Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ НАДАННЯ МЕДИЧНИХ ОНЛАЙН-КОНСУЛЬТАЦІЙ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-3
Лазаренко Т.Б.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д.О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Лазаренко Тетяні Борисівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебзастосунку для надання медичних онлайн-консультацій

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, фреймворк React, платформа Node.js та середовище розробки WebStorm.

4. Перелік питань, що потрібно опрацювати в роботі
1. Проаналізувати наявні аналоги програмних рішень на ринку.
2. Дослідити сучасні технології розробки вебзастосунків.
3. Обрати оптимальний технологічний стек для реалізації проєкту.
4. Розробити вебзастосунок для медичних онлайн-консультацій.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми розробки вебзастосунку для медичних онлайн-консультацій, постановка задачі, програмна реалізація, висновки.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-05.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-13.06.26	
10	Перевірка на плагіат	14.06.26-16.06.26	
11	Рецензування	16.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26	
14	Попередній захист кваліфікаційної роботи	30.06.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Руденко Д.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 78 с., 2 табл., 19 рис., 36 джерел.

ВЕБЗАСТОСУНОК, ТЕЛЕМЕДИЦИНА, JAVASCRIPT, REACT, NESTJS, WEBRTC, SOCKET.IO, MUI.

Об’єктом роботи є процес організації та функціонування вебзастосунків для надання медичних консультацій із застосуванням технологій штучного інтелекту.

Метою роботи є розробка ефективного вебзастосунку для медичних консультацій, що використовує технології штучного інтелекту для забезпечення якісної взаємодії з користувачем, формування індивідуальних шаблонів запитів, а також гарантування безпеки та конфіденційності даних.

Під час роботи використано: методи штучного інтелекту, методи веброзробки, фреймворк React, платформу Node.js та середовище розробки WebStorm.

Практична цінність вебзастосунку полягає в підвищенні якості медичного обслуговування та забезпеченні доступності медичних консультацій для ширшого кола осіб.

Область застосування: медицина, онлайн-консультації лікарів.

WEB APPLICATION, TELEMEDICINE, JAVASCRIPT, REACT, NESTJS, WEBRTC, SOCKET.IO, MUI.

The object of the work is the process of organizing and operating web applications for providing medical consultations using artificial intelligence technologies.

The purpose of the work is to develop an effective web application for medical consultations that leverages artificial intelligence to ensure high-quality user interaction, generate individualized query templates, and guarantee data security and confidentiality.

During the work, the following methods were used: artificial intelligence, web development, the React framework, the Node.js platform, and the WebStorm development environment.

The practical value of the web application is to improve the quality of medical care and ensure the availability of medical consultations for a wider range of people.

Scope: medicine, online doctor consultations.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence (Штучний інтелект)

LaMDA – Language Model for Dialogue Applications (Мовна модель для діалогових програм)

SQL – Structured query language (Мова структурованих запитів)

WebRTC – Web Real-Time Communication (Вебспілкування в режимі реального часу

SSR – Server-Side Rendering (Рендеринг на стороні сервера)

VSCode – Visual Studio Code (Код Visual Studio)

ВСТУП

Попри те, що інтерес до питань здоров'я та добробуту людини бере свій початок ще з часів античної Греції, сфера охорони здоров'я й сьогодні продовжує активно розвиватися, демонструючи стрімке зростання. За тисячоліття людство накопичило колосальні обсяги знань про захворювання, їхні симптоми та методи лікування. Втім, навіть за наявності таких досягнень людський мозок не здатен повною мірою опрацьовувати весь масив цієї інформації.

У цьому контексті інтеграція медицини з сучасними комп'ютерними технологіями виглядає закономірним етапом подальшого розвитку галузі. Штучний інтелект уже активно застосовується для підтримки процесів діагностики, лікування та наукових досліджень. Особливістю цієї сфери є опора на великі обсяги даних, які безперервно зростають і оновлюються.

Завдяки здатності працювати з масштабними масивами інформації, штучний інтелект дедалі ширше впроваджується в медичну практику. Хоча людський мозок вирізняється унікальними можливостями критичного мислення, креативності та інтуїції, він обмежений у швидкості й обсягах обробки даних. Щодня в медицині генеруються терабайти інформації – від результатів лабораторних досліджень до електронних медичних карт пацієнтів, і ефективно опрацювати такі обсяги людині вкрай складно.

Натомість системи штучного інтелекту здатні аналізувати великі датасети в масштабах, недосяжних для людини, виявляти приховані закономірності та формувати узагальнені висновки, що мають практичне значення для діагностики та вибору оптимальної стратегії лікування. Крім того, завдяки механізмам машинного навчання такі системи можуть постійно вдосконалюватися, накопичуючи нові дані та адаптуючись до них.

У сучасних умовах технології штучного інтелекту дедалі ширше впроваджуються в медичну сферу завдяки їхній здатності швидко аналізувати, обробляти та інтерпретувати значні обсяги медичних даних.

Незважаючи на стрімкий розвиток онлайн-платформ для надання медичних консультацій, значна частина таких сервісів досі не використовує можливості штучного інтелекту для підвищення ефективності взаємодії з користувачами.

Актуальність даної роботи зумовлена розробкою вебзастосунку, який інтегрує сучасні технології штучного інтелекту для надання користувачам швидкого та зручного доступу до медичної інформації. У зв'язку зі стрімким розвитком цифрових технологій та зростанням попиту на дистанційні медичні консультації виникає потреба у створенні інструментів, здатних забезпечити оперативне отримання відповідей на запитання користувачів без необхідності тривалого пошуку інформації в різних джерелах. Використання штучного інтелекту дозволяє автоматизувати процес аналізу запитів, надавати рекомендації та сприяти підвищенню доступності медичних консультацій для широкого кола користувачів.

Запропонований вебзастосунок має низку переваг порівняно з наявними аналогами. Однією з ключових особливостей є можливість взаємодії з AI-моделлю без обов'язкової реєстрації та авторизації, що значно спрощує доступ до функціоналу системи та скорочує час, необхідний для початку роботи. Крім того, користувачам надається можливість створювати власні шаблони запитів відповідно до індивідуальних потреб, що сприяє більш ефективній та зручній взаємодії із системою. Також у застосунку передбачено набір готових шаблонів, які допомагають швидко формувати запити та отримувати необхідну інформацію навіть користувачам без досвіду роботи з системами штучного інтелекту.

Важливою перевагою розробленого рішення є орієнтація на простоту використання та доступність інтерфейсу для різних категорій користувачів. Завдяки поєднанню сучасних вебтехнологій та можливостей штучного інтелекту вебзастосунок забезпечує ефективний інструмент для отримання медичних консультацій, що відповідає сучасним тенденціям цифровізації медичної сфери та сприяє підвищенню якості взаємодії користувачів з інформаційними медичними сервісами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Розвиток штучного інтелекту розпочався ще у 1960-х роках, і сьогодні його результати стали невід'ємною частиною повсякденного життя [1-4]. Одним із найбільш помітних досягнень останнього часу є поява інтелектуальних чат-ботів, зокрема ChatGPT, який привернув значну увагу завдяки широким можливостям обробки природної мови.

У ході написання роботи було здійснено порівняльний аналіз традиційних консультацій із лікарем та онлайн-консультацій із використанням сучасних моделей на основі GPT. Результати показали, що онлайн-консультації мають низку суттєвих переваг.

Насамперед варто відзначити їхню доступність. Використання GPT-моделей дозволяє пацієнтам отримувати консультації незалежно від географічного розташування, що є особливо важливим для жителів віддалених регіонів або осіб з обмеженим доступом до медичних установ.

Ще однією важливою перевагою є зручність. Онлайн-консультації дають змогу звертатися по допомогу в будь-який час і з будь-якого місця, не витрачаючи час на дорогу до медичного закладу та очікування в чергах, що значно підвищує комфорт користувачів і оптимізує процес отримання медичної допомоги. Крім того, моделі GPT забезпечують широкий доступ до інформації. Вони здатні опрацьовувати великі обсяги медичних даних і знань, що дає змогу надавати актуальні рекомендації на основі сучасних досліджень і медичної літератури [5, 6].

Важливим аспектом є також забезпечення анонімності та конфіденційності. Онлайн-взаємодія з використанням таких систем дозволяє користувачам зберігати конфіденційність своїх медичних запитів, що є особливо значущим для обговорення чутливих тем. Окремо слід підкреслити швидкість і ефективність роботи таких систем. GPT-моделі здатні оперативно обробляти запити та надавати відповіді практично миттєво, що дозволяє користувачам швидко отримувати необхідну інформацію, особливо

в умовах обмеженого доступу до лікаря. Таким чином, враховуючи наведені переваги, онлайн-консультації з використанням моделей штучного інтелекту можуть розглядатися як ефективний і зручний інструмент для отримання медичної інформації та підтримки пацієнтів.

Варто підкреслити, що вже сьогодні системи штучного інтелекту [7,8] здатні надавати ефективні рекомендації у різних галузях медицини. Однією з ключових переваг є можливість оперативного отримання консультації без потреби в записі на прийом і особистому відвідуванні лікаря. Такі моделі мають доступ до значних обсягів даних, накопичених людством на електронних носіях, і здатні аналізувати їх у великих масштабах, обсяг яких постійно зростає експоненційно. Очікується, що точність їхніх висновків, ефективність запропонованих рішень і здатність вирішувати складні медичні завдання з часом лише підвищуватимуться.

Враховуючи відносну новизну цих технологій [9], їх активне вдосконалення фахівцями [10] може призвести до того, що в майбутньому значна частина консультаційної роботи лікарів буде виконуватися за допомогою подібних систем. Інтеграція штучного інтелекту з доступом до мільярдів наукових досліджень, книг і статей, а також можливість проведення медичних консультацій, має потенціал здійснити суттєві зміни у вітчизняній системі охорони здоров’я [11, 12], що відкриває нові можливості як для пацієнтів, так і для лікарів: перші отримують швидкий доступ до якісних рекомендацій, а другі – додатковий інструмент для підтримки прийняття рішень щодо діагностики та лікування. У перспективі це сприятиме зниженню навантаження на медичних працівників і покращенню раннього виявлення захворювань.

1.1 Основи машинного навчання

Попри те, що галузь штучного інтелекту лише нещодавно почала активно розвиватися та привертати широку увагу, уже сьогодні відповідні

технології застосовуються в діагностиці захворювань, аналізі геному та розробці лікарських засобів. Однією з ключових складових штучного інтелекту є машинне навчання, яке надає комп'ютерним системам здатність до самонавчання, тобто покращення результатів виконання конкретних завдань на основі здобутого досвіду. Важливою особливістю такого підходу є відсутність явного програмування кінцевих результатів.

Історично машинне навчання бере свій початок у 1959 році, коли Артур Самюел [13] увів відповідний термін, визначивши його як здатність комп'ютера навчатися без прямого програмування.

На сьогодні виокремлюють кілька основних типів машинного навчання, зокрема:

- навчання з учителем;
- напівкероване навчання;
- навчання без учителя;
- навчання з підкріпленням.

Кожен із цих підходів має свої характерні особливості, переваги та обмеження, що визначають доцільність їх застосування залежно від специфіки поставлених завдань.

Навчання з учителем – це один із типів машинного навчання, який можна охарактеризувати як навчання на основі прикладів. Воно ґрунтується на використанні набору даних, у якому заздалегідь відомі як вхідні, так і відповідні вихідні значення. Основною метою алгоритму є встановлення залежності між цими даними, тобто визначення того, які вихідні результати відповідають певним вхідним параметрам, що дає змогу виявити закономірності в обраному наборі даних.

Таким чином, процес навчання ґрунтується на аналізі наявних спостережень. У ході навчання модель формує прогнози, які поступово коригуються для підвищення точності. Навчання триває доти, доки результати роботи алгоритму не досягнуть прийнятного рівня якості, тобто забезпечать достатню точність прогнозування.

Навчання з частковим залученням учителя є близьким до навчання з учителем, однак має суттєву відмінність у використанні даних. У цьому підході для навчання застосовуються як розмічені, так і нерозмічені дані. Це означає, що модель навчається на одній підмножині даних, для якої відомі правильні відповіді, і водночас використовує іншу підмножину без міток.

У такому випадку «учитель» надає лише часткову інформацію, тобто мітки доступні не для всіх прикладів. Модель, у свою чергу, намагається максимально ефективно використати нерозмічені дані для виявлення загальних закономірностей, що дозволяє покращити якість прогнозування та підвищити загальну ефективність навчання.

Навчання без учителя – це тип машинного навчання, у якому алгоритм самостійно досліджує, аналізує та інтерпретує дані з метою виявлення прихованих закономірностей. У цьому підході відсутні заздалегідь визначені правильні відповіді, тому модель самостійно встановлює зв’язки та визначає кореляції на основі наявних даних.

У процесі такого навчання, як правило, використовують великі обсяги інформації, на основі яких формують узагальнення та висновки. Основне завдання алгоритму полягає в тому, щоб виявити структуру даних і певним чином їх впорядкувати, наприклад, шляхом кластеризації або зменшення розмірності. Водночас варто зазначити, що збільшення обсягу даних зазвичай сприяє підвищенню точності та надійності отриманих результатів.

Навчання з підкріпленням є одним із напрямів машинного навчання, у межах якого алгоритм взаємодіє з навколишнім середовищем, маючи визначений набір можливих дій і параметрів. На початковому етапі моделі задаються базові правила поведінки, після чого вона самостійно досліджує простір можливих рішень, оцінюючи наслідки кожної дії.

Ключовим принципом даного підходу є механізм «спроб і помилок», за якого система отримує зворотний зв’язок у вигляді винагороди або штрафу залежно від якості прийнятого рішення. На основі цього зворотного сигналу алгоритм поступово коригує свою стратегію поведінки, підсилюючи

ефективні дії та зменшуючи ймовірність використання неефективних або помилкових рішень.

У процесі навчання модель накопичує досвід взаємодії із середовищем та адаптується до змінних умов, що дозволяє їй поступово підвищувати якість прийнятих рішень. Завдяки цьому досягається оптимізація поведінки системи в межах поставленої задачі та підвищується ефективність її роботи з плином часу.

У результаті аналізу основних підходів машинного навчання було визначено специфіку функціонування кожного з них, а також їхні ключові відмінності, що дозволяє краще розуміти сферу застосування різних методів та обґрунтовано обирати відповідний підхід для вирішення конкретних задач.

1.2 Аналіз аналогів

Аналіз наявних на ринку аналогів є одним із ключових етапів проєктування вебзастосунку будь-якого призначення, оскільки він дозволяє комплексно оцінити сучасний стан предметної області, визначити функціональні можливості наявних рішень, а також виявити їхні сильні й слабкі сторони. Проведення такого аналізу створює основу для формування обґрунтованих вимог до майбутнього програмного продукту, що сприяє підвищенню його якості та конкурентоспроможності.

Окрім цього, дослідження аналогічних систем дає можливість краще зрозуміти очікування та потреби кінцевих користувачів, а також визначити найбільш затребувані функціональні можливості, що дозволяє сформувати більш точне уявлення про потенційний попит на розроблюваний продукт і адаптувати його функціонал відповідно до реальних сценаріїв використання.

Результати такого аналізу також мають значний вплив на вибір технологічного стеку та інструментів розробки, оскільки дозволяють врахувати перевірені підходи, що використовуються в подібних рішеннях, і

уникнути типових помилок. У сукупності це забезпечує більш ефективний процес розробки та підвищує загальну якість кінцевого вебзастосунку.

1.2.1 Chat GPT

Одним із найбільш очевидних аналогів є ChatGPT – застосунок, розроблений компанією OpenAI [5], який за короткий час здобув широку популярність у всьому світі. Його успіх зумовлений здатністю надавати швидкі та змістовні відповіді на запити користувачів у різних галузях, зокрема й у сфері медицини. Інтерфейс цього застосунку проілюстровано на рисунку 1.1.

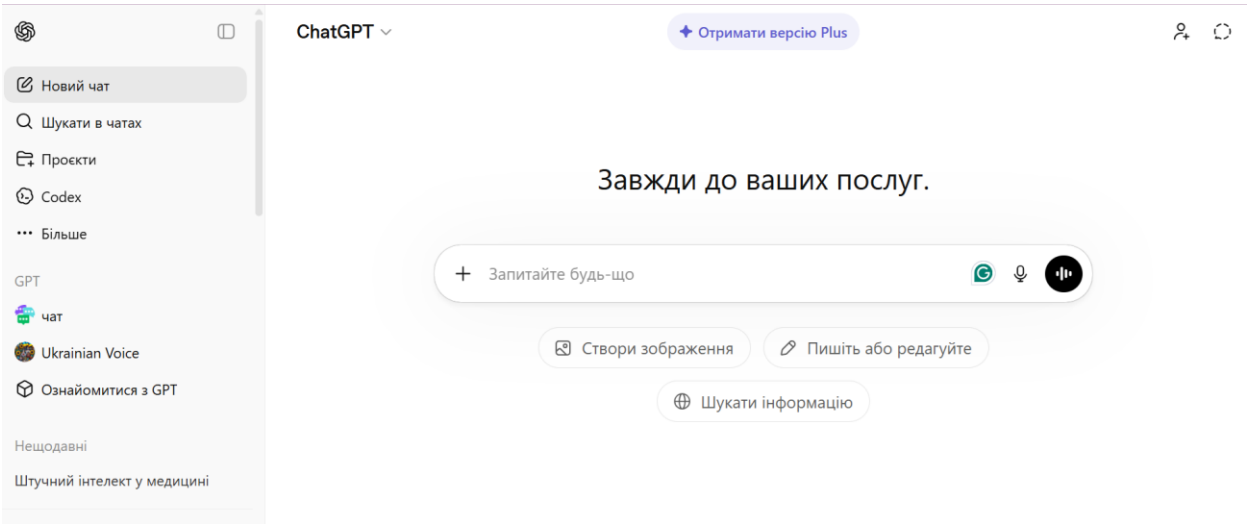

Рисунок 1.1 – Інтерфейс застосунку Chat GPT

На рисунку 1.1 представлено інтерфейс безкоштовної версії ChatGPT. Як зазначено під полем введення запиту, ця модель у цьому режимі використовується як інструмент для дослідження можливостей технологій GPT. Водночас користувач має можливість оформити платну підписку, яка відкриває доступ до розширеного функціоналу та додаткових переваг. Відповідні можливості платної версії наведено на рисунку 1.2.

