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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: розробка веб-застосунку для комунікації в режимі реального часу з використанням текстового, аудіо та відео-чату

Виконав(ла): студент(ка) 4 курсу, групи СП-42
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

		Швець Н. В.
	(підпис)	(прізвище та ініціали)
Керівник		Кишкевич О. О.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Стоянов Ю. М.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Петрик М. Р.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2024

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

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

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

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

студенту Швець Назарію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-застосунку для комунікації в режимі реального часу з використанням текстового, аудіо та відео-чату

Керівник роботи Кишкевич О. О.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*15*» *квітня* 2024 року № *4/7-342*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи мова програмування TypeScript, фреймворки React, TailwindCSS, Express.js, середовище розробки WebStorm

4. Зміст роботи (перелік питань, які потрібно розробити)
Проаналізувати предметну область, спроєктувати, розробити та протестувати вебзастосунок для комунікації в режимі реального часу з використанням текстового, аудіо та відеочату

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Діаграми варіантів використання, зв'язків, діяльності та послідовності, зображення графічного інтерфейсу користувача та результатів тестування, слайди презентації

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми роботи		
2	Аналіз предметної області		
3	Проєктування загальної структури системи		
4	Проєктування структур окремих модулів		
5	Програмна реалізація вебзастосунка		
6	Тестування розробленої системи		
7	Виконання розділу з безпеки життєдіяльності та основ		
	охорони праці		
8	Оформлення пояснювальної записки		
9	Попередній захист		
10	Нормоконтроль та рецензування		
11	Захист роботи		

Студент _____ Швець Н. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кишкевич О. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра на тему «Розробка веб-застосунку для комунікації в режимі реального часу з використанням текстового, аудіо та відео-чату» виконана Швець Назарієм Володимировичем, студентом Тернопільського національного технічного університету імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42.

Швець Н. В. Розробка веб-застосунку для комунікації в режимі реального часу з використанням текстового, аудіо та відео-чату : кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю „121 — інженерія програмного забезпечення“ / Назарій Володимирович Швець. — Тернопіль: ТНТУ, 2024. — 63 с.

Відомості про обсяг: сторінок – 63, рисунків – 32, таблиць – 2, частин – 4, додатків – 1, посилань – 21.

Метою даної роботи є розробка вебзастосунку, що дозволяє користувачам здійснювати текстову, аудіо та відео комунікацію в режимі реального часу. Для досягнення цієї мети було використано технології MERN (MongoDB, Express.js, React, Node.js) та сервіси Agora. Запропоноване рішення включає розробку бази даних, клієнтської та серверної частини, що забезпечують обробку запитів і передачу даних в реальному часі.

Результатом роботи стала система, яка дозволяє користувачам ефективно спілкуватися незалежно від їхнього місцезнаходження. Вебзастосунок забезпечує високу надійність і продуктивність, відповідає сучасним стандартам якості та має зручний інтерфейс користувача.

ABSTRACT

Bachelor’s qualification work on the topic “Development of a Real-Time Communication Web Application Using Text, Audio, and Video Chat” was written by Shvets Nazarii Volodymyrovych, a student of Ivan Puluj Ternopil National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, group SP-42.

Shvets, N. V. (2024). Development of a Real-Time Communication Web Application Using Text, Audio, and Video Chat: Bachelor's Qualification Work for Obtaining the Educational Degree of Bachelor in the Specialty “121 - Software Engineering” / Nazarii Volodymyrovych Shvets. - Ternopil: TNTU. - 63 p.

Information about the scope: pages - 63, figures - 32, tables - 2, sections - 4, appendices - 1, references - 21.

The aim of this work is to develop a web application that allows users to perform text, audio, and video communication in real-time. To achieve this goal, MERN technologies (MongoDB, Express.js, React, Node.js) and Agora services were utilized. The proposed solution includes the development of a database, client-side, and server-side parts that ensure real-time request processing and data transmission.

The result of the work is a system that allows users to communicate effectively regardless of their location. The web application provides high reliability and performance, meets modern quality standards, and features a user-friendly interface.

