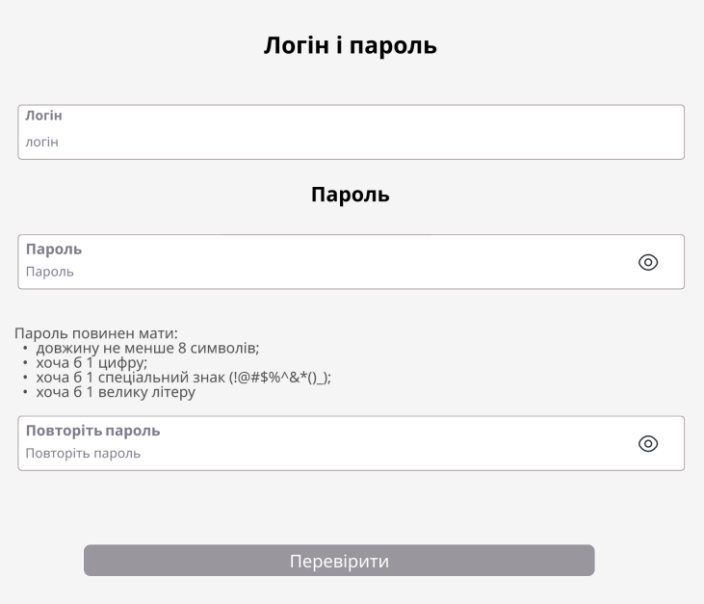


Рис. 10.3. Етап перевірки заповненої інформації про лікаря

Після перевірки адміністратором введеної інформації слід натиснути на кнопку «Створити» (рис. 10.4).

Також адміністратор має право активувати чи деактивувати обліковий запис лікаря. Для цього, необхідно натиснути на «Лікарі» в панелі навігації, обрати лікаря, у випадку, коли обліковий запис лікаря не активний , адміністратор має право лише активувати обліковий запис, натиснувши на кнопку «Активувати» (рис. 10.5). У випадку, якщо необхідно деактивувати

обліковий запис лікаря, адміністратор має натиснути на кнопку «Деактивувати» і має змогу редагувати облікованої (рис. 10.6).

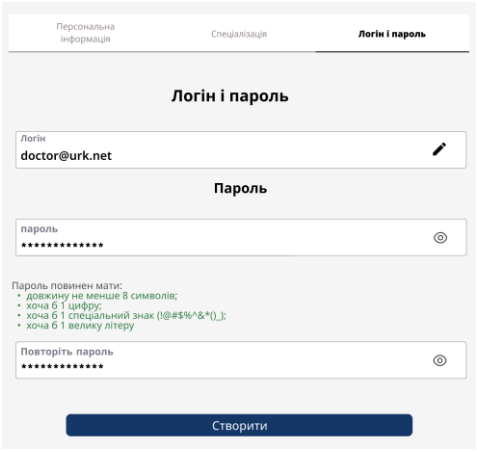


Рис. 10.4. Сторінка, на якій можливе створення облікового запису лікаря

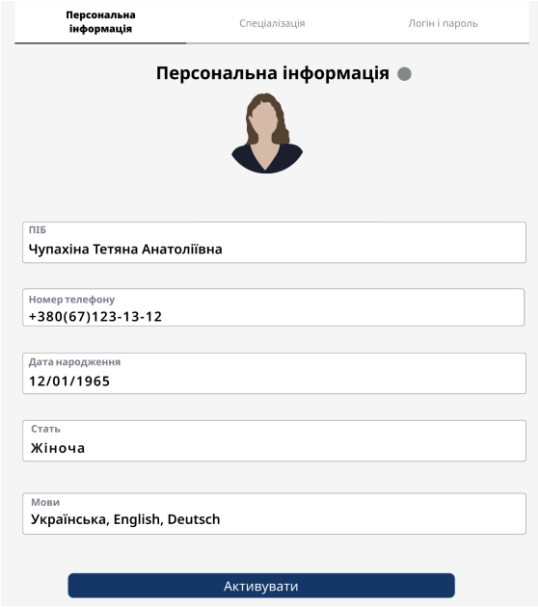


Рис. 10.5. Сторінка з активацією облікового запису лікаря

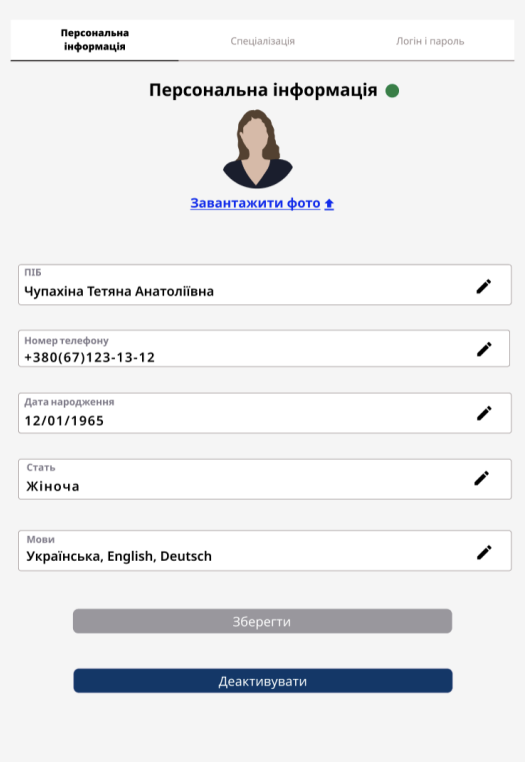


Рис. 10.6. Сторінка з деактивацією облікового запису лікаря