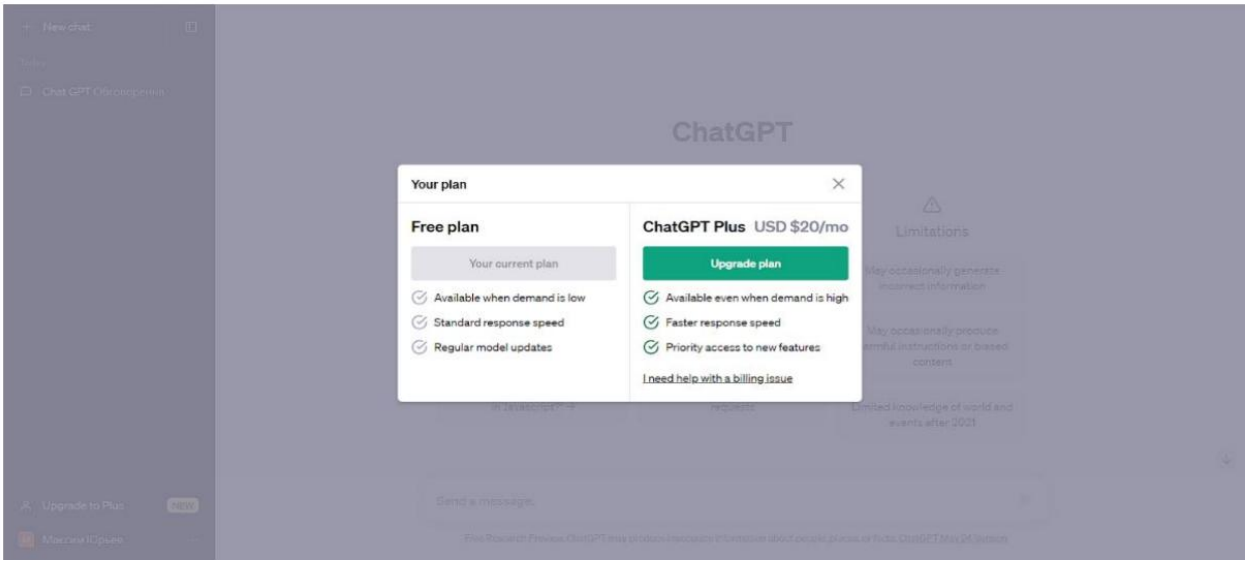

Рисунок 1.2 – Можливості платної версії Chat GPT

До переваг платної версії ChatGPT належать розширені функціональні можливості та покращені умови використання сервісу, що забезпечують більш стабільну, швидку й ефективну взаємодію з моделлю штучного інтелекту [1, 2, 5].

Однією з ключових переваг є пріоритетний доступ до моделі під час високого навантаження. У моменти, коли кількість користувачів суттєво зростає, безкоштовні облікові записи можуть стикатися з обмеженнями доступу або зниженням швидкості роботи сервісу. Натомість користувачі платної версії отримують пріоритет у черзі запитів, що дозволяє їм продовжувати роботу без затримок і перебоїв. Це особливо важливо для професійного використання, коли стабільність доступу до інструменту є критичним фактором.

Ще однією суттєвою перевагою є скорочення часу генерації відповідей. Завдяки оптимізованій обробці запитів та виділеним обчислювальним ресурсам платна версія забезпечує швидше формування відповідей порівняно з базовим доступом. Це дозволяє значно підвищити продуктивність користувача, особливо під час виконання великих обсягів роботи, таких як написання текстів, програмування, аналіз даних або підготовка навчальних

матеріалів. У результаті взаємодія з системою стає більш динамічною та ефективною.

Також важливою перевагою є ранній доступ до нових функціональних можливостей моделі. Користувачі платної версії першими отримують можливість тестувати оновлення, покращені алгоритми обробки запитів, нові інструменти та додаткові функції. Це дозволяє не лише раніше використовувати нові технологічні можливості, а й адаптувати їх до власних потреб ще до офіційного масового релізу. У багатьох випадках це дає конкурентну перевагу, особливо в професійній та освітній діяльності.

Таким чином, платна версія ChatGPT забезпечує підвищений рівень стабільності, швидкодії та функціональної насиченості, що робить її більш ефективним інструментом для користувачів, які працюють із великими обсягами інформації або потребують гарантованої доступності сервісу.

Перший аспект є особливо важливим, оскільки безкоштовна версія може бути недоступною у разі великої кількості одночасних запитів від користувачів. Крім того, підвищена швидкість формування відповідей у платній версії є суттєвою перевагою, особливо у випадках, коли необхідно отримати інформацію у стислі терміни.

Серед додаткових можливостей платної версії варто також відзначити доступ до моделі GPT-4, яка є більш потужною у задачах генерації тексту. Вона навчається на більш об’ємній та актуальній вибірці даних, однак має певні обмеження, зокрема нижчу швидкість обробки запитів, а також ліміт на кількість повідомлень – до 35 запитів кожні три години для одного користувача.

Вебзастосунок OpenAI у безкоштовній версії надає базовий набір функціональних можливостей, достатніх для повсякденного використання системи штучного інтелекту в режимі діалогу. Основним інструментом взаємодії є чат-інтерфейс, який дозволяє користувачеві формулювати запити природною мовою та отримувати текстові відповіді від моделі. Такий формат

забезпечує інтуїтивно зрозумілу комунікацію та не потребує спеціальних технічних знань для роботи із системою.

Окрім базової взаємодії з моделлю у форматі чату, користувач має можливість створювати декілька окремих чатів. Це дозволяє структурувати робочі діалоги за темами, проєктами або завданнями, що значно підвищує зручність використання сервісу та сприяє кращій організації інформації. Кожен чат зберігає свою історію взаємодії, що дає змогу повертатися до попередніх обговорень у будь-який момент.

Додатково реалізовано функцію поширення посилання на чат, яка дозволяє користувачам ділитися результатами взаємодії з іншими користувачами. Це може бути корисним у навчальних, робочих або дослідницьких цілях, коли необхідно продемонструвати хід розв’язання задачі або передати сформовану інформацію колегам. Таким чином, навіть у безкоштовній версії сервіс забезпечує достатній рівень функціональності для базового використання та обміну результатами роботи.

Водночас слід зазначити, що відсутня функція створення шаблонів запитів до моделі, яка могла б значно спростити та пришвидшити роботу користувачів із сервісом.

1.2.2 Bard AI

Наступним аналогом розроблюваного вебзастосунку є Bard AI [14], створений компанією Google, яка також є розробником одного з найпопулярніших веббраузерів у світі – Google Chrome. Даний сервіс є чат-орієнтованою системою штучного інтелекту, призначеною для взаємодії з користувачем у форматі діалогу та генерації відповідей на основі великих мовних моделей.

Bard AI побудований на основі мовної моделі LaMDA (Language Model for Dialogue Applications) [15], розробленої компанією Google спеціально для

обробки природної мови та підтримки діалогових сценаріїв. Використання цієї моделі дозволяє системі формувати контекстно узгоджені відповіді, враховувати попередні повідомлення користувача та підтримувати більш природну взаємодію в режимі багатоетапного діалогу.

Інтерфейс застосунку реалізований у вигляді простого та мінімалістичного чату, що забезпечує зручність використання та швидкий доступ до основної функціональності. Загальний вигляд інтерфейсу цього застосунку наведено на рисунку 1.3.

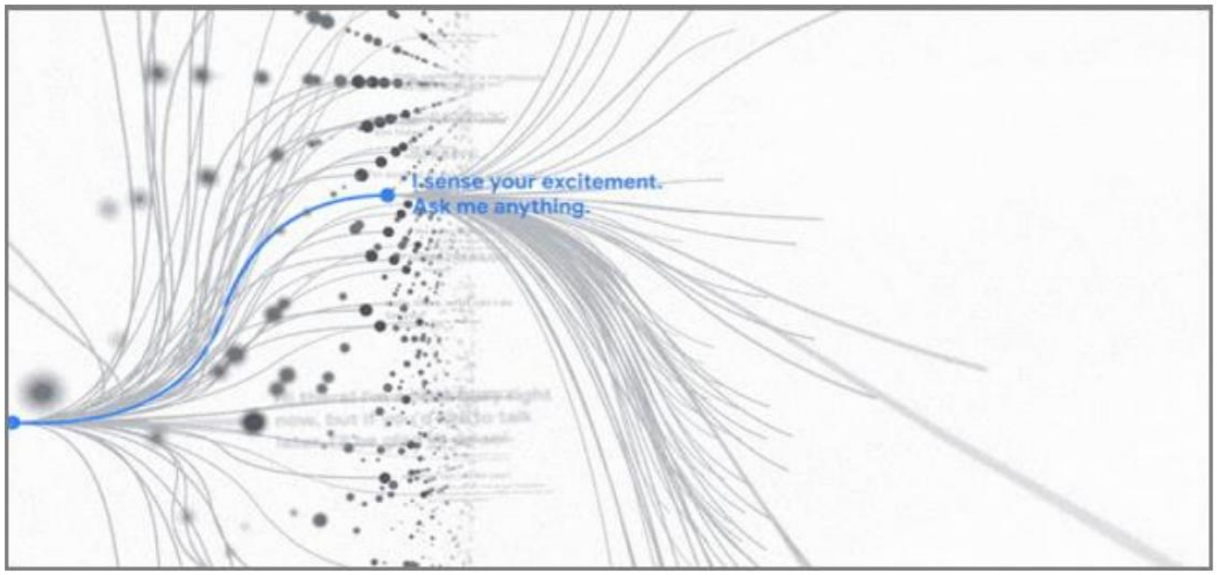

Рисунок 1.3 – LaMDA моделі від Google

До основних можливостей моделі LaMDA від Google можна віднести здатність розуміти складні запити користувачів, брати участь в інтелектуальних діалогах на широкий спектр тем, а також виконувати класифікаційні завдання. Це дозволяє моделі ефективно працювати з контекстом розмови та підтримувати більш природну взаємодію з користувачем. Важливою особливістю LaMDA є потенційно актуальніші дані, оскільки модель може регулярно оновлюватися та доповнюватися. Для порівняння, у безкоштовних версіях GPT існує обмеження щодо актуальності навчальних даних (зокрема до 2019 року). Водночас слід зазначити, що

LaMDA наразі перебуває на етапі активного розвитку, тому остаточні сфери її найбільш ефективного застосування ще не визначені повністю.

1.2.3 Bing AI

Наступною розглянутою альтернативою є розширена пошукова система для браузера Microsoft Edge від компанії Microsoft – Bing AI. Bing AI було розроблено з метою розширення можливостей пошуку та підвищення його швидкості, точності й ефективності [16]. Зазначена система побудована на основі технологій штучного інтелекту, створених компанією OpenAI, які забезпечують більш розвинену функціональність порівняно з попередніми моделями, зокрема GPT-3.5.

Bing AI використовує ключові напрацювання попередніх моделей, щоб покращити якість пошукових результатів і загальний користувацький досвід в екосистемі браузера Microsoft Edge. Система вже доступна для використання користувачами актуальних версій Microsoft Edge, що проілюстровано на рисунку 1.4.

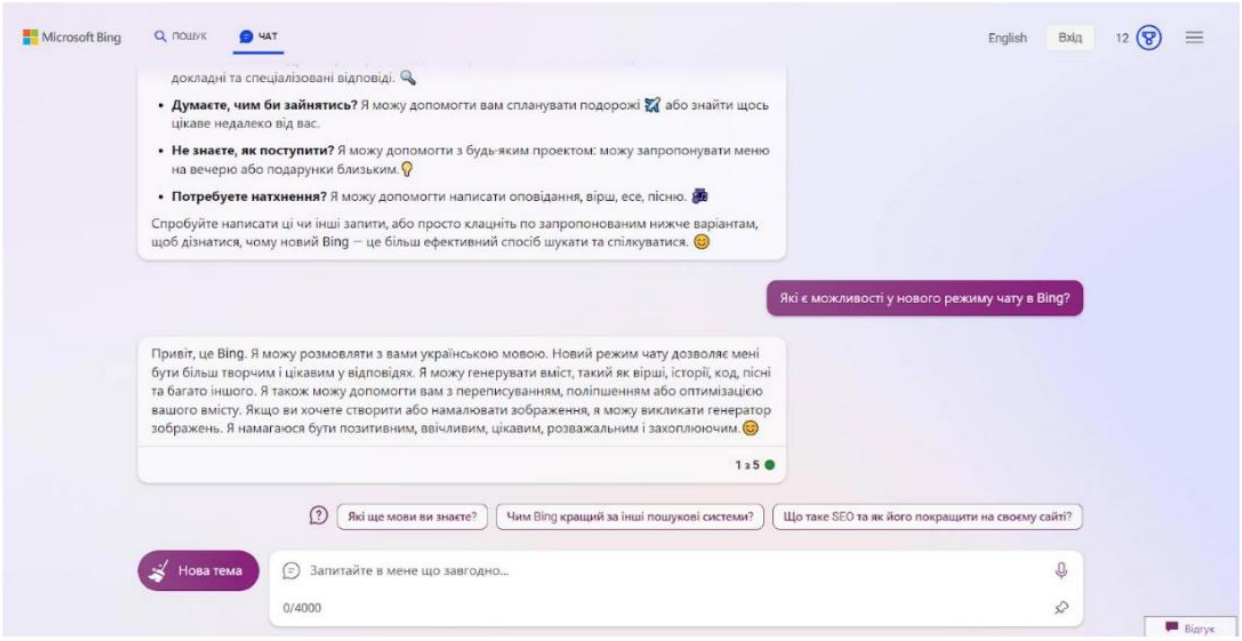

Рисунок 1.4 – Інтерфейс Bing AI

Bing AI, так само, як і Bard AI від Google, підтримує функцію голосового введення запитів. Водночас він має низку додаткових переваг, які відрізняють його від аналогів. Зокрема, однією з ключових особливостей є можливість вибору режиму роботи моделі, серед яких «Креативніше», «Урівноваженіше», «Точніше».

Кожен із цих режимів визначає стиль формування відповідей та рівень їх деталізації, що дозволяє користувачу адаптувати поведінку системи відповідно до конкретного типу запиту й очікуваного результату.

Також важливою функціональною особливістю Bing AI є можливість вибору запропонованих системою варіантів запитів. Ці підказки автоматично формуються та змінюються залежно від теми і контексту діалогу, що дозволяє користувачеві швидше формулювати запити та ефективніше взаємодіяти з системою. Bing AI також має важливу перевагу, яка полягає в автоматичному наданні посилань на джерела інформації безпосередньо у відповідях. Це дозволяє користувачеві одразу перевірити походження даних, використаних для формування відповіді, та за потреби звернутися до першоджерел для детальнішого ознайомлення з матеріалом.

Водночас, попри потужні можливості Bing AI, слід зазначити, що в першу чергу він позиціонується як інструмент для розширення функціональності пошуку в мережі Інтернет, що зумовлює певні обмеження його використання, зокрема необхідність авторизації для доступу до повного функціоналу. Крім того, система все ще перебуває на етапі активного розвитку, що означає можливу змінність її можливостей і поведінки в процесі подальших оновлень.

1.3 Постановка задачі

Штучний інтелект сьогодні набуває дедалі більшого значення у сфері медицини, насамперед завдяки своїй здатності обробляти та аналізувати

значні обсяги медичних даних. Водночас, попри наявність великої кількості вебзастосунків для онлайн-консультацій, значна частина з них не використовує повною мірою можливості сучасних моделей штучного інтелекту.

Об’єктом роботи є процес організації та функціонування вебзастосунків для надання медичних консультацій із застосуванням технологій штучного інтелекту.

Метою роботи є розробка ефективного вебзастосунку для медичних консультацій, що використовує технології штучного інтелекту для забезпечення якісної взаємодії з користувачем, формування індивідуальних шаблонів запитів, а також гарантування безпеки та конфіденційності даних.

Для досягнення мети необхідно вирішити такі завдання:

- проаналізувати існуючі аналоги програмних рішень на ринку;
- опрацювати теоретичні основи теми роботи;
- дослідити сучасні технології розробки вебзастосунків;
- обрати оптимальний технологічний стек для реалізації проєкту;
- розробити вебзастосунок для медичних онлайн-консультацій.

2 ОГЛЯД МЕТОДІВ ТА АЛГОРИТМІВ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКА

2.1 Аналіз сучасних моделей комунікацій: дані, варіативність та динаміка

Функціонування будь-якого програмного продукту нерозривно пов’язане з опрацюванням інформації. Передача даних між окремими компонентами системи здійснюється безперервно, забезпечуючи узгоджену роботу всіх модулів. Для гарантування надійного зберігання, швидкого доступу та коректної обробки інформаційних ресурсів у таких системах застосовуються бази даних.

База даних являє собою впорядковану сукупність структурованих відомостей, що, як правило, зберігаються в електронному вигляді в межах комп’ютерної системи. Керування цими даними здійснюється за допомогою системи управління базами даних, яка забезпечує створення, модифікацію, пошук, оновлення та контроль інформації. Сукупність самих даних, системи керування та прикладних програм, що взаємодіють із ними, формує єдину систему баз даних.

У сучасних інформаційних системах найчастіше використовуються реляційні бази даних, у яких інформація організовується у вигляді таблиць, що складаються з рядків і стовпців. Такий підхід значно підвищує ефективність виконання операцій пошуку, фільтрації та аналізу даних. Для взаємодії з подібними структурами широко застосовується мова структурованих запитів SQL, що дозволяє здійснювати введення, вибірку, редагування та адміністрування інформації.

У розробленому програмному вебзастосунку реалізовано моделі комунікаційної взаємодії, які передбачають організацію безпосереднього відеозв’язку між користувачами, а також обмін текстовими повідомленнями в режимі реального часу, що зумовлює необхідність використання гнучкої та

оптимізованої структури бази даних для зберігання як персональних відомостей, так і даних, пов’язаних із консультаційною сесією та процесом комунікації.

Спроєктована модель бази даних включає такі основні сутності:

- користувач;
- консультація;
- підключення;
- повідомлення.

Сутність «Користувач» призначена для зберігання персональної інформації про зареєстрованих учасників системи. До її основних атрибутів належать електронна адреса, роль у системі, ім’я, прізвище та дата народження. Введення зазначених даних здійснюється за допомогою відповідних форм графічного інтерфейсу користувача, після чого інформація передається до бази даних для подальшого збереження та використання.