ЗМІСТ

ВСТУП

У сучасному світі інформаційні технології розвиваються надзвичайно швидкими темпами, вносячи свій внесок у зміни в різних сферах життя, в тому числі і в комунікації. З появою інтернету та мобільних технологій комунікація стала набагато зручнішою та швидшою, дозволяючи людям взаємодіяти незалежно від місця розташування. Інтернет забезпечив миттєву передачу інформації між користувачами з різних куточків світу, створивши форму спілкування, яка раніше була немислима. Технологічний прогрес значно розширив горизонти взаємодії – від електронної пошти до соціальних мереж.

В останні роки особливої популярності набули вебзастосунки для спілкування в режимі реального часу, що пропонують можливість обміну текстовими повідомленнями, аудіо та відеодзвінками. Такі платформи, як Slack, Microsoft Teams, Zoom та Discord стали важливими інструментами для особистого спілкування, бізнесу та освіти [1,2,3,4]. Користувачі можуть не тільки обмінюватися миттєвими повідомленнями, але й проводити відеоконференції, співпрацювати над проєктами та проводити онлайн-зустрічі. Завдяки цьому люди можуть працювати, вчитися та підтримувати соціальні зв’язки з будь-якої точки світу.

Тема даної кваліфікаційної роботи присвячена розробці вебзастосунка для спілкування в режимі реального часу з використанням текстового, аудіо та відеочату. Такий застосунок не лише полегшить процес комунікації між користувачами, але й забезпечить багатофункціональність, включаючи можливості для спільного використання екрана, налаштування профілів та багато іншого.

Реалізація такого застосунку передбачає використання сучасного технологічного стека MERN (MongoDB, Express.js, React, Node.js), за допомогою якого можна створювати високопродуктивні, масштабовані та зручні у використанні вебпрограми, які відповідають останнім вимогам ринку, та

інтеграцію сервісів Agora для забезпечення якісного зв’язку, стабільної та високоякісної передачі аудіо та відеоданих.

Отже, розробка вебдодатка для спілкування в режимі реального часу є актуальною і перспективною темою, що відповідає сучасним тенденціям розвитку інформаційних технологій і задовольняє потреби користувачів в зручному і багатофункціональному засобі комунікації.

Актуальність теми. З розвитком дистанційної роботи, онлайн-навчання та зростанням попиту на зручні та функціональні засоби комунікації, створення подібних вебзастосунків стає все більш актуальним [5]. Відповідно, дана тема є надзвичайно важливою та перспективною.

Метою даної роботи є розробка вебдодатка для спілкування в режимі реального часу, що надає текстовий, аудіо та відеочат, а також додаткові функції, такі як спільне використання екрана та інтерактивну дошку для малювання. Для досягнення цієї мети необхідно здійснити наступні кроки:

- провести аналіз предметної області та визначити основні вимоги до системи;
- розробити архітектуру системи та вибрати відповідні технології для її реалізації;
- реалізувати основні модулі системи, включаючи текстовий, аудіо та відеочат;
- провести тестування розробленого застосунку та верифікацію вимог.

Об’єктом цього дослідження є процеси комунікації в режимі реального часу у вебдодатках. Предметом дослідження є технології та методи розробки вебдодатків для комунікації в режимі реального часу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз наявних вебсистем

Було здійснено аналіз типових рішень для даного типу вебзастосунків та виокремлено три продукти для здійснення порівняльного аналізу: BigBlueButton, Discord та Zoom. Кожен з цих продуктів має свої особливості, переваги та недоліки, що дозволяє зробити висновки щодо їх придатності для різних сценаріїв використання та визначити переваги розроблюваного вебзастосунка.

BigBlueButton є платформою з відкритим кодом, що дозволяє користувачам налаштовувати та інтегрувати її відповідно до своїх потреб. Вона підтримує багатокористувацькі відеоконференції з можливістю запису, спільне використання екрана для презентацій або навчальних сесій, а також інтерактивну дошку для спільного малювання та анотацій. Текстовий чат для учасників конференції також є частиною функціонала. Переваги включають можливість повної персоналізації та інтеграції, орієнтацію на освітні потреби та безкоштовне використання без обмежень на базові функції. Проте BigBlueButton має складну установку та налаштування для користувачів без технічного досвіду, обмежену підтримку мобільних пристроїв та інтерфейс користувача, що може здаватися застарілим.