Сутність «Консультація» акумулює інформацію щодо проведеної взаємодії між пацієнтом і лікарем. До її складу входять відомості про пацієнта, лікаря, тривалість сеансу з’єднання, а також медичний висновок, який формується лікарем після завершення опитування та аналізу звернення. При цьому більшість зазначених даних, окрім заключення спеціаліста, генеруються та заносяться до бази автоматично після завершення консультаційного виклику.

Сутність «Підключення» містить службову інформацію, необхідну для коректної ідентифікації учасників у мережі зв’язку. Основними її атрибутами є унікальний ідентифікаційний номер підключення та ідентифікаційний номер користувача, з яким асоціюється відповідне мережеве з’єднання. Використання цієї сутності забезпечує можливість відстеження активних сеансів та коректне встановлення взаємодії між користувачами системи.

Останньою сутністю моделі є «Повідомлення», яка відповідає за збереження даних текстової комунікації між користувачами. У ній фіксується інформація про відправника та отримувача повідомлення, дата і точний час

його надсилання, а також текстовий вміст повідомлення. Така структура дозволяє організувати історію листування та забезпечити оперативний доступ до обміну повідомленнями в режимі реального часу.

Діаграму бази даних зображено на рисунку 2.1.

Рисунок 2.1 – Діаграма бази даних

Важливим етапом розвитку сучасних систем став перехід до використання технології WebRTC [17], яка реалізує метод прямого з’єднання (peer-to-peer). Архітектурна модель такої взаємодії представлена на рисунку 2.2.

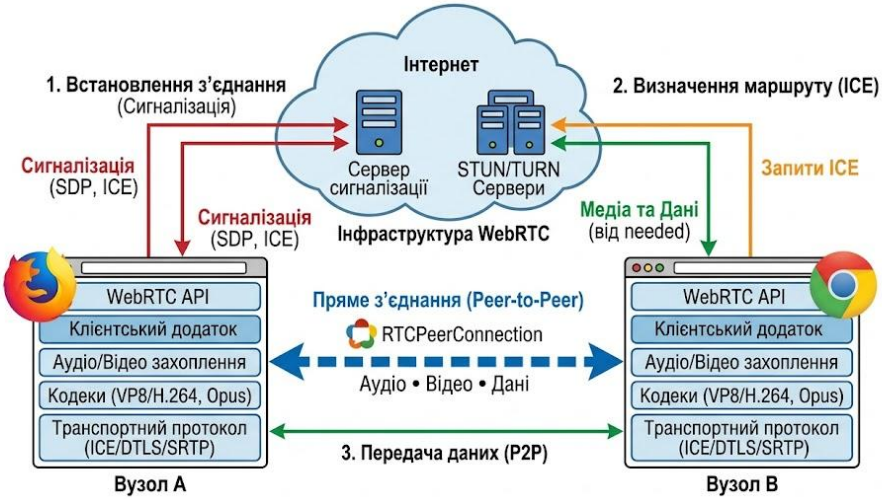

Рисунок 2.2 – Модель peer to peer з’єднання

Однорангова мережа (P2P) – це специфічна IT-інфраструктура, що забезпечує пряму взаємодію між двома або більше обчислювальними системами. Головною особливістю такого підходу є можливість спільного використання ресурсів без залучення виділеного сервера чи спеціалізованого серверного ПЗ. Формування P2P-сегмента можливе як через фізичне об’єднання пристроїв, так і шляхом створення віртуального мережевого простору, де кожен вузол здатен одночасно виконувати ролі клієнта та сервера.

WebRTC (Web Real-Time Communication) – це сучасна технологія, що надає можливість вебзастосункам здійснювати захоплення, передачу та прийом аудіо- й відеопотоків, а також організовувати обмін довільними даними між браузерами в режимі реального часу без необхідності використання проміжних серверів-посередників. Принцип функціонування цієї технології наведено на рисунку 2.3.

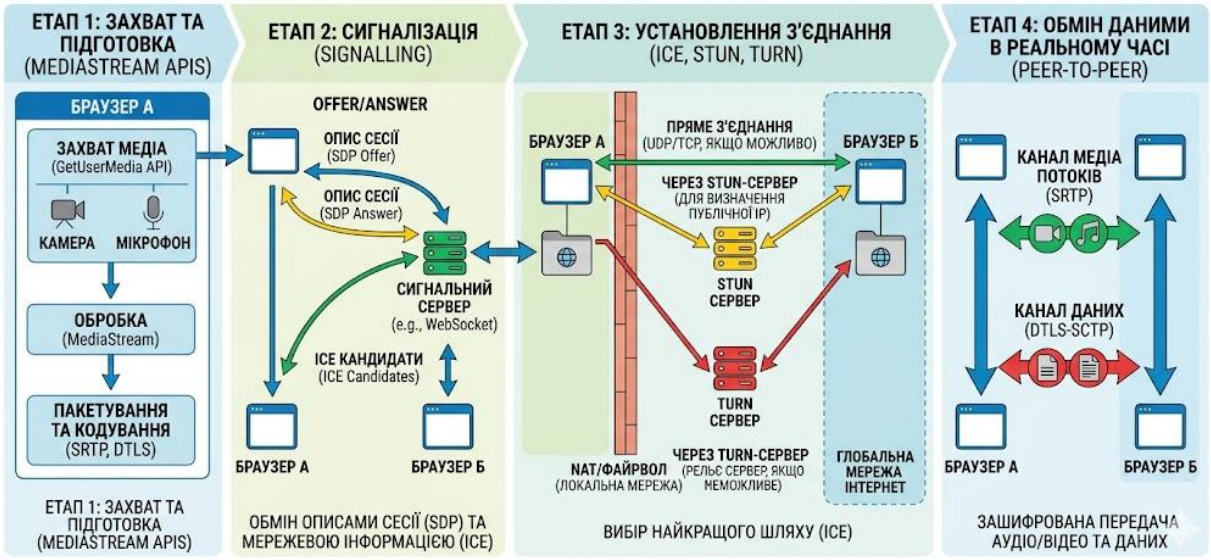

Рисунок 2.3 – Схема роботи технології WebRTC

Комплекс стандартів WebRTC забезпечує реалізацію однорангової взаємодії між користувачами, що дозволяє проводити телеконференції, голосові та відеодзвінки, а також передавати службову чи текстову інформацію напряму між клієнтськими пристроями. Важливою перевагою

даної технології є відсутність потреби у встановленні додаткових плагінів, розширень або стороннього програмного забезпечення, оскільки підтримка WebRTC інтегрована безпосередньо в сучасні веббраузери.

Архітектурно WebRTC включає набір взаємопов'язаних програмних інтерфейсів (API) та мережевих протоколів, які спільно забезпечують встановлення з'єднання, передачу мультимедійних потоків, синхронізацію учасників сеансу та підтримку стабільного обміну даними в реальному часі. Саме поєднання цих компонентів робить технологію ефективним інструментом для реалізації систем дистанційної комунікації у вебсередовищі.

2.2 Розробка алгоритмів модуля відеозв'язку

Алгоритм є впорядкованою послідовністю команд або інструкцій, спрямованих на розв'язання певної задачі чи виконання визначеної операції. Як приклад алгоритму можна розглядати кулінарний рецепт, у якому покроково описано процес приготування страви. Аналогічно, будь-який комп'ютерний або цифровий пристрій функціонує на основі сукупності алгоритмів, реалізованих у вигляді програмних або апаратних засобів.

Однією з ключових функціональних можливостей розробленого вебдодатку є забезпечення аудіо- та відеозв'язку між користувачами системи. Для реалізації зазначеного функціоналу було обрано технологію WebRTC, яка дозволяє організувати передавання мультимедійних потоків у режимі реального часу без використання стороннього програмного забезпечення.

Для створення модуля відеозв'язку необхідним етапом є розробка загального алгоритму його функціонування. Даний алгоритм включає кілька послідовних етапів: встановлення з'єднання із серверною частиною за допомогою технології WebSocket, перевірку успішності підключення, обмін мережевими адресами між користувачами, ініціалізацію безпосереднього

з’єднання між браузерами учасників сеансу та подальше відображення аудіо- і відеопотоків на екрані.

WebSocket є протоколом постійного двостороннього з’єднання між клієнтом і сервером, який функціонує поверх одного TCP/IP-з’єднання. На відміну від класичної моделі HTTP-запитів, де кожна взаємодія ініціюється окремо, WebSocket підтримує безперервний канал зв’язку, що дозволяє сторонам надсилати повідомлення одна одній у будь-який момент часу [18].

Фактично протокол WebSocket спрощує та пришвидшує передавання службових повідомлень між клієнтською та серверною частинами застосунку. Його використання забезпечує передачу даних у режимі реального часу з високою надійністю, успадкованою від транспортного протоколу TCP. Початково WebSocket використовує механізм HTTP для встановлення сеансу, однак після отримання підтвердження переходить у режим постійного TCP-з’єднання, яке надалі використовується для швидкого двостороннього обміну повідомленнями.

Завдяки таким властивостям технологія WebSocket є ефективним рішенням для побудови інтерактивних вебзастосунків, що потребують миттєвого реагування системи, зокрема чатів, систем сповіщень, онлайн-конференцій та сервісів відеозв’язку, без застосування механізму тривалого опитування сервера. Блок-схему загального алгоритму функціонування модуля відеозв’язку наведено на рисунку 2.4.

На початковому етапі роботи модуль відеозв’язку встановлює з’єднання із серверною частиною системи за допомогою протоколу HTTP. Сервер виконує функції координації процесу підключення користувачів, обробки службових запитів і передачі необхідних даних для організації сеансу зв’язку. Після успішного встановлення з’єднання клієнтський застосунок проходить процедуру обміну службовою інформацією із сервером, у результаті чого отримує мережеву адресу браузера співрозмовника та інші параметри, необхідні для подальшої взаємодії. Далі система виконує налаштування каналів передавання мультимедійних даних.

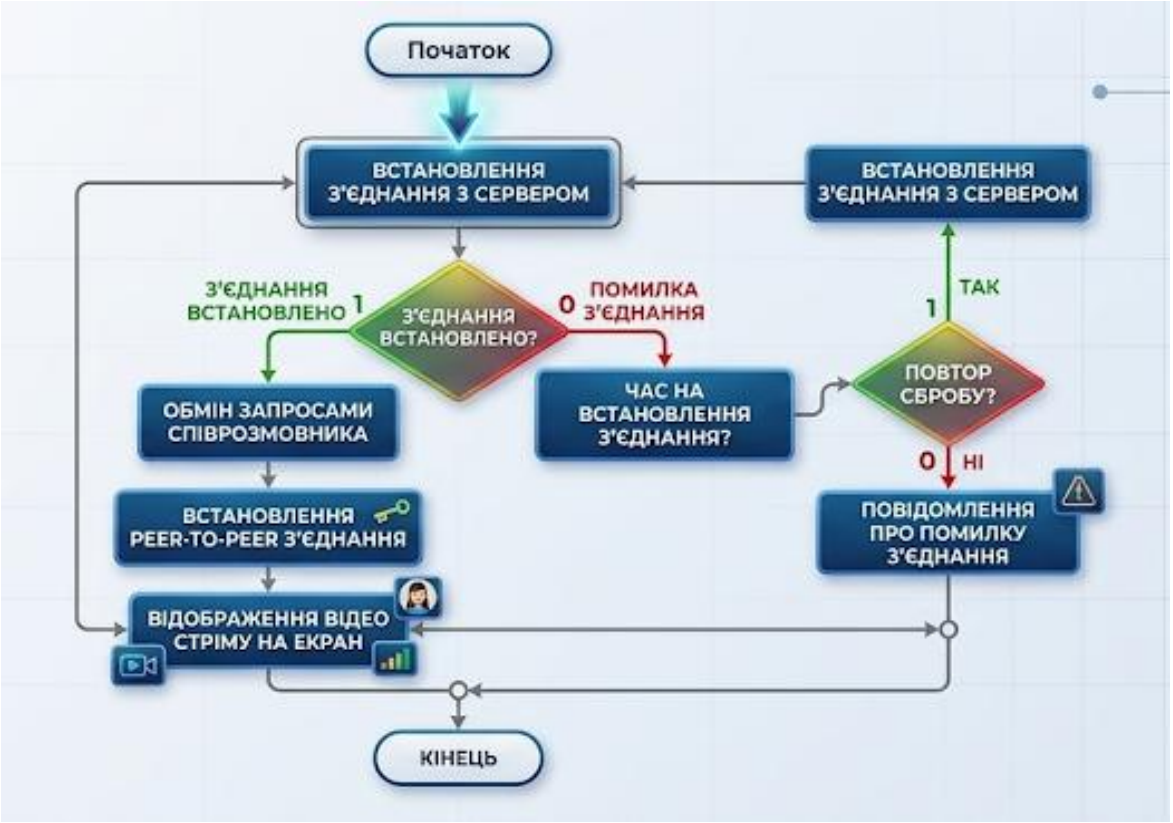

Рисунок 2.4 – Блок-схема алгоритму загальної роботи модуля відеозв’язку

Після отримання всіх необхідних службових параметрів через протокол WebSocket надсилається повідомлення про ініціювання виклику. Сервер, у свою чергу, очікує на відповідь іншого користувача та обробляє отримане рішення щодо прийняття або відхилення дзвінка.

У разі позитивної відповіді здійснюється встановлення однорангового peer-to-peer-з’єднання між браузерами учасників комунікації за допомогою технології WebRTC. Результатом такого з’єднання є безпосередня передача аудіо- та відеопотоків між користувачами з подальшим відображенням отриманого мультимедійного сигналу на екрані.

2.3 Розробка алгоритмів функціонування модуля текстових повідомлень

Важливою складовою функціоналу розробленого вебзастосунку є модуль обміну текстовими повідомленнями, який забезпечує оперативну

комунікацію між користувачами під час консультаційної взаємодії. Реалізація даного модуля потребує використання технологій, здатних забезпечувати швидке надсилання, приймання та синхронізацію повідомлень у режимі реального часу. Для цього було проаналізовано кілька сучасних підходів, зокрема REST API, Firebase Cloud Messaging та Socket.IO.

REST API [19] – це архітектурний стиль організації взаємодії між клієнтською та серверною частинами програмного забезпечення. На відміну від окремого протоколу передачі даних, REST визначає набір обмежень і принципів, за якими здійснюється обмін інформацією. Взаємодія в даному випадку здійснюється через програмний інтерфейс API, який регламентує формат запитів від клієнта та структуру відповідей сервера. Таким чином, API виступає своєрідним договором між постачальником і споживачем інформації, визначаючи правила передачі даних між програмними компонентами.

Разом із тим використання лише REST API не є достатньо ефективним для організації чату в реальному часі, оскільки така модель передбачає ініціювання кожного запиту клієнтом, що може призводити до затримок під час доставки повідомлень.

Іншим розглянутим підходом є Firebase Cloud Messaging [20] – хмарний сервіс для надсилання push-сповіщень і повідомлень у мобільних та вебзастосунках. Ця технологія забезпечує доставку повідомлень від серверної частини до кінцевих користувачів через інфраструктуру Firebase. Вона є ефективною для реалізації систем сповіщень, однак її основне призначення полягає саме в передачі нотифікацій, а не в підтримці постійного інтерактивного діалогу між двома співрозмовниками.

Найбільш доцільним рішенням для побудови модуля текстових повідомлень виявилася бібліотека Socket.IO, що працює на основі технології вебсокетів. Вона забезпечує двосторонній безперервний канал обміну даними між клієнтом і сервером, дозволяючи миттєво передавати повідомлення всім учасникам сеансу.

На відміну від класичних WebSocket-з’єднань, Socket.IO надає розширені можливості для керування підключеннями. Зокрема, бібліотека підтримує автоматичне перепідключення у випадку втрати зв’язку, сумісність із проксі-серверами та балансувальниками навантаження, а також значно спрощує реалізацію широкомовної передачі повідомлень усім активним клієнтам, що особливо важливо для чат-систем, де необхідно швидко інформувати співрозмовників про надходження нових повідомлень, зміну статусу користувача або підключення нового учасника.

Таким чином, з огляду на високу швидкодію, стабільність з’єднання та простоту інтеграції з серверною платформою, для реалізації модуля текстових повідомлень у розробленому застосунку було обрано саме бібліотеку Socket.IO.

Для проведення порівняльного аналізу методів створення модуля текстових повідомлень і вибору оптимальної технології його реалізації складено таблицю 2.1.

Таблиця 2.1 – Порівняння методів реалізації модуля текстових повідомлень

Параметр / Назва методу	**REST API**	**FCM (Firebase Cloud Messaging)**	**Socket.IO**
Продуктивність	-	+	+
Можливість відправляти повідомлення декільком користувачам одночасно	-	+	+
Простота впровадження	+	-	+
Можливість отримувати сповіщення	-	+	+
Можливість передачі медіафайлів	+	-	+
Сумарний коефіцієнт	2	3	5

За результатами проведеного порівняльного аналізу найбільш ефективним виявився підхід, заснований на використанні бібліотеки Socket.IO, оскільки його сумарний коефіцієнт переважає показники інших розглянутих технологій. Зокрема, ефективність даного рішення є вищою порівняно з REST API на 60%, а порівняно з Firebase Cloud Messaging – на 20%.

Перед безпосередньою реалізацією програмного модуля важливим етапом є побудова загального алгоритму його функціонування. Це дає можливість повноцінно визначити послідовність обробки даних, передбачити можливі сценарії взаємодії користувачів та описати всі ключові стани роботи системи, що, у свою чергу, суттєво спрощує та прискорює процес програмної розробки.

З огляду на це було прийнято рішення сформувати й детально описати загальний алгоритм роботи модуля текстових повідомлень.

Робота модуля розпочинається з ініціалізації WebSocket-з’єднання та його подальшої верифікації. У разі успішного підключення користувачі авторизуються у відповідній чат-кімнаті. Ключовим елементом системи є механізм відстеження стану повідомлень, побудований на базі патерну «Спостерігач» (Observer). Цей поведінковий шаблон забезпечує автоматичне оновлення даних: щойно стан повідомлень змінюється, суб’єкт (постачальник) миттєво сповіщає всіх зареєстрованих спостерігачів (користувачів). Кінцевим етапом є рендеринг оновленого списку повідомлень в інтерфейсі чату.

2.4 Аналіз JavaScript бібліотек розробки інтерфейсу користувача

Після визначення мови програмування для створення фронтенд-частини застосунку наступним важливим етапом став вибір найбільш ефективної бібліотеки для розробки користувацького інтерфейсу. Було

прийнято рішення зосередитись саме на бібліотеках проєктування інтерфейсів, оскільки вони надають широкий набір інструментів і функціональних можливостей для створення сучасних вебзастосунків. Серед найбільш популярних рішень у Front-end спільноті на сьогодні можна виділити наступні: React.js, Vue.js, Angular.js.

Для визначення найбільш доцільного варіанту було проведено аналіз переваг і недоліків кожної бібліотеки. Це дозволило обрати рішення, яке найкраще відповідає вимогам проєкту та сприятиме підвищенню якості користувацької взаємодії із застосунком.

2.4.1 Аналіз бібліотеки React.js

React є однією з найпопулярніших бібліотек для розробки користувацьких інтерфейсів, що зумовлено її високою продуктивністю, гнучкістю та широкими можливостями для створення сучасних вебзастосунків. Однією з ключових переваг React є використання технології віртуального DOM, яка дозволяє оптимізувати оновлення інтерфейсу користувача. Замість повного перезавантаження сторінки або оновлення всього DOM-дерева система визначає лише ті елементи, які зазнали змін, і повторно відображає їх. Завдяки цьому суттєво підвищується швидкодія застосунку, зменшується навантаження на браузер і покращується загальний користувацький досвід, особливо під час роботи зі складними інтерфейсами та великими обсягами даних.

Важливою перевагою React є також активна підтримка спільноти розробників. Бібліотека широко використовується як окремими програмістами, так і великими компаніями для створення вебзастосунків різного рівня складності. Це сприяє постійному розвитку технології, регулярному оновленню документації та появі великої кількості навчальних матеріалів. Крім того, для React створено значну кількість додаткових

бібліотек та готових рішень, що дозволяє швидше реалізовувати необхідний функціонал і скорочувати час розробки.

Ще однією характерною особливістю React є його гнучкість та можливість адаптації до різних вимог проєкту. На відміну від багатьох повноцінних фреймворків, бібліотека не нав’язує жорсткої структури застосунку, що дає змогу розробникам самостійно обирати архітектурні рішення та інструменти для реалізації окремих функцій. Завдяки цьому React успішно використовується як для створення невеликих вебсайтів, так і для розробки масштабних корпоративних систем.

Додатковою перевагою є підтримка мультиплатформеної розробки. За допомогою технології React Native можна створювати мобільні застосунки для різних операційних систем, використовуючи підходи та принципи, аналогічні веброзробці. Це дозволяє повторно використовувати частину програмного коду та спрощує процес створення мобільних версій програмних продуктів. Такий підхід сприяє скороченню витрат часу та ресурсів на розробку, а також забезпечує швидше впровадження нових функцій на різних платформах.

Незважаючи на значну кількість переваг, React також має певні недоліки, які необхідно враховувати під час вибору технології для розробки вебзастосунку. Однією з особливостей бібліотеки є її висока гнучкість, яка надає розробникам широкі можливості для вибору архітектурних рішень, організації структури проєкту та підбору додаткових інструментів. З одного боку, це дозволяє адаптувати застосунок до конкретних потреб і вимог, проте з іншого – покладає на розробника значну відповідальність за проєктування якісної та масштабованої архітектури. Відсутність чітко визначених правил організації проєкту може призвести до використання різних підходів навіть у межах однієї команди, що ускладнює підтримку та розвиток програмного забезпечення.

Крім того, недостатньо продумана структура застосунку на початкових етапах розробки може стати причиною суттєвих труднощів у майбутньому. Зі

збільшенням кількості компонентів, сторінок та функціональних можливостей підтримка такого проєкту стає складнішою, а внесення змін потребує додаткових часових витрат. Помилки в організації структури коду можуть негативно впливати на продуктивність команди розробників, ускладнювати тестування та інтеграцію нових функцій. Саме тому під час використання React особливу увагу необхідно приділяти плануванню архітектури застосунку, вибору підходів до керування станом та організації взаємодії між компонентами ще на ранніх етапах реалізації проєкту.

2.4.2 Аналіз бібліотеки Vue.js

Подібно до React.js, Vue.js [21, 22] також має низку вагомих переваг:

– реактивність даних. Vue.js використовує систему реактивності, яка автоматично оновлює відображення компонентів під час зміни їхнього стану. Це забезпечує зручне керування компонентами на логічному рівні та ефективний зв’язок між логікою застосунку і шаром відображення;

– використання механізму «віртуального DOM». Подібно до React.js, Vue.js застосовує технологію віртуального DOM, що дозволяє оптимізувати процес оновлення інтерфейсу та підвищити продуктивність застосунку;

– активна спільнота розробників. Хоча спільнота Vue.js дещо поступається React.js за масштабом, вона також є достатньо великою та активною, що забезпечує доступ до документації, навчальних матеріалів і готових рішень для розробки;

– простота у вивченні. Однією з ключових переваг Vue.js порівняно з React.js є легкість освоєння. Завдяки декларативному підходу до створення компонентів та лаконічному синтаксису, розробники можуть швидше розпочати створення необхідного інтерфейсу;

– гнучкість використання. Vue.js може застосовуватися як для створення невеликих інтерактивних елементів, так і як повноцінний

фреймворк для розробки складних користувацьких інтерфейсів, що робить його універсальним рішенням для різних типів проєктів.

2.4.3 Аналіз бібліотеки Angular.js

Angular є повноцінним фреймворком для розробки вебзастосунків, який надає розробникам широкий набір вбудованих інструментів і можливостей. Однією з його основних переваг є наявність великої кількості функцій, доступних одразу після встановлення. Зокрема, фреймворк містить засоби для організації маршрутизації, керування станом застосунку, роботи з формами, валідації даних та інші компоненти, необхідні для створення сучасних вебсистем. Це дозволяє скоротити час розробки та зменшити потребу у використанні сторонніх бібліотек.

Важливою особливістю Angular є використання мови TypeScript як основного інструменту розробки. Завдяки підтримці суворої типізації забезпечується підвищення надійності програмного коду, спрощується пошук помилок на етапі розробки та покращується підтримуваність проєкту. Крім того, використання TypeScript сприяє створенню більш структурованого та зрозумілого коду, що особливо важливо під час роботи над великими програмними системами.

Ще однією перевагою Angular є його висока масштабованість. Фреймворк орієнтований на розробку складних і функціонально насичених застосунків, тому передбачає чітку архітектуру проєкту та механізми організації коду. Підтримка модульної структури дозволяє розділяти функціональність на окремі компоненти, сервіси та модулі, що значно спрощує супровід, тестування та подальший розвиток програмного забезпечення.

Також Angular надає широкий набір готових компонентів та інструментів, які можуть використовуватися без додаткового налаштування.

Це дозволяє прискорити процес створення інтерфейсу користувача, забезпечити єдність дизайну та функціональності різних частин застосунку, а також зменшити обсяг програмного коду, який необхідно реалізовувати вручну. Завдяки цьому розробники можуть зосередитися на реалізації бізнес-логіки та специфічних функцій системи, підвищуючи ефективність процесу розробки.Порівнюючи React.js та Vue.js у контексті поставленого завдання, доцільно надати перевагу React.js, оскільки він забезпечує достатню гнучкість для налаштування архітектури застосунку, має ефективну інтеграцію з TypeScript та пропонує значно більшу кількість сторонніх бібліотек, що спрощують процес розробки.

У порівнянні з Angular.js, React.js також має низку переваг. Зокрема, він є більш популярним на ринку праці, що розширює можливості вибору підходів та інструментів для реалізації застосунку. Крім того, для даного проєкту відсутня необхідність у використанні повноцінного фреймворку, оскільки значна частина функціональності Angular.js не буде застосовуватися, але при цьому ускладнюватиме процес розробки. Також Angular.js має меншу кількість сторонніх бібліотек для вирішення спеціалізованих задач. Враховуючи наведені фактори, у межах даного проєкту було прийнято рішення обрати React.js як основну бібліотеку для розробки користувацького інтерфейсу.

2.5 Аналіз технологій побудови інфраструктури вебдодатків

Наступним етапом було проведення аналізу підходів до реалізації безпечного API, яке забезпечуватиме надійну взаємодію між Front-end та Back-end частинами застосунку. Першочерговим завданням у процесі проєктування API є вибір найбільш оптимальної мови програмування та технології, що відповідатимуть поставленим вимогам. Для реалізації такого завдання існує значна кількість рішень, серед яких найбільш популярними є Node.js та Python.

Для вибору найбільш доцільного варіанту було виконано послідовний аналіз кожної з наведених технологій.

2.5.1 Аналіз Node.js

Node.js є серверною платформою, яка надає можливість виконувати код мовою JavaScript на стороні сервера, що робить її одним із найпопулярніших рішень для розробки сучасних вебзастосунків. Однією з основних переваг Node.js є можливість використання єдиної мови програмування як для клієнтської, так і для серверної частини застосунку. Такий підхід значно спрощує процес розробки, оскільки розробники можуть працювати в межах одного технологічного стеку, використовуючи спільні підходи, інструменти та структури даних. Це сприяє підвищенню продуктивності команди, спрощує підтримку програмного забезпечення та зменшує витрати часу на інтеграцію різних технологій.

Важливою перевагою є також підтримка Full-stack розробки, яка дозволяє об’єднати клієнтську та серверну логіку в межах одного проєкту. Завдяки цьому зменшується складність архітектури системи та спрощується взаємодія між окремими компонентами застосунку. Сучасні фреймворки, побудовані на основі JavaScript, надають можливість реалізовувати серверні API та клієнтські інтерфейси в межах єдиного середовища розробки, що сприяє більш швидкому створенню та розгортанню програмних продуктів.

Node.js використовує асинхронну подієво-орієнтовану модель виконання, яка дозволяє ефективно працювати з великою кількістю одночасних запитів без блокування основного потоку виконання. Завдяки цьому платформа демонструє високу продуктивність під час обробки мережевих операцій, роботи з файлами та взаємодії з базами даних. Такий підхід особливо важливий для вебзастосунків, які повинні забезпечувати

швидкий відгук системи навіть за умов значної кількості активних користувачів.

Основою Node.js є високопродуктивний рушій V8, розроблений для виконання JavaScript-коду. Він забезпечує швидку компіляцію та виконання програм, що позитивно впливає на швидкодію застосунку та зменшує час обробки запитів. Завдяки цьому Node.js успішно використовується для створення високонавантажених вебсистем, сервісів реального часу, чатів, платформ потокової передачі даних та інших програмних рішень, де важливими є швидкість роботи та масштабованість.

Ще однією суттєвою перевагою платформи є наявність розвиненої екосистеми програмних пакетів, доступних через менеджер пакетів NPM. Розробники мають доступ до великої кількості готових бібліотек і модулів для реалізації різноманітної функціональності, починаючи від роботи з базами даних і закінчуючи інструментами автентифікації, тестування та забезпечення безпеки. Це дозволяє значно прискорити процес розробки, уникнути повторного створення вже існуючих рішень та забезпечити швидке впровадження нових можливостей у програмний продукт.

Серед недоліків Node.js можна виділити:

– однопоточну модель виконання, яка може створювати труднощі при виконанні ресурсомістких або блокуючих операцій;

– нерівномірну підтримку сторонніх модулів, через що деякі бібліотеки можуть втрачати актуальність або потребувати додаткових ресурсів на підтримку.

2.5.2 Аналіз Python

Python [25] є однією з найпопулярніших мов програмування завдяки простому синтаксису, високій читабельності коду та ефективності процесу розробки. Вона широко використовується для створення вебзастосунків, API-

сервісів, систем автоматизації, аналітики даних і рішень у сфері штучного інтелекту. Завдяки своїй універсальності Python підходить як для невеликих навчальних проєктів, так і для масштабних промислових систем.

Однією з ключових переваг мови є простий та інтуїтивно зрозумілий синтаксис. Код на Python легко читається та сприймається навіть розробниками з невеликим досвідом, що значно скорочує час на навчання та пришвидшує процес розробки програмного забезпечення. Це робить мову особливо зручною для командної роботи та швидкого прототипування рішень.

Ще однією важливою перевагою є наявність великої кількості готових фреймворків для створення API та вебзастосунків, серед яких найбільш популярними є Flask і Django. Flask забезпечує легкість і гнучкість під час розробки невеликих сервісів, тоді як Django пропонує повноцінний набір інструментів для створення складних і масштабованих вебсистем. Це дозволяє обирати оптимальний інструмент залежно від вимог конкретного проєкту.

Також варто відзначити велику та активну спільноту розробників Python, яка постійно підтримує розвиток екосистеми мови. Завдяки цьому регулярно з’являються нові бібліотеки, інструменти та рішення, що спрощують реалізацію різноманітних задач і забезпечують швидке розв’язання технічних проблем.

У сукупності ці фактори роблять Python ефективним та універсальним інструментом для розробки сучасних програмних систем.Особливу увагу було приділено фреймворкам Flask та Django.

Django є повноцінним фреймворком із широким набором вбудованих інструментів для створення вебзастосунків. Він містить ORM для роботи з базами даних, систему маршрутизації, механізми автентифікації та авторизації користувачів, адміністративну панель та інші можливості. Крім того, Django пропонує чіткі правила організації коду, що спрощує взаємодію

розробників у команді. До недоліків можна віднести високу складність для новачків і надмірну громіздкість для невеликих проєктів.

Flask, навпаки, є мінімалістичним фреймворком, який містить лише базовий набір інструментів. Це дозволяє швидко розпочати розробку та забезпечує більшу свободу у виборі архітектури застосунку. Однак відповідальність за побудову структури проєкту значною мірою покладається на розробника, а кількість готових рішень і розмір спільноти є меншими порівняно з Django.

Основними перевагами Ruby є зрозумілий та елегантний синтаксис, що полегшує читання коду, а також наявність потужних фреймворків із готовими інструментами для автоматизації типових задач розробки. Крім того, Ruby має достатньо розвинену екосистему бібліотек і пакетів.

Серед недоліків можна виділити нижчу продуктивність порівняно з деякими іншими мовами програмування, що особливо помітно під час роботи з великими обсягами користувацького трафіку. Також, попри елегантність синтаксису, його освоєння може бути складнішим для нових розробників.

На окремому етапі проєктування було розглянуто Full-stack фреймворк Next.js для React.js, який у межах поставленої задачі демонструє низку суттєвих переваг, що роблять його доцільним для використання під час розробки сучасних вебзастосунків.

Однією з ключових переваг є використання єдиної кодової бази для front-end і back-end частин застосунку. Такий підхід дозволяє об’єднати логіку клієнтської та серверної взаємодії в межах одного проєкту, що спрощує структуру коду, зменшує кількість дублювання логіки та полегшує супровід програмного продукту. Це також сприяє підвищенню продуктивності розробки, оскільки команда працює в єдиному технологічному середовищі.

Важливою особливістю Next.js є підтримка Server-Side Rendering (SSR), що дозволяє формувати сторінки на стороні сервера перед їх відправленням

клієнту. Це забезпечує можливість використання CDN для прискорення доставки контенту, покращує SEO-оптимізацію вебзастосунку та суттєво скорочує час початкового завантаження сторінок. Завдяки цьому користувач отримує швидший і стабільніший доступ до інтерфейсу застосунку.

Окремо слід відзначити зручну систему маршрутизації, яка спрощує організацію сторінок у проєкті, а також можливість створення API-маршрутів без необхідності підключати окремий серверний фреймворк. Це дозволяє реалізовувати як REST, так і GraphQL підходи в межах одного застосунку, що значно підвищує гнучкість архітектури.

Крім того, Next.js вирізняється простотою розгортання застосунку на сучасних хмарних платформах, таких як Vercel або Heroku. Це дозволяє швидко публікувати проєкт, автоматизувати процеси деплойменту та забезпечувати стабільну роботу сервісу в продакшн-середовищі.

Також важливою перевагою є повна інтеграція з екосистемою React.js, що відкриває доступ до великої кількості готових бібліотек, компонентів та інструментів. Це значно прискорює процес розробки та дозволяє ефективно реалізовувати складний функціонал із мінімальними витратами часу.

У сукупності зазначені переваги роблять Next.js потужним та універсальним інструментом для створення сучасних вебзастосунків, особливо в умовах необхідності швидкої розробки, масштабованості та високої продуктивності. Водночас Next.js має і певні недоліки, які слід враховувати при виборі технології для розробки вебзастосунків. Зокрема, орієнтація на серверний рендеринг може певною мірою обмежувати можливості створення складних і високонавантажених API-систем порівняно зі спеціалізованими серверними фреймворками. Це пов’язано з тим, що основна архітектура Next.js спрямована на поєднання фронтенд- та бекенд-логіки в межах одного проєкту, що не завжди є оптимальним для масштабних серверних рішень.

Іншим недоліком є відносно високий поріг входження для розробників, які не мають досвіду роботи з бібліотекою React.js. Оскільки Next.js

базується саме на React, для ефективної роботи з фреймворком необхідно розуміти принципи компонентної архітектури, управління станом та особливості рендерингу, що може ускладнювати процес навчання та збільшувати час адаптації нових розробників до проєкту.

Також варто відзначити певні обмеження, пов’язані з масштабуванням серверної частини застосунку під час створення великих і складних програмних систем. На початкових етапах розробки використання вбудованих API-маршрутів Next.js є зручним та ефективним рішенням, оскільки дозволяє реалізувати клієнтську та серверну логіку в межах одного проєкту. Проте зі збільшенням кількості функціональних можливостей, бізнес-процесів та інтеграцій із зовнішніми сервісами структура серверного коду може суттєво ускладнюватися.

Крім того, зі зростанням навантаження та кількості користувачів виникають додаткові вимоги щодо забезпечення продуктивності, відмовостійкості та балансування навантаження між серверами. У таких випадках реалізація всієї серверної логіки в межах одного Next.js-застосунку може стати менш ефективною порівняно з використанням спеціалізованих серверних фреймворків або мікросервісної архітектури. Це може вимагати додаткових зусиль для реорганізації структури проєкту, перенесення окремих модулів у незалежні сервіси та впровадження механізмів взаємодії між ними.