Discord підтримує текстовий, голосовий та відеочат, багатоканальні текстові чати, голосові канали та відеоконференції. Користувачі можуть створювати власні сервери з різними каналами для організації спілкування. Платформа орієнтована на геймерів з можливістю транслювати ігри в режимі реального часу та підтримкою ботів для автоматизації завдань та інтеграції з іншими сервісами. Переваги Discord включають простоту у використанні та налаштуванні, багатофункціональність та підтримку інтеграцій, а також потужні можливості для спільноти та командного спілкування. Водночас платформа має обмежену можливість для професійних відеоконференцій та бізнес-зустрічей, невеликий контроль над конфіденційністю у великих спільнотах, а деякі функції доступні лише в платній версії (Discord Nitro).

Zoom підтримує HD відео та аудіо для багатокористувацьких конференцій, можливість записувати зустрічі для подальшого перегляду, віртуальні фони під час відеозустрічей, спільне використання екрана та проведення вебінарів з великою кількістю учасників. Переваги Zoom включають високу якість відео та аудіо, інтуїтивний інтерфейс користувача та підтримку великої кількості учасників у конференціях та вебінарах. Недоліки включають обмеження в безкоштовній версії (обмеження за часом зустрічей), проблеми з конфіденційністю та безпекою, які викликали занепокоєння у минулому, а також високу вартість платних підписок для додаткових функцій.

Кожен з розглянутих продуктів – BigBlueButton, Discord та Zoom – має свої унікальні особливості, переваги та недоліки, що робить їх придатними для різних сценаріїв використання. BigBlueButton, орієнтований на освітні потреби, пропонує високий рівень персоналізації завдяки відкритому коду, але вимагає технічних знань для налаштування та підтримки. Discord є чудовим вибором для геймерів та спільнот завдяки своїй простоті у використанні та багатофункціональності, але обмежений у професійних налаштуваннях та конфіденційності. Zoom забезпечує високу якість відео та аудіо для конференцій і вебінарів, проте має обмеження у безкоштовній версії та виклики щодо безпеки та конфіденційності.

Основні переваги проєктованого вебзастосунка над вищеподаними рішеннями включають персоналізацію завдяки відкритому коду, інтеграцію функцій в одному застосунку, додаткові функції, гнучкість та масштабованість. Вебзастосунок, що створюється, прагне об’єднати найкращі риси цих рішень, додавши нові функції та можливості для забезпечення високої якості комунікації та задоволення потреб користувачів.

1.2 Аналіз вимог до системи

Аналіз вимог до системи є критично важливим етапом у процесі розробки програмного забезпечення, оскільки він визначає, які функціональні та нефункціональні вимоги повинна задовольняти система для успішного виконання своїх задач. На цьому етапі здійснюється збір, документування та перевірка вимог, що дозволяє забезпечити чітке розуміння очікувань користувачів та зацікавлених сторін. Визначення вимог допомагає уникнути двозначностей, зменшити ризики виникнення помилок у подальших етапах розробки та забезпечити відповідність кінцевого продукту початковим цілям проєкту. Таким чином, аналіз вимог до системи є фундаментальним кроком для створення ефективного, надійного та зручного у використанні програмного забезпечення.

Вебсистема повинна забезпечувати наступні функціональні можливості:

- реєстрація користувача;
- відправлення електронного листа для здійснення активації облікового запису;
- відправлення вітального електронного листа;
- авторизація користувача;
- вихід з облікового запису;
- надсилання електронного листа з посиланням на сторінку для відновлення пароля;
- можливість зміни пароля;
- встановлення статусу користувача (онлайн/офлайн);
- можливість копіювання унікального ідентифікатора користувача та каналу;
- надсилання, отримання, відкликання, відхилення та підтвердження запитів щодо додавання контакту, вступу та запрошення в канал;
- можливість приєднання до відкритого каналу;
- створення власного каналу;
- надсилання, відкликання, відхилення та прийняття аудіо та відео виклику;

- видалення контакту та вихід із каналу;
- можливість надсилання текстових (в тому числі емодзі) та файлових повідомлень;
- запис, попереднє прослуховування та надсилання аудіоповідомлень;
- можливість відкриття та