Також ускладнюється процес тестування, моніторингу та підтримки системи. Зі збільшенням кількості API-маршрутів та бізнес-логіки підтримка єдиного серверного застосунку може потребувати значних ресурсів і ретельного контролю залежностей між окремими модулями. У довгостроковій перспективі це може впливати на швидкість впровадження нових функцій, підвищувати складність супроводу та збільшувати витрати на подальший розвиток програмного продукту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1 Обґрунтування вибору засобів для реалізації вебзастосунку

JavaScript – це динамічна скриптова мова програмування, яка підтримує прототипно-орієнтовану модель побудови об’єктів. Її синтаксис схожий на мови Java та C++, що спрощує процес освоєння для розробників, які вже працювали з цими технологіями. JavaScript може використовуватись як для процедурного, так і для об’єктно-орієнтованого програмування. Об’єкти в цій мові створюються динамічно шляхом додавання властивостей і методів під час виконання програми, на відміну від класичного підходу з використанням синтаксичних класів, характерного для мов, як-от C++ або Java. Після створення об’єкт може використовуватись як прототип для побудови інших подібних об’єктів.

Java [26] є універсальною об’єктно-орієнтованою мовою програмування, заснованою на класах, а також потужною платформою для розробки програмних застосунків. Вона характеризується високим рівнем безпеки та надійності, що зробило її популярною для розробки програмного забезпечення для персональних комп’ютерів, серверів, мобільних пристроїв, ігрових консолей і суперкомп’ютерів. Скомпільований Java-код у вигляді байт-коду може виконуватись на різних операційних системах, зокрема Windows, Linux та macOS. За синтаксисом Java значною мірою подібна до мов C і C++.

Python – це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Завдяки вбудованим структурам даних та динамічній типізації Python є ефективним інструментом для швидкої розробки застосунків, а також часто використовується як скриптова мова або як проміжний рівень інтеграції між компонентами системи. Простий і зрозумілий синтаксис Python забезпечує високу читабельність коду, що спрощує його підтримку та супровід. Мова підтримує модульність

через систему модулів і пакетів, що сприяє повторному використанню програмного коду.

PHP – це інтерпретована скриптова мова програмування з відкритим вихідним кодом, яка підтримує об’єктно-орієнтований підхід і виконується на сервері. PHP широко використовується у веброзробці для створення динамічних вебзастосунків. Скрипти PHP виконуються безпосередньо на сервері, формуючи HTML-вміст для клієнта. Однією з переваг PHP є висока швидкість виконання сценаріїв порівняно з деякими іншими технологіями, зокрема JSP та ASP. Крім того, PHP ефективно використовує серверні ресурси, що дозволяє зменшити навантаження на сервер і скоротити час обробки запитів, забезпечуючи кращу продуктивність вебзастосунків.

Для вибору середовища розробки програмного застосунку було проведено аналіз таких інструментів: WebStorm, VSCode, Sublime Text 3 та Brackets.

WebStorm – це інтегроване середовище розробки, призначене для створення програм на JavaScript та суміжних технологіях, зокрема TypeScript, React, Vue, Angular, Node.js, HTML і CSS. Це середовище побудоване на платформі IntelliJ IDEA. На рисунку 3.1 наведено інтерфейс середовища розробки WebStorm.

WebStorm [27] надає широкі можливості для розробки програмного забезпечення, зокрема автодоповнення коду, аналіз помилок у режимі реального часу, зручну навігацію, рефакторинг, відлагодження та інтеграцію із системами контролю версій.

Однією з ключових переваг цього інтегрованого середовища розробки є ефективна робота з проєктами різної структури. Зокрема, підтримується рефакторинг JavaScript-коду, який може бути розміщений у різних файлах і папках або вбудований в HTML-документи. Також середовище коректно працює з багаторівневою вкладеністю коду, коли HTML містить JavaScript, який, у свою чергу, включає інші HTML-фрагменти та скрипти.

Додатковою перевагою WebStorm є наявність безкоштовної ліцензії для студентів.

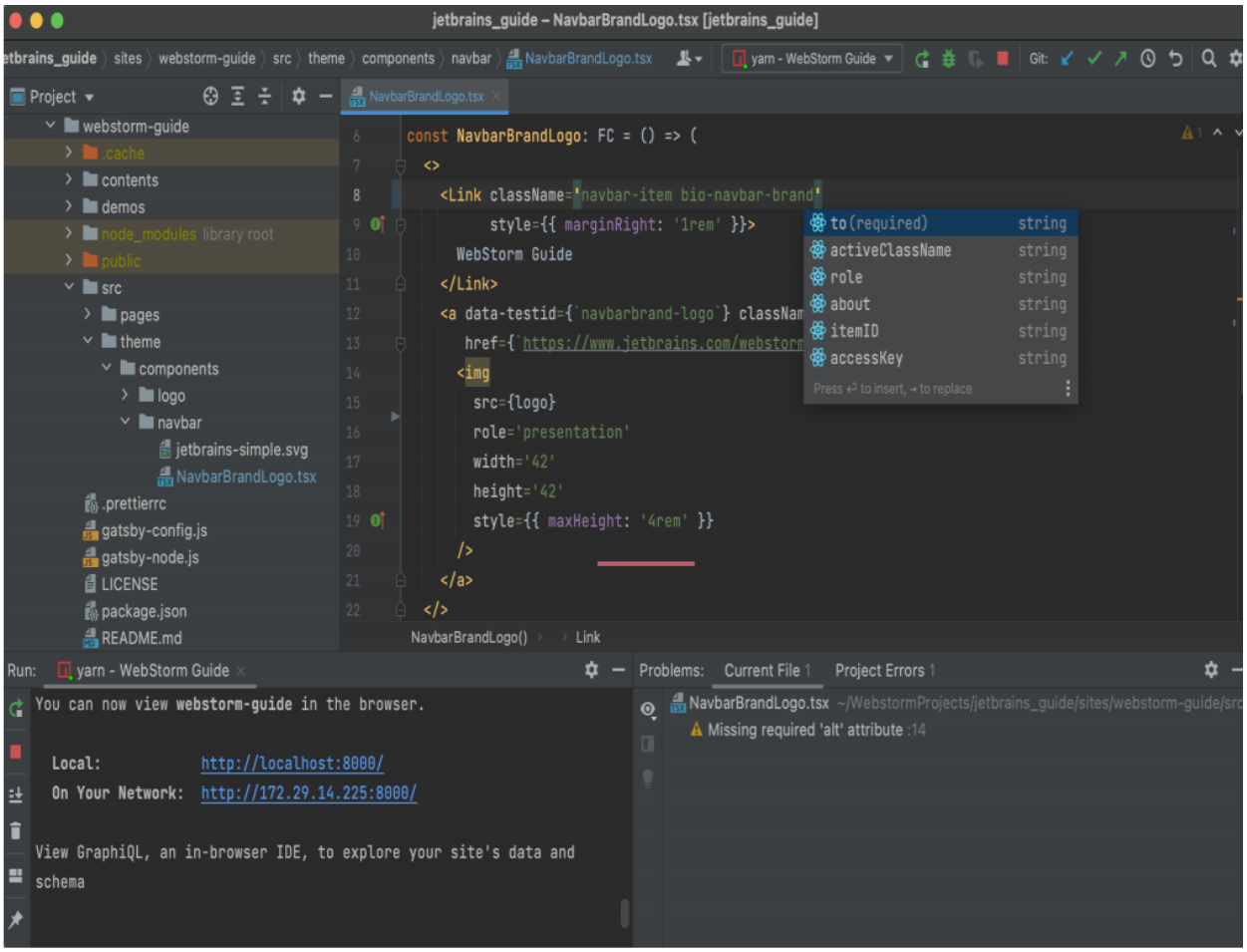

Рисунок 3.1 – Інтерфейс середовища WebStorm

Visual Studio Code (VSCode) – це сучасний редактор вихідного коду, розроблений компанією Microsoft для операційних систем Windows, Linux та macOS. Дане середовище розробки характеризується високою швидкодією, гнучкою налаштовністю і підтримкою великої кількості мов програмування та технологій [28, 29].

VSCode підтримує роботу з JavaScript, TypeScript, Python, Java, C++, PHP та багатьма іншими мовами програмування. Завдяки вбудованій системі розширень користувач може додавати додаткові модулі для роботи з фреймворками, базами даних, системами контролю версій, контейнерами Docker та іншими інструментами розробки.

Серед основних можливостей середовища варто виділити автодоповнення коду, підсвічування синтаксису, інтелектуальний аналіз помилок, інтегрований термінал, засоби налагодження програм і підтримку Git. Крім того, VSCode забезпечує зручну навігацію в проєкті, підтримує одночасну роботу з великою кількістю файлів і має інтуїтивно зрозумілий інтерфейс.

Ще однією важливою перевагою Visual Studio Code є його безкоштовне розповсюдження та активна підтримка спільноти розробників, що забезпечує регулярне оновлення функціоналу й наявність великої кількості готових розширень. Інтерфейс середовища розробки VSCode наведено на рисунку 3.2.

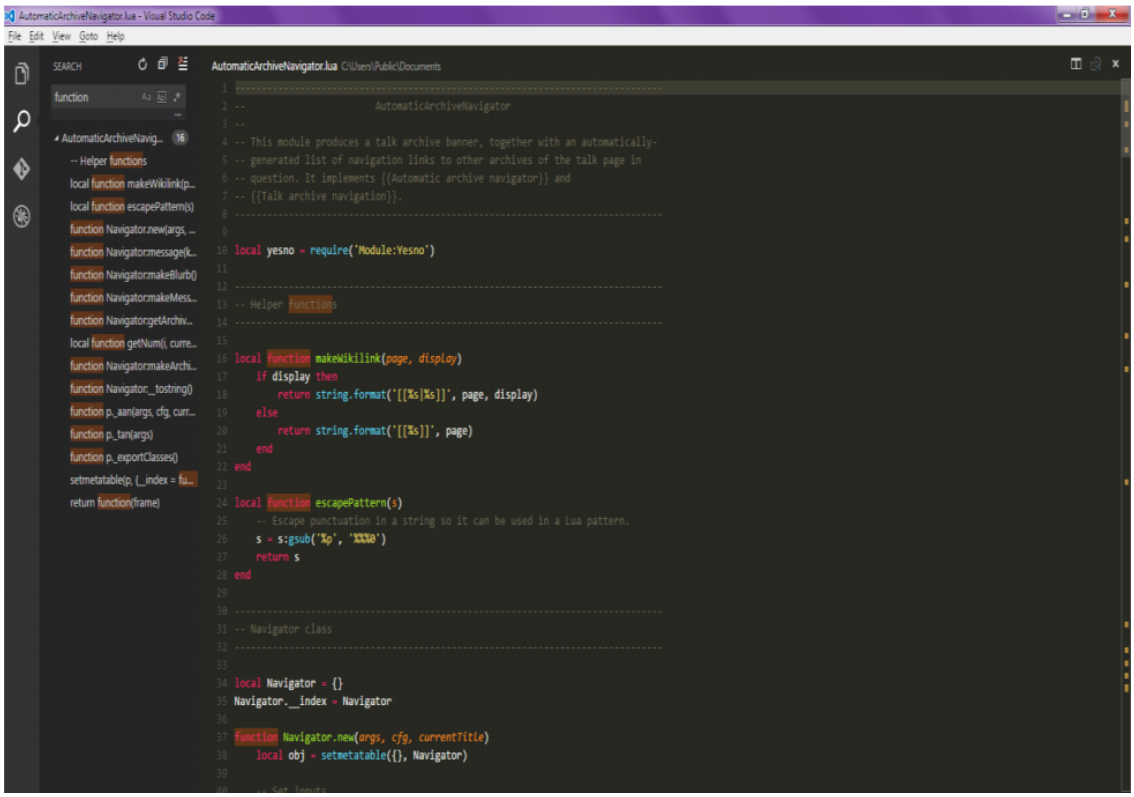

Рисунок 3.2 – Інтерфейс середовища VSCode

Отже, враховуючи всі переваги та особливості, прийнято рішення, що середовище розробки WebStorm найбільше підходить для виконання поставленої задачі.

3.2 Програмна реалізація вебзастосунку

Модуль відеозв’язку реалізовано у файлі useStartPeerSession.ts і використовується для встановлення з’єднання між користувачами під час відеозв’язку. Основним призначенням цього модуля є організація обміну медіаданими між учасниками конференції та забезпечення стабільного з’єднання в режимі реального часу [30-34].

Для спрощення процесу підключення нових користувачів до відеокімнати було створено клас PeerConnectionSession, який відповідає за керування peer-to-peer з’єднаннями між учасниками. Використання окремого класу дозволяє централізовано реалізувати логіку створення, підтримки та завершення з’єднань, а також підвищує зручність подальшого супроводу й масштабування програмного застосунку.

Клас PeerConnectionSession є одним із ключових компонентів модуля відеозв’язку та відповідає за керування WebRTC-з’єднаннями між учасниками сеансу. Його функціональність охоплює створення, підтримку та завершення peer-to-peer-з’єднань, а також обробку сигналізаційних повідомлень, необхідних для встановлення прямого зв’язку між користувачами.

Метод addPeerConnection використовується для створення нового peer-з’єднання та додавання його до колекції активних з’єднань. Під час виконання цього методу відбувається ініціалізація об’єкта RTCPeerConnection, налаштування необхідних обробників подій і підготовка каналу передавання мультимедійних даних між учасниками сеансу. Це забезпечує можливість встановлення стабільного відео- та аудіозв’язку.

Метод removePeerConnection призначений для видалення активного з’єднання з конкретним користувачем. У процесі його виконання здійснюється закриття каналу зв’язку, звільнення задіяних ресурсів і видалення інформації про з’єднання зі структури даних класу. Використання цього методу дозволяє підтримувати актуальний стан системи та уникати

накопичення неактивних з’єднань.

Метод callUser забезпечує ініціалізацію виклику іншого учасника системи. Під час його роботи формується SDP-пропозиція (Session Description Protocol), яка містить параметри майбутнього мультимедійного сеансу. Після створення пропозиції вона передається іншому користувачеві через сигнальний сервер, який запускає процес встановлення прямого з’єднання між учасниками.

Метод joinRoom відповідає за підключення користувача до кімнати відеозв’язку. Після виклику цього методу користувач реєструється в обраній кімнаті, отримує інформацію про інших учасників і може розпочати процес встановлення peer-з’єднань із ними. Завдяки цьому забезпечується можливість організації групових відеоконференцій та онлайн-консультацій.

Метод onCallMade виконує обробку події надходження нового виклику. Після отримання SDP-пропозиції від іншого учасника система аналізує отримані параметри сеансу, створює відповідне з’єднання та готує відповідь для завершення процедури узгодження параметрів зв’язку. Цей метод є важливим елементом процесу сигналізації в архітектурі WebRTC.

Метод onAnswerMade призначений для обробки відповіді на раніше створений виклик. Після отримання SDP-відповіді від віддаленого користувача виконується оновлення параметрів активного з’єднання та завершення процедури встановлення сеансу зв’язку. У результаті між учасниками формується повноцінний канал передавання аудіо- та відеоданих.

Метод clearConnections використовується для завершення роботи всіх активних peer-з’єднань. Під час його виконання здійснюється послідовне закриття кожного каналу зв’язку, очищення внутрішніх колекцій і звільнення зайнятих ресурсів. Застосування цього методу є необхідним під час завершення сеансу відеозв’язку, виходу користувача з кімнати або завершення роботи додатка.

Таким чином, клас PeerConnectionSession реалізує повний цикл

керування WebRTC-з’єднаннями та забезпечує надійну взаємодію між учасниками системи відеозв’язку, починаючи зі створення виклику та підключення до кімнати і завершуючи коректним закриттям усіх активних сеансів зв’язку.

Таким чином, клас PeerConnectionSession забезпечує повний цикл управління відеоз’єднаннями між користувачами та є ключовим елементом реалізації функціоналу відеозв’язку в застосунку. Код класу PeerConnectionSession наведено на рисунку 3.3.

```
import io from 'socket.io-client'

const { RTCPeerConnection, RTCSessionDescription } = window

function capitalizeFirstLetter(str: any) {
  return str.charAt(0).toUpperCase() + str.slice(1)
}

class PeerConnectionSession {
  _onConnected: any
  _onDisconnected: any
  _room: any
  peerConnections: any = {};
  senders: any[] = [];
  listeners: any = {};
  socket: any;

  constructor(socket: any) {
    this.socket = socket
    this.onCallMade()
  }
```

Рисунок 3.3 – Клас PeerConnectionSession

Метод addPeerConnection використовується для створення нового peer-to-peer з’єднання між користувачами системи відеозв’язку. Основним призначенням цього методу є ініціалізація підключення, налаштування передачі медіапотоків та збереження інформації про активного учасника конференції.

Метод приймає три основні параметри, необхідні для коректного створення та налаштування peer-з’єднання між учасниками системи відеозв’язку.

Першим параметром є id – унікальний ідентифікатор користувача. Даний параметр використовується для однозначного розпізнавання учасника в межах системи та забезпечує можливість встановлення зв’язку між конкретними користувачами. Ідентифікатор зберігається у внутрішній структурі даних, що містить інформацію про всі активні підключення, завдяки чому система може швидко знаходити необхідне з’єднання, оновлювати його або завершувати його. Використання унікальних ідентифікаторів також дозволяє уникнути конфліктів між учасниками під час одночасної роботи кількох користувачів у межах однієї кімнати відеозв’язку.

Іншим параметром є stream, який містить мультимедійний потік користувача. Цей об’єкт формується браузером після отримання дозволу на використання камери та мікрофона і включає аудіо- та відеодоріжки, що передаються в режимі реального часу. Потік використовується для додавання локальних медіаданих до peer-з’єднання, після чого вони стають доступними іншим учасникам сеансу. Завдяки використанню об’єкта stream забезпечується передача зображення з вебкамери, голосового сигналу та інших мультимедійних даних безпосередньо між клієнтами, що є основою функціонування технології WebRTC.

Третім параметром є callback – функція зворотного виклику, яка передається з функціонального компонента клієнтської частини вебзастосунку. Вона виконується після успішного встановлення з’єднання або настання визначеної події під час процесу підключення. Основним

призначенням цієї функції є забезпечення взаємодії між логікою встановлення з’єднання та користувацьким інтерфейсом. За її допомогою може виконуватися оновлення стану компонентів, відображення інформації про нового учасника, додавання відеопотоку в інтерфейс або виконання інших дій, необхідних для коректної роботи системи. Використання механізму зворотних викликів дозволяє підвищити гнучкість архітектури вебзастосунку та забезпечити чітке розділення бізнес-логіки й елементів інтерфейсу користувача.

Під час виконання метод ініціалізує нове peer-to-peer з’єднання між учасниками системи, що є базовим елементом архітектури модуля відеозв’язку. На цьому етапі створюється об’єкт peer-з’єднання, який відповідає за встановлення прямого каналу передачі даних між клієнтами без посередництва сервера для мультимедійного трафіку. У межах цього процесу виконується конфігурація параметрів з’єднання, ініціалізація необхідних обробників подій, а також підготовка інфраструктури для передачі аудіо- та відеоданих.

Далі до створеного з’єднання додаються медіапотоки користувача, які включають аудіо- та відеодоріжки, отримані з пристроїв введення (мікрофона та вебкамери). Інтеграція цих потоків у канал передачі даних забезпечує можливість двосторонньої комунікації між учасниками конференції в режимі реального часу. Таким чином, кожен користувач може одночасно передавати власний медіаконтент і отримувати потоки від інших учасників сеансу.

Окремим етапом роботи методу є реєстрація користувача у структурі активних підключень. Це дозволяє системі вести облік усіх поточних peer-з’єднань, відстежувати їхній стан та оперативно керувати життєвим циклом кожного з них. Такий підхід є важливим для забезпечення стабільності роботи системи, оскільки дозволяє своєчасно виявляти неактивні або розірвані з’єднання та виконувати їх коректне завершення.

Завдяки описаній логіці реалізується надійний механізм обміну відео- та аудіоданими між учасниками конференції, що забезпечує високу якість

комунікації та мінімальні затримки під час взаємодії. Це є критично важливим для системи онлайн-консультацій, де стабільність зв’язку та швидкість передачі даних безпосередньо впливають на ефективність спілкування між лікарем і пацієнтом.

Код методу addPeerConnection наведено на рисунку 3.4.

```
addPeerConnection(id: any, stream: any, callback: (stream: any) => void) {
  this.peerConnections[id] = new RTCPeerConnection( configuration: {
    iceServers: [{ urls: 'stun:stun.l.google.com:19302' }],
  })

  stream.getTracks().forEach((track: MediaStreamTrack) => {
    this.senders.push(this.peerConnections[id].addTrack(track, stream))
  })

  this.listeners[id] = (event: any) => {
    const fn = this[('_on' + capitalizeFirstLetter(this.peerConnections[id].connectionState))
    fn && fn(event, id)
  }

  this.peerConnections[id].addEventListener('connectionstatechange', this.listeners[id])

  this.peerConnections[id].ontrack = function ({ streams: [stream] }: { streams: any }) {
    console.log({ id, stream })
    callback(stream)
  }

  console.log(this.peerConnections)
}
```

Рисунок 3.4 – Метод addPeerConnection

Метод removePeerConnection приймає аргумент id, який є ідентифікатором активного підключення, та використовується для завершення peer-to-peer з’єднання між користувачами. Під час виконання методу відбувається видалення інформації про підключення зі списку

активних з’єднань, а також закриття каналу передавання даних між браузерами користувачів. Це дає змогу коректно завершити сеанс відеозв’язку та звільнити системні ресурси.

Метод callUser приймає аргумент to, що містить ідентифікаційний номер користувача, з яким потрібно встановити з’єднання. Даний метод використовується для ініціалізації виклику та надсилання співрозмовнику повідомлення про вхідний дзвінок. Після виконання методу сервер передає необхідні сигнали іншому користувачу для подальшого встановлення з’єднання peer-to-peer.

Метод joinRoom приймає аргумент room, який визначає ідентифікатор кімнати відеозв’язку. Основним призначенням цього методу є додавання користувача до відповідної кімнати та інформування інших учасників про його приєднання. Завдяки цьому всі учасники конференції можуть отримати інформацію про нові підключення та встановити необхідні peer-to-peer з’єднання між собою.

Метод onCallMade використовується для обробки події успішного створення вихідного виклику. Після ініціалізації виклику метод надсилає серверу повідомлення про створення нового з’єднання, а сервер, у свою чергу, сповіщає співрозмовника про вхідний виклик. Це забезпечує синхронізацію процесу встановлення з’єднання між двома користувачами.

Метод onAnswerMade приймає аргумент callback, який являє собою функцію зворотного виклику, що виконується у випадку позитивної відповіді співрозмовника на вхідний виклик. Даний метод використовується для підтвердження успішного встановлення з’єднання із сервером та передачі клієнтській частині адреси співрозмовника. Це дозволяє встановити пряме peer-to-peer з’єднання між браузерами користувачів без використання додаткових посередників.

Метод clearConnections призначений для очищення даних, накопичених під час дзвінка, а також для завершення активних з’єднань між користувачами. Під час виконання методу закривається WebSocket-з’єднання

із сервером, а також видаляється інформація про всі активні peer-to-peer підключення.

Модуль текстових повідомлень реалізовано у файлі useChat.ts та використовується для обміну повідомленнями між користувачами у режимі реального часу. Робота модуля базується на використанні WebSocket-з'єднань, що забезпечують миттєву передачу повідомлень між клієнтською та серверною частинами застосунку.

Серверна частина модуля реалізована у вигляді набору спостерігачів (обробників подій), які забезпечують координацію взаємодії між користувачами, контроль стану підключень та передачу службових повідомлень між клієнтською та серверною частинами системи. Такий підхід дозволяє організувати подієво-орієнтовану архітектуру, у якій кожна подія обробляється окремим компонентом, що відповідає за виконання визначеного набору функцій.

Спостерігач IS_USER використовується для ідентифікації користувача та перевірки його наявності в системі. Після отримання відповідного запиту сервер виконує пошук користувача за його унікальними даними та визначає поточний стан підключення. Отримана інформація використовується для перевірки можливості встановлення зв'язку між учасниками, а також для запобігання спробам взаємодії з неавторизованими або відключеними користувачами. Крім того, цей спостерігач дозволяє підтримувати актуальний перелік активних користувачів і контролювати їхню присутність у системі.

Спостерігач NEW_USER відповідає за обробку подій, пов'язаних із підключенням нових користувачів. Після встановлення нового з'єднання сервер реєструє користувача в системі та надсилає відповідні повідомлення іншим учасникам кімнати або чату. Завдяки цьому клієнтська частина отримує актуальну інформацію про появу нового співрозмовника та може автоматично оновити інтерфейс користувача. Також цей спостерігач бере участь у процесі синхронізації інформації між учасниками під час створення

нових сеансів зв’язку.

Спостерігач LOGOUT призначений для контролю завершення роботи користувача із системою. У момент виходу користувача сервер фіксує факт розірвання з’єднання, оновлює інформацію про його статус та надсилає відповідні повідомлення іншим учасникам активного сеансу. Це дозволяє клієнтським застосункам своєчасно відобразити зміну стану співрозмовника, завершити активні з’єднання та виконати очищення пов’язаних ресурсів. Робота цього спостерігача забезпечує підтримання актуального стану системи та коректне завершення комунікаційних сеансів.

Спостерігач CHECK_CHANNEL виконує перевірку наявності каналу обміну текстовими повідомленнями між користувачами. Після отримання запиту сервер аналізує існуючі записи бази даних та визначає, чи був раніше створений канал для відповідної пари користувачів. Якщо канал уже існує, система повертає інформацію про нього та забезпечує подальшу роботу через наявний механізм обміну повідомленнями. У випадку відсутності каналу автоматично створюється новий запис із необхідними параметрами та налаштуваннями. Такий підхід дозволяє забезпечити безперервність комунікації між користувачами та автоматизувати процес організації текстового чату без необхідності додаткових дій з боку учасників.

Загалом використання системи спостерігачів у серверній частині забезпечує ефективну організацію обробки подій у режимі реального часу, що є критично важливим для роботи таких функціональних модулів, як відеозв’язок і текстовий обмін повідомленнями. Така архітектура дозволяє централізовано керувати потоками подій, оперативно реагувати на дії користувачів та забезпечувати стабільну взаємодію між усіма компонентами системи. Крім того, застосування подієво-орієнтованого підходу підвищує масштабованість програмного продукту, оскільки кожен обробник подій реалізує окрему логіку та може бути незалежно модифікований або розширений без впливу на інші частини системи.

На клієнтській стороні реалізовано аналогічний набір спостерігачів,

який за структурою та призначенням відповідає серверній реалізації. Під час ініціалізації сторінки відбувається автоматичне підключення та активація всіх необхідних обробників подій, що забезпечують коректну роботу інтерфейсу користувача. Ці обробники відповідають за відстеження ключових змін у модулі текстових повідомлень, зокрема за появу нових користувачів у чаті, отримання та відображення вхідних повідомлень, оновлення статусу співрозмовників (наприклад, онлайн або офлайн), а також обробку подій розриву або завершення з'єднання. Завдяки цьому клієнтська частина системи підтримує постійну синхронізацію з сервером і забезпечує актуальне відображення стану всіх учасників комунікації в реальному часі.

Окремо слід відзначити, що така симетрія між серверною та клієнтською логікою підвищує узгодженість системи, спрощує її супровід і дозволяє швидше масштабувати функціональність у разі розширення вимог до програмного продукту. Це особливо важливо для застосунків, що працюють із мультимедійними потоками та миттєвими повідомленнями, де затримки або розсинхронізація можуть суттєво впливати на якість користувацького досвіду. Перелік спостерігачів, що активуються під час ініціалізації модуля, наведено на рисунку 3.5.

```
componentDidMount(){
  let { socket } = this.props
  socket.emit(events.INIT_CHATS, this.initChats )
  socket.on( events.MESSAGE_SEND, this.addMessage )
  socket.on( events.TYPING, this.addTyping )
  socket.on( events.P_MESSAGE_SEND, this.addPMessage )
  socket.on( events.P_TYPING, this.addPTyping)
  socket.on( events.CREATE_CHANNEL, this.updateChats )
}
```

Рисунок 3.5 – Перелік спостерігачів на стороні клієнта

3.3 Тестування вебзастосунку

Вебтестування – це процес перевірки вебзастосунку з метою забезпечення його коректної роботи, стабільності та відповідності встановленим вимогам. Основним завданням тестування є виявлення помилок, недоліків або збоїв у роботі системи ще до моменту її впровадження чи під час подальшого використання. Завдяки тестуванню підвищується якість програмного забезпечення, надійність роботи вебзастосунку та зручність його використання для кінцевого користувача.

Під час тестування вебзастосунків перевіряється робота інтерфейсу користувача, взаємодія клієнтської та серверної частин, обробка даних, робота баз даних, API та механізми безпеки. Крім того, тестування дозволяє оцінити продуктивність системи, швидкість її роботи та стійкість до навантажень.

Одним із основних видів перевірки є функціональне тестування. Воно спрямоване на перевірку того, чи відповідає функціонал вебзастосунку поставленим вимогам та очікуваним результатам. У процесі функціонального тестування аналізується робота окремих компонентів системи, правильність обробки даних, коректність переходів між сторінками, робота форм, авторизації, API та інших функцій застосунку.

Функціональне тестування може виконуватися вручну або автоматизовано. Ручне тестування дозволяє оцінити систему з точки зору користувача, тоді як автоматизоване забезпечує швидке повторне виконання тестових сценаріїв та спрощує перевірку великих обсягів функціоналу після внесення змін у програмний код.

Статичне тестування є методом перевірки програмного забезпечення без запуску програми. Під час такого тестування проводиться аналіз вихідного коду, технічної документації, специфікацій та вимог до системи. Основною метою статичного тестування є виявлення помилок на ранніх етапах розробки, що дозволяє зменшити витрати на їх подальше виправлення

та покращити якість програмного продукту.

Динамічне тестування, на відміну від статичного, передбачає виконання програмного коду та перевірку роботи системи у реальних умовах. У процесі тестування аналізується функціональна поведінка вебзастосунку, правильність обробки запитів, використання системних ресурсів і загальна продуктивність програмного забезпечення. Основною метою динамічного тестування є підтвердження того, що система працює відповідно до бізнес-вимог та технічного завдання.

Під час розробки програмного забезпечення також широко використовуються підходи тестування білої та чорної скрині.

Тестування білої скрині базується на аналізі внутрішньої структури програмного забезпечення. У процесі такого тестування перевіряється логіка роботи коду, коректність алгоритмів, шляхи проходження даних та взаємодія окремих модулів системи. Для проведення тестування білої скрині необхідні знання структури програми та навички програмування, оскільки тестові сценарії створюються на основі внутрішньої реалізації програмного забезпечення.

Тестування чорної скрині, навпаки, орієнтоване на перевірку функціональних можливостей системи без аналізу її внутрішньої структури. У даному випадку тестувальник взаємодіє із застосунком як звичайний користувач, перевіряючи відповідність отриманих результатів очікуваній поведінці системи. Основна увага приділяється вхідним та вихідним даним, а також коректності виконання функціональних вимог.

Одним із найважливіших елементів вебзастосунку є головна сторінка, оскільки саме її користувач бачить під час першого входу до системи. Тому особливу увагу необхідно приділити перевірці правильності її відображення, працездатності основних елементів інтерфейсу та швидкості завантаження. Коректна робота головної сторінки безпосередньо впливає на зручність використання застосунку та формує перше враження користувача про систему.

Головну сторінку вебзастосунку «MedSupport», яка відображається користувачу після відкриття системи та забезпечує доступ до основного функціоналу застосунку, наведено на рисунку 3.6.

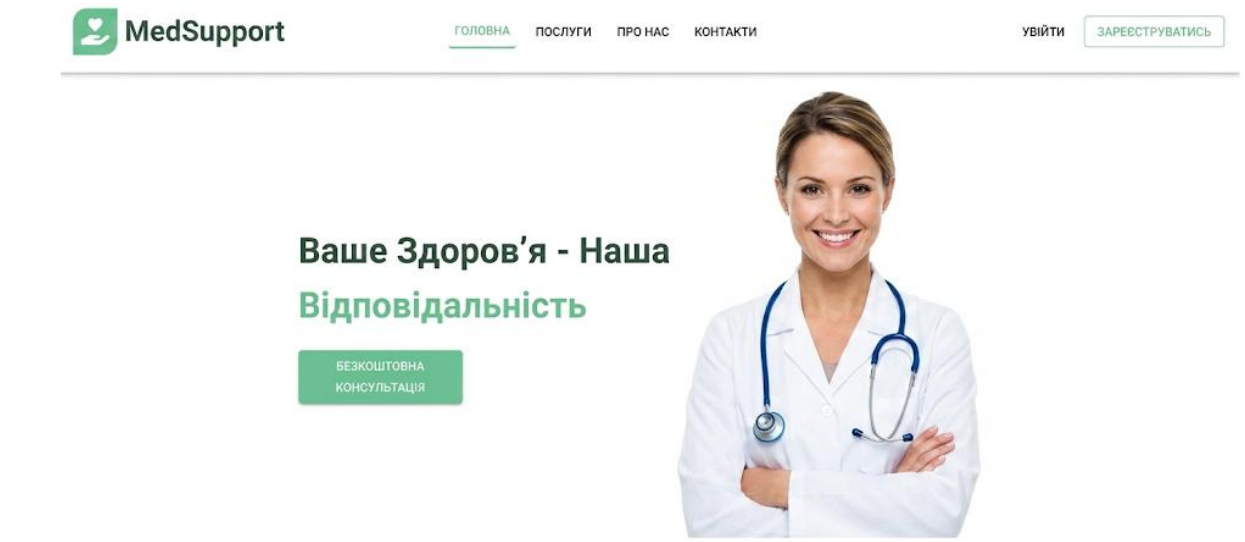

Рисунок 3.6 – Головна сторінка

Форма реєстрації – це вебсторінка, спливаюче вікно або модальне вікно, яке використовується для введення користувачем необхідних даних з метою отримання доступу до функціоналу вебзастосунку. Через цю форму здійснюється первинна ідентифікація користувача та створення його облікового запису в системі.

Обсяг і тип інформації, що збирається під час реєстрації, залежать від призначення сервісу та набору доступних функцій. Зазвичай стандартна форма реєстрації містить обов'язкові поля для введення імені користувача, адреси електронної пошти, логіна та пароля.

У деяких випадках також можуть додатково запитуватися інші дані, необхідні для персоналізації сервісу або підвищення рівня безпеки облікового запису.

Форму реєстрації зображено на рисунку 3.7.

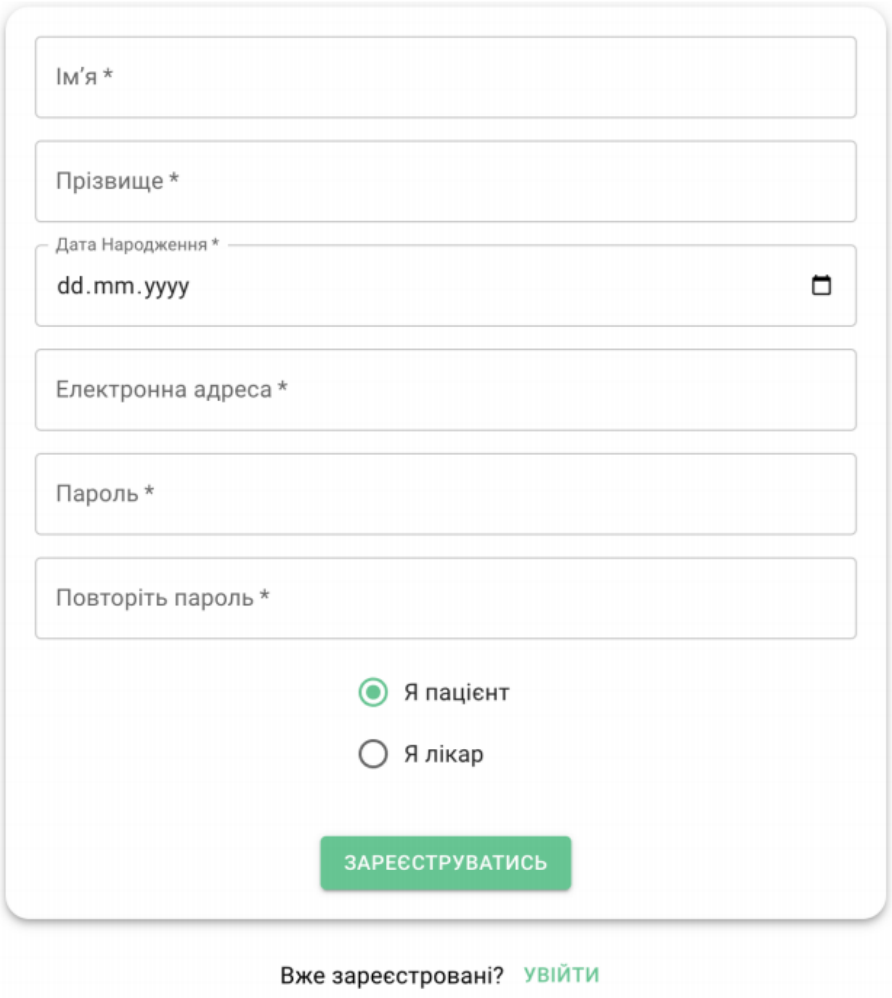

Рисунок 3.7 – Форма реєстрації користувача

Важливим етапом у роботі вебзастосунку є перевірка даних, які вводяться користувачем у форми, оскільки вони можуть містити некоректні, неповні або недопустимі значення. Некоректна обробка таких даних може призвести до помилок у роботі системи, порушення логіки автентифікації або зниження загальної стабільності програмного продукту. Саме тому валідація введеної інформації є обов'язковою складовою забезпечення коректного функціонування системи та безпечної взаємодії користувача з інтерфейсом.

Використання JavaScript для реалізації перевірки форм на стороні клієнта дозволяє здійснювати первинну валідацію без необхідності звернення до серверної частини. Такий підхід значно підвищує швидкість обробки даних, оскільки перевірка виконується безпосередньо в браузері користувача. У результаті користувач отримує миттєвий зворотний зв'язок щодо

коректності введених значень, що покращує зручність використання системи та зменшує кількість помилкових запитів, які надходять на сервер.

Крім того, клієнтська валідація дозволяє суттєво знизити навантаження на серверну частину застосунку, оскільки відсіює значну частину некоректних даних ще до їх надсилання. Це особливо важливо для вебсистем із великою кількістю користувачів, де ефективність обробки запитів має критичне значення. Саме тому клієнтська перевірка даних широко використовується як перший рівень контролю коректності введеної інформації, який часто доповнюється серверною валідацією для забезпечення додаткового рівня безпеки та надійності.За допомогою JavaScript можна здійснювати валідацію різних типів даних, зокрема імені користувача, пароля, адреси електронної пошти, дати, номерів телефонів та інших полів форми. Це дозволяє забезпечити правильність введених даних ще до їх надсилання на сервер і підвищує загальний рівень безпеки та стабільності вебзастосунку.

Форма авторизації користувача призначена для введення облікових даних, необхідних для отримання доступу до функціоналу вебзастосунку. Вона забезпечує процес ідентифікації користувача в системі та перевірку його прав доступу до відповідних ресурсів.

Зазвичай форма входу містить поля для введення імені користувача (логіна) та пароля. Після заповнення цих полів і надсилання форми дані передаються на сервер або перевіряються на стороні клієнта, залежно від реалізації системи.

Далі відбувається перевірка коректності введених облікових даних шляхом їх зіставлення з інформацією, що зберігається в базі даних. У разі успішної автентифікації користувач отримує доступ до функціональних можливостей вебзастосунку, а у випадку помилки – відповідне повідомлення про некоректність введених даних.

Також у системі розроблено панель керування для пацієнта, яка надає користувачеві доступ до основної інформації про його взаємодію з вебзастосунком.

Форму авторизації користувача показано на рисунку 3.8.

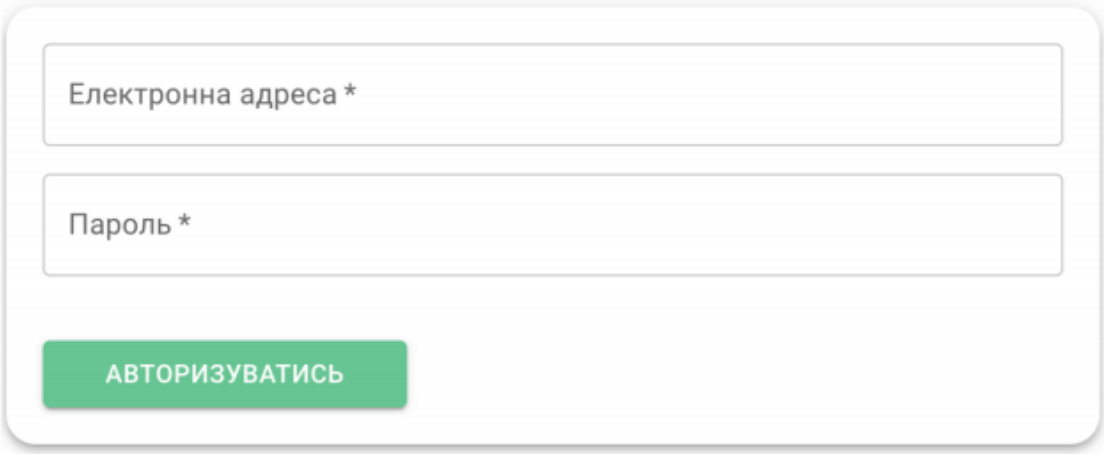

Рисунок 3.8 – Форма авторизації користувача

У цьому розділі пацієнту надається доступ до інформації про найближчу заплановану консультацію. Інтерфейс відображає основні відомості про майбутній сеанс, зокрема дату та час його проведення, що дозволяє користувачеві своєчасно підготуватися до консультації та контролювати власний графік взаємодії з лікарем. Дані автоматично оновлюються відповідно до інформації, що зберігається в системі, забезпечуючи актуальність відображених даних.

Крім перегляду інформації про консультацію, пацієнт має можливість приєднатися до відеокімнати, створеної лікарем для проведення онлайн-прийому. Після активації відповідної функції система підключає користувача до сеансу відеозв’язку, що забезпечує безпосередню комунікацію між пацієнтом і медичним спеціалістом у режимі реального часу. Такий підхід дозволяє організувати дистанційне надання консультацій без необхідності особистого відвідування медичного закладу.

Панель керування є центральним елементом взаємодії пацієнта із системою та забезпечує швидкий доступ до найважливіших функцій вебзастосунку. Завдяки інтуїтивно зрозумілому інтерфейсу користувач може оперативно отримувати необхідну інформацію про заплановані консультації, контролювати їхній статус і підключатися до онлайн-сеансів. Це підвищує зручність використання системи та сприяє ефективній взаємодії між пацієнтом і лікарем у дистанційному форматі.

Приклад реалізації панелі керування пацієнта наведено на рисунку 3.9.

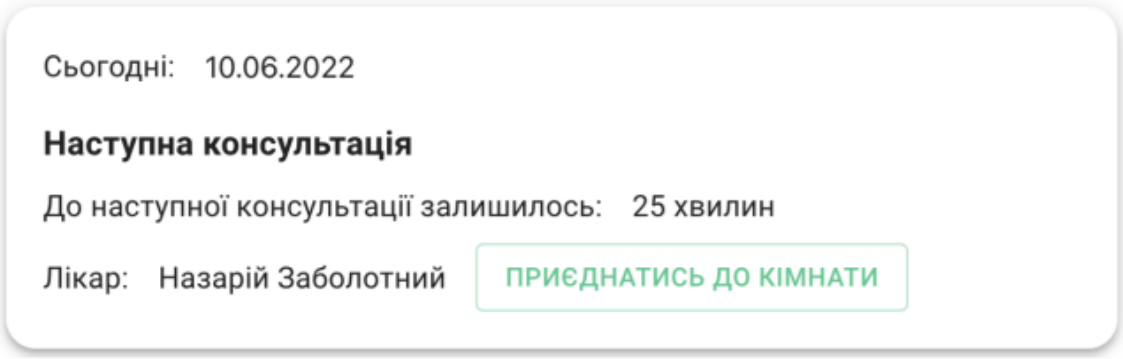

Рисунок 3.9 – Панель керування пацієнта

Після створення кімнати лікар автоматично переходить до приватного каналу відеозв’язку, де очікує підключення пацієнта. У цьому режимі користувач має доступ до власного відеопотоку, що дозволяє контролювати зображення з камери, а також керувати основними параметрами дзвінка. Зокрема, лікар може вимкнути або увімкнути камеру, активувати чи деактивувати мікрофон, відкрити чат для текстового спілкування та завершити сеанс, закривши доступ до кімнати.

Після успішного встановлення з’єднання між учасниками система автоматично відкриває інтерфейс модуля відеозв’язку, який забезпечує проведення онлайн-консультації в режимі реального часу. Даний інтерфейс є основним робочим середовищем для взаємодії лікаря та пацієнта під час сеансу зв’язку та надає доступ до всіх необхідних засобів комунікації.

Центральним елементом інтерфейсу є область відображення відеопотоку співрозмовника, що дозволяє користувачеві спостерігати за учасником консультації та підтримувати повноцінне візуальне спілкування. Одночасно система відображає локальне відео користувача, отримане з вебкамери пристрою. Це дає можливість контролювати якість передаваного зображення та переконатися у правильності роботи обладнання під час сеансу зв’язку.

Крім відеопотоків, інтерфейс містить панель керування дзвінком, яка забезпечує доступ до основних функцій керування сеансом. За допомогою відповідних елементів користувач може вмикати або вимикати мікрофон, керувати передачею відео, завершувати поточний сеанс зв’язку та виконувати інші дії, необхідні для комфортної роботи з модулем відеоконференцій. Наявність таких засобів керування підвищує зручність використання системи та дозволяє швидко реагувати на зміни під час проведення консультації.

Використання технології WebRTC забезпечує передачу аудіо- та відеоданих безпосередньо між учасниками з мінімальними затримками, що є особливо важливим для медичних консультацій, де необхідна оперативна комунікація між лікарем і пацієнтом. Завдяки цьому користувачі можуть здійснювати повноцінне спілкування, обговорювати результати обстежень, отримувати рекомендації та ставити запитання в умовах, максимально наближених до особистої зустрічі.

Таким чином, розроблений модуль відеозв’язку забезпечує надійну та ефективну організацію дистанційної взаємодії між лікарем і пацієнтом. Використання сучасних технологій передачі мультимедійних даних у режимі реального часу дозволяє реалізувати якісний аудіо- та відеозв’язок із мінімальними затримками, що є важливою умовою для проведення медичних консультацій у дистанційному форматі.

Інтерфейс модуля надає користувачам усі необхідні інструменти для комфортного спілкування, включаючи перегляд відеопотоків

співрозмовників та засоби керування сеансом зв’язку. Завдяки цьому забезпечується ефективний обмін інформацією між учасниками консультації, що підвищує якість надання медичних послуг та спрощує процес комунікації.

Запропоноване рішення дозволяє проводити онлайн-консультації незалежно від місцезнаходження користувачів, забезпечуючи доступність медичної допомоги та створюючи зручні умови для взаємодії між пацієнтом і лікарем у межах вебзастосунку.

Приклад роботи модуля відеозв’язку наведено на рисунку 3.10.

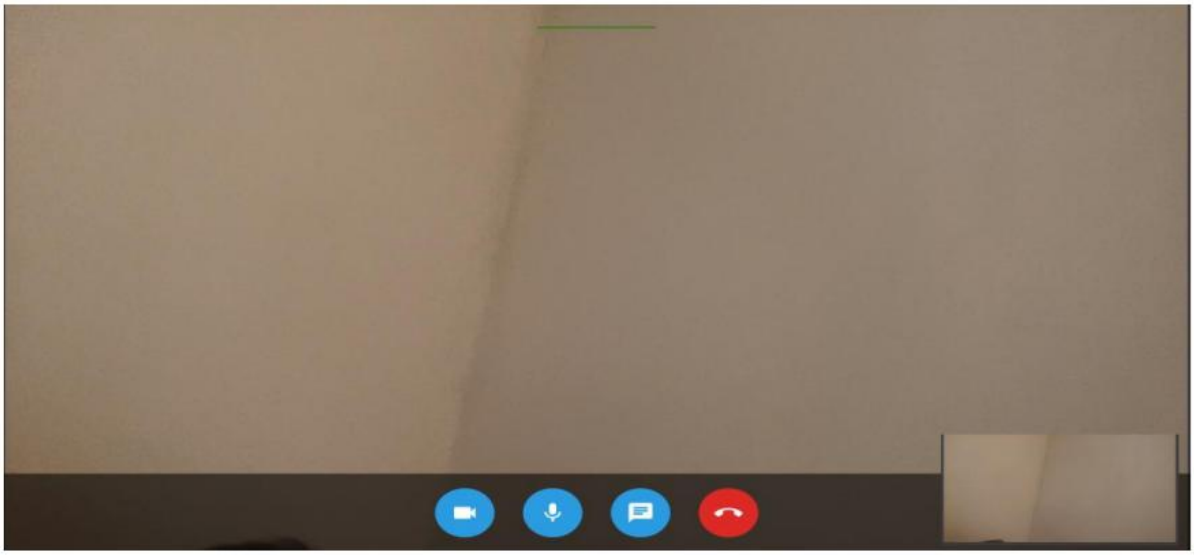

Рисунок 3.10 – Робота модуля відеозв’язку

Кнопка «Відеокамера» призначена для керування станом відеопотоку користувача в межах активного сеансу відеозв’язку. Вона забезпечує можливість швидко вмикати або вимикати передачу відеозображення з вебкамери залежно від потреб користувача. Після натискання кнопки відбувається перемикання стану відеокамери: якщо вона була активована, система припиняє передачу відеопотоку, а якщо вимкнена – повторно ініціює його трансляцію в межах поточного peer-to-peer з’єднання.

Окрему увагу приділено інтерфейсному відображенню стану функції. Іконка кнопки динамічно змінюється відповідно до поточного статусу відеокамери, що дозволяє користувачеві візуально контролювати, чи

ведеться передача відео в цей момент. Такий підхід підвищує зручність використання системи та зменшує ймовірність випадкового вимкнення або увімкнення камери без усвідомлення користувача.

Крім того, керування відеопотоком реалізовано таким чином, щоб не порушувати загальний сеанс зв'язку: навіть у разі вимкнення камери аудіозв'язок та інші канали взаємодії продовжують функціонувати. Це забезпечує гнучкість у використанні модуля відеозв'язку та дозволяє адаптувати режим спілкування до поточних потреб користувача.

Кнопка мікрофона використовується для керування аудіопідключенням. Її функціонал полягає у вмиканні або вимиканні мікрофона під час відеодзвінка. Після натискання стан мікрофона змінюється на протилежний, а іконка кнопки оновлюється відповідно до поточного режиму роботи, що забезпечує зручність користування та зрозумілий інтерфейс.

Кнопка чату призначена для керування відображенням вікна текстових повідомлень під час активного сеансу відеозв'язку. Її використання дозволяє користувачеві відкривати або приховувати чат без необхідності завершувати основний відеодзвінок. У відкритому стані користувач отримує доступ до історії повідомлень із співрозмовником, а також може обмінюватися текстовими повідомленнями в режимі реального часу. Такий функціонал забезпечує паралельну комунікацію, що є особливо важливим під час проведення онлайн-консультацій, коли необхідно одночасно підтримувати як аудіо-/відеозв'язок, так і текстову взаємодію.

Кнопка «Завершити виклик» використовується для повного розриву активного з'єднання між учасниками сесії. Після її активації система ініціює процес коректного завершення відеодзвінка, що включає зупинку та закриття всіх активних медіапотоків, розірвання peer-to-peer-з'єднання та очищення пов'язаних ресурсів. У результаті користувач автоматично перенаправляється на сторінку керування або головну панель застосунку, де може переглянути доступні функції або розпочати новий сеанс.

Такий механізм забезпечує правильне завершення сесії та запобігає збереженню неактивних підключень у системі.

Приклад вікна модуля текстових повідомлень зображено на рисунку 3.11.

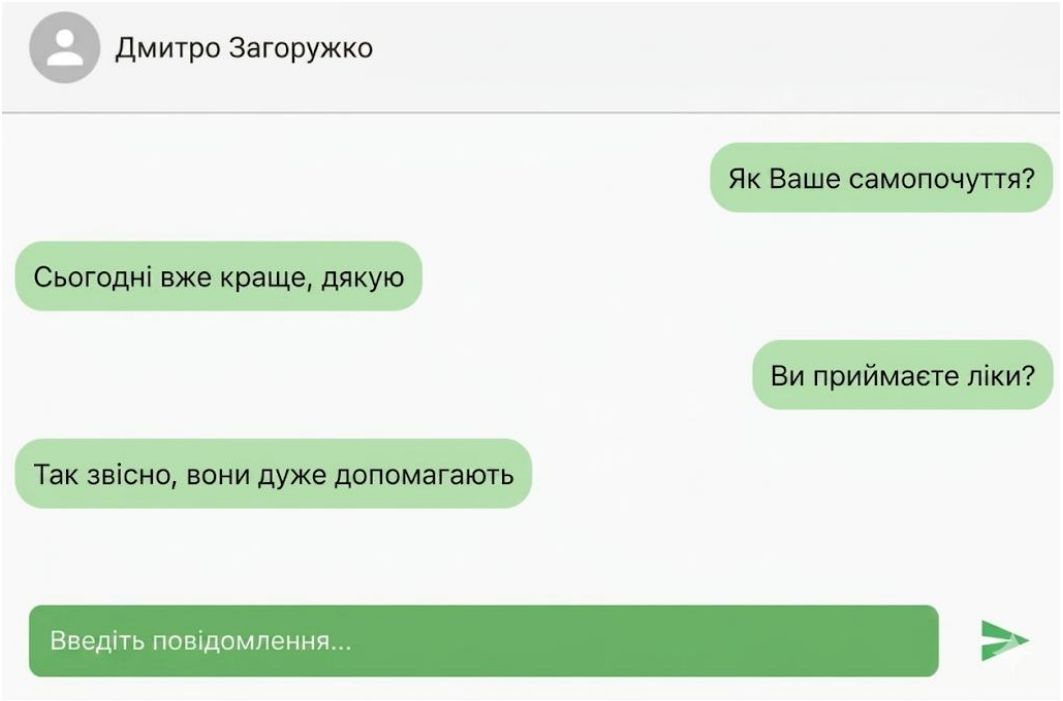

Рисунок 3.11 – Вікно модуля текстових повідомлень

Усі повідомлення про помилки у програмному продукті реалізовані у вигляді спливаючих сповіщень, що забезпечують оперативне та зрозуміле інформування користувача про виникнення некоректних дій або помилок під час взаємодії із системою. Такий підхід дозволяє швидко привернути увагу користувача до проблеми та надати йому необхідну інформацію для її усунення.

Валідація полів введення у вебзастосунку реалізована коректно та виконується на стороні клієнта ще до моменту надсилання даних на сервер. Такий підхід дозволяє здійснювати первинну перевірку інформації, введеної користувачем, без необхідності додаткових серверних запитів, що суттєво підвищує швидкість реагування системи. У процесі валідації перевіряється

відповідність даних заданим форматам, обов’язковість заповнення полів, а також допустимість введених значень.

Завдяки реалізації клієнтської перевірки система здатна своєчасно виявляти помилки введення та блокувати відправлення некоректних або неповних даних до серверної частини. Це дозволяє зменшити кількість невалідних запитів, які надходять на сервер, і, відповідно, знизити навантаження на серверну інфраструктуру.

Окрім цього, користувач отримує миттєвий зворотний зв’язок щодо правильності введеної інформації, що підвищує зручність використання інтерфейсу та покращує загальний користувацький досвід. Повідомлення про помилки зазвичай відображаються у зрозумілому форматі безпосередньо у полі введення, що дозволяє швидко виправити допущені неточності.

У сукупності ці механізми сприяють підвищенню загальної стабільності програмного забезпечення, зменшенню ймовірності виникнення помилок на серверній стороні та забезпеченню цілісності й достовірності даних, оброблюваних у системі.

3.4 Підготовка опису порядку роботи користувача із вебзастосунком

Інструкція користувача містить детальний перелік технічних вимог, необхідних для коректного запуску програмного продукту, а також описує послідовність дій щодо розгортання та запуску серверної і клієнтської частин вебзастосунку. У документі наведено основні етапи підготовки програмного середовища, включаючи встановлення необхідних залежностей, налаштування служб і запуск основних компонентів системи, що забезпечує її стабільне функціонування.

Відомості щодо мінімальної та рекомендованої конфігурації обчислювальної системи, необхідної для роботи програмного забезпечення, подано у таблиці 3.1.

Дана таблиця містить параметри апаратного та програмного забезпечення, які забезпечують коректну роботу застосунку в різних умовах використання, а також дозволяють оцінити вимоги системи до ресурсів для його ефективної експлуатації.

Таблиця 3.1 – Конфігурації обчислювальної систе

	Мінімальні системні вимоги	**Рекомендована конфігурація**
Тип процесора	32-розрядний (x86), 64-розрядний (x64) або ARM процесор з тактовою частотою 1 ГГц	32-розрядний (x86), 64-розрядний (x64) або ARM процесор з тактовою частотою 2.4 ГГц
Об’єм оперативної пам’яті	2 ГБ	4 ГБ
Місце на жорсткому диску	600 МБ	1 ГБ
Операційна система	Ubuntu 16.04/ Windows 7/ MacOS High Sierra	Ubuntu 21.04/ Windows 10/ MacOS Monterey

Для запуску програмного продукту необхідно попередньо встановити платформу Docker і менеджер пакетів NPM. Після цього за допомогою Docker слід завантажити образ системи керування чергами повідомлень Redis і створити на його основі контейнер. Запущений контейнер забезпечує роботу сервісу Redis, який додатком використовується для обробки та передачі повідомлень між його компонентами.

Redis є високопродуктивною системою зберігання даних у пам’яті з відкритим вихідним кодом, що розповсюджується за ліцензією BSD та реалізована мовою програмування C. Завдяки орієнтації на швидке опрацювання даних Redis широко використовується як база даних, система кешування та брокер повідомлень. Цю технологію часто називають сервером

структур даних, оскільки вона підтримує різноманітні типи даних, аналогічні тим, що використовуються в сучасних мовах програмування, зокрема рядки, списки, хеш-таблиці, множини та впорядковані множини. Крім базових структур, Redis надає засоби для роботи з геопросторовими даними, потоками подій, механізмами приблизного підрахунку та іншими інструментами, що розширюють його функціональні можливості й дозволяють застосовувати систему в широкому спектрі програмних рішень.

Серед NoSQL-систем Redis вирізняється тим, що його структури даних максимально наближені до стандартних структур, які програмісти використовують під час розробки програмного забезпечення та реалізації алгоритмів. Завдяки цьому робота з Redis є інтуїтивно зрозумілою та не потребує складних механізмів перетворення даних. Такий підхід спрощує процес розробки, прискорює створення програмних продуктів і забезпечує високу продуктивність застосунків. Крім того, підтримувані Redis структури даних можуть ефективно використовуватися для обміну інформацією між різними процесами та сервісами, що робить цю систему зручним інструментом для побудови розподілених програмних рішень.

За замовчуванням Redis зберігає дані в оперативній пам’яті, забезпечуючи високу швидкість доступу до них, а також підтримує механізми періодичного збереження даних на диску. Завдяки цьому систему можна використовувати як повноцінну базу даних для багатьох прикладних задач або як високопродуктивний кеш.

У разі досягнення встановленого ліміту пам’яті Redis може або повертати повідомлення про помилку під час спроби запису нових даних, або працювати в режимі кешування з автоматичним видаленням менш важливих чи застарілих записів для звільнення місця під нові дані. Незалежно від обраного режиму роботи, ключовим фактором, що визначає можливості використання Redis, залишається обсяг доступної оперативної пам’яті.

NPM (Node Package Manager) є стандартним менеджером пакетів для платформи Node.js, який використовується для встановлення, оновлення та

керування програмними залежностями. Він містить один із найбільших у світі репозиторіїв програмного забезпечення, що об’єднує мільйони пакетів і бібліотек, доступних розробникам.

Завдяки NPM розробники можуть швидко інтегрувати готові програмні компоненти у власні проєкти, автоматизувати процес керування залежностями та спрощувати розробку програмного забезпечення. Відкрита екосистема NPM дозволяє програмістам з усього світу публікувати власні пакети, поширювати вихідний код і використовувати напрацювання інших учасників спільноти, що сприяє прискоренню розробки сучасних вебзастосунків.

Після завантаження необхідного образу наступним етапом є запуск контейнера Redis за допомогою графічного інтерфейсу Docker. Для цього потрібно перейти до списку доступних контейнерів, обрати контейнер Redis і запустити його. Після успішного запуску контейнер переходить у робочий стан і стає доступним для взаємодії з іншими компонентами програмного продукту, забезпечуючи зберігання даних та обмін повідомленнями між сервісами системи.

Після запуску контейнера Redis необхідно перейти до каталогу server, у якому розміщена серверна частина проєкту. У командному рядку слід виконати команду npm install, яка встановлює всі необхідні залежності, визначені у файлі конфігурації проєкту. Після завершення процесу встановлення потрібно виконати команду npm start, що ініціює запуск серверної частини вебзастосунку. У результаті сервер буде готовий до обробки запитів та взаємодії з іншими компонентами системи.

Після успішного запуску серверної частини необхідно перейти до каталогу client, який містить клієнтську частину вебзастосунку. Аналогічно до попереднього етапу слід виконати команду npm install для встановлення всіх необхідних залежностей, а потім команду npm start для запуску клієнтської частини системи. Після завершення цього процесу вебзастосунок буде повністю розгорнутий і готовий до використання через веббраузер.

Після запуску системи користувач повинен пройти процедуру автентифікації, ввівши свої облікові дані. Після успішного входу до системи він буде автоматично перенаправлений на сторінку керування, де відображатиметься інформація про найближчу заплановану консультацію. На цій сторінці користувач має можливість створити кімнату для взаємодії з лікарем або пацієнтом. Після створення та підтвердження сеансу система перенаправляє користувача на сторінку комунікації, яка містить модулі відеозв'язку та обміну текстовими повідомленнями, що забезпечують проведення онлайн-консультації в режимі реального часу.

Одним із ключових етапів розробки програмного забезпечення є тестування, оскільки воно дає змогу своєчасно виявити помилки, недоліки та можливі збої в роботі системи ще до її впровадження в експлуатацію. Проведення тестування дозволяє перевірити коректність функціонування всіх компонентів додатку, оцінити його стабільність та відповідність встановленим вимогам. Усунення виявлених недоліків на етапі тестування сприяє підвищенню надійності програмного продукту, покращенню користувацького досвіду та формуванню позитивної оцінки системи з боку кінцевих користувачів.

У процесі тестування було підтверджено коректність роботи програмного продукту. Результати виконання всіх тестових сценаріїв продемонстрували повну відповідність між вхідними даними та очікуваними вихідними результатами. Під час перевірки функціональних можливостей системи помилок, збоїв або некоректної поведінки не виявлено, що свідчить про стабільність роботи вебзастосунку та його готовність до експлуатації.

Крім того, було розроблено інструкцію користувача, яка містить опис вимог до програмного середовища, порядок запуску системи та рекомендації щодо її використання. В інструкції наведено необхідні вхідні дані, а також детально описано послідовність дій, які користувач повинен виконати для коректної роботи з програмним продуктом.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та реалізовано вебзастосунок для надання медичних онлайн-консультацій.

Робота охоплює повний цикл створення застосунку – від аналізу аналогічних рішень до реалізації функціоналу із використанням сучасних інструментів, та вирішено такі завдання:

– проаналізовано наявні на ринку програмні аналоги та здійснено їх порівняння з розроблюваним вебзастосунком. У межах проведеного аналізу було розглянуто функціональні можливості подібних рішень, їхні архітектурні особливості, переваги та обмеження, а також визначено ступінь їхньої відповідності сучасним вимогам до систем дистанційної взаємодії користувачів;

– сформовано стратегію забезпечення послідовного виконання кроків алгоритму, яка передбачає поетапну обробку даних із контролем коректності кожної операції;

– розроблено схеми інтерфейсів модулів відеозв’язку та текстових повідомлень. У межах цього процесу було визначено основні елементи користувацького інтерфейсу, їх розташування та логіку взаємодії з користувачем, що дозволило забезпечити узгодженість між різними модулями системи, а також створити інтуїтивно зрозумілий та зручний інтерфейс для кінцевого користувача, який сприяє ефективному використанню функціональних можливостей вебзастосунку;

– розглянуто різні стратегії тестування та визначено, що для розробленого застосунку найбільш доцільним є метод «чорної скриньки». Даний підхід дозволяє оцінювати роботу системи з позиції кінцевого користувача, без аналізу внутрішньої реалізації коду. Завдяки цьому тестування максимально наближене до реальних умов експлуатації та охоплює основні сценарії використання програмного продукту, що дає змогу виявити можливі помилки в його роботі;

– розроблено вебзастосунок з урахуванням проведеного аналізу;

– розроблено інструкцію користувача, яка містить опис вхідних даних, необхідних для коректного функціонування програмного продукту, а також послідовність дій, які повинен виконати користувач під час роботи із системою для забезпечення її правильного використання.

На основі отриманих результатів обґрунтовано доцільність розробки власного програмного продукту, який дозволяє усунути виявлені недоліки та підвищити ефективність користування системою.

У ході роботи було обґрунтовано вибір методів розробки та сформовано вимоги до програмного продукту, спроєктовано структуру та алгоритми функціонування вебзастосунку для онлайн-консультацій, реалізовано основні програмні модулі системи, а також проведено їх тестування.

Перспективи розвитку вебзастосунку полягають у розширенні його функціональних можливостей шляхом удосконалення механізмів штучного інтелекту, впровадження персоналізованих рекомендацій та інтеграції з електронними медичними системами. У майбутньому можливе створення мобільної версії застосунку, додавання багатомовної підтримки та розширення можливостей онлайн-консультацій. Це сприятиме підвищенню доступності медичних послуг, покращенню якості взаємодії між користувачами та медичними фахівцями, а також подальшому розвитку цифрової медицини.

Результати роботи апробовано у вигляді двох тез доповідей:

– під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [35];

– під час X Всеукраїнської мультидисциплінарної студентської наукової конференції «Розвиток сучасної науки: актуальні питання теорії та практики» [36].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hunt, E. B. (2014). *Artificial intelligence*. Academic Press, 484.
2. Holmes, J., Sacchi, L., & Bellazzi, R. (2004). Artificial intelligence in medicine. *Ann R Coll Surg Engl*, *86*(86), 334-8.
3. Ertel, W. (2024). *Introduction to artificial intelligence*. Springer Nature, 382.
4. Шафроненко, А., Бодянський, Є., & Плісс, І. (2022). Нечіткі методи інтелектуального аналізу даних. *United Kingdom, London. GlobeEdit*, *104*.
5. Floridi, L., & Chiriatti, M. (2020). GPT-3: Its nature, scope, limits, and consequences. *Minds and machines*, *30*(4), 681-694.
6. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., & Tang, J. (2024). GPT understands, too. *AI open*, *5*, 208-215.
7. Котенко, О., Шафроненко, А., Сенніков, І., & Гаврилов, А. (2024). ОГЛЯД МЕТОДІВ ПІДТРИМКИ ПЕРСОНАЛІЗОВАНИХ МЕДИЧНИХ ДАНИХ. *Матеріали конференцій МЦНД*, (01.11. 2024; Хмельницький,, Україна), 455-457.
8. Шафроненко, А. Ю., Бодянський, Є. В., Руденко, Д. О., & Шафроненко, Є. О. (2026). ОНЛАЙН БЕГГІНГ В ЗАДАЧАХ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ. *Збірник наукових праць Харківського національного університету Повітряних Сил*, (1 (87)), 84-89.
9. Шафроненко, А. Ю., Бодянський, Є. В., Шейко, С. О., & Шафроненко, Є. О. (2026). АДАПТИВНА РОБАСТНА НЕЧІТКА КЛАСТЕРИЗАЦІЯ ПОТОКІВ ДАНИХ НА ОСНОВІ ПРАВДОПОДІБНОГО ПІДХОДУ. *Системи обробки інформації*, (1 (184)), 112-115.
10. Shafronenko, A. Y., Kasatkina, N. V., Bodyanskiy, Y. V., & Shafronenko, Y. O. (2023). Credibilistic robust online fuzzy clustering in data stream mining tasks. *Radio Electronics, Computer Science, Control*, (3), 97-97.
11. Шафроненко, А. Ю., Бодянський, Є. В., & Руденко, Д. О. (2023). Модифікований рекурентний метод достовірної нечіткої кластеризації з

використанням оптимізаційної процедури на основі косяків риб. *Системи обробки інформації*, (1 (172)), 92-96.

12. Bodyanskiy, Y., Pliss, I., & Shafronenko, A. (2022). Adaptive neurofuzzy clustering of distorted data based on prototype-centroid strategy using evolutionary procedures. *Artificial Intelligence*, *27*, 239-244.

13. Samuel, A. L. (1959). Machine learning. *The Technology Review*, *62*(1), 42-45.

14. Waisberg, E., Ong, J., Masalkhi, M., Zaman, N., Sarker, P., Lee, A. G., & Tavakkoli, A. (2024). Google's AI chatbot "Bard": a side-by-side comparison with ChatGPT and its utilization in ophthalmology. *Eye*, *38*(4), 642-645.

15. Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H. T., ... & Le, Q. (2022). Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

16. Gwon, Y. N., Kim, J. H., Chung, H. S., Jung, E. J., Chun, J., Lee, S., & Shim, S. R. (2024). The use of generative AI for scientific literature searches for systematic reviews: ChatGPT and Microsoft Bing AI performance evaluation. *JMIR Medical Informatics*, *12*, e51187.

17. Sredojev, B., Samardzija, D., & Posarac, D. (2015, May). WebRTC technology overview and signaling solution design and implementation. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1006-1009). IEEE.

18. Murley, P., Ma, Z., Mason, J., Bailey, M., & Kharraz, A. (2021, April). Websocket adoption and the landscape of the real-time web. In *Proceedings of the Web Conference 2021* (pp. 1192-1203).

19. Ehsan, A., Abuhaliqa, M. A. M., Catal, C., & Mishra, D. (2022). RESTful API testing methodologies: Rationale, challenges, and solution directions. *Applied Sciences*, *12*(9), 4369.

20. Singh, K. U., Varshney, N., Gupta, P., Kumar, G., Singh, T., & Dogiwal, S. R. (2024, February). Mobile application control with firebase cloud messaging.

In *International Conference On Innovative Computing And Communication* (pp. 527-535). Singapore: Springer Nature Singapore.

21. Hanchett, E., & Listwon, B. (2018). *Vue. js in Action*. Simon and Schuster.

22. Pham, D. Q. (2025). A comparative study of React and Vue. js in single page website (SPW) development.

23. Green, B., & Seshadri, S. (2013). *AngularJS*. " O'Reilly Media, Inc.".

24. Lei, K., Ma, Y., & Tan, Z. (2014, December). Performance comparison and evaluation of web development technologies in php, python, and node. js. In *2014 IEEE 17th international conference on computational science and engineering* (pp. 661-668). IEEE.

25. Python, W. (2021). Python. *Python releases for windows*, *24*.

26. Pigeaud, T. G. T. (2013). *Literature of java*. Springer Science & Business Media.

27. Talekar, S. (2008). *WebStorm: Web based support tool for organization of requirements modeling*. University of Nevada, Reno.

28. Welling, L., & Thomson, L. (2003). *PHP and MySQL Web development*. Sams publishing.

29. Johnson, B. (2019). *Visual studio code: end-to-end editing and debugging tools for web developers*. John Wiley & Sons.

30. Герасимова, С. С. (2024). Розробка застосунку для онлайн розпізнавання облич у відеопотоці.

31. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Irina, K. (2021, September). Online credibilistic fuzzy clustering method based on cauchy density distribution function. In *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)* (pp. 704-707). IEEE.

32. Bodyanskiy, Y. V., Shafronenko, A., & Klymova, I. (2021, April). Adaptive Recovery of Distorted Data Based on Credibilistic Fuzzy Clustering Approach. In *COLINS* (pp. 6-15).

33. Bodyanskiy, Y., Shafronenko, A., & Volkova, V. (2012). Adaptive fuzzy probabilistic clustering of incomplete data. *INFORMATION MODELS & ANALYSES, 112*.

34. Shafronenko, A., Bodyanskiy, Y. V., Klymova, I., & Holovin, O. (2020, May). Online credibilistic fuzzy clustering of data using membership functions of special type. In *CMIS* (pp. 744-753).

35. Лазаренко Т.Б. (2026) Проєктування та реалізація інформаційної системи для медичних онлайн-консультацій. 30-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті». Зб. матеріалів форуму. Т. 7. Харків: ХНУРЕ. 2026. 92-93

36. Лазаренко Т.Б. (2026) Розробка вебзастосунка для дистанційного консультування хворих. X Всеукраїнська мультидисциплінарна студентська наукова конференція «Розвиток сучасної науки: актуальні питання теорії та практики». м. Тернопіль. 2026. 450-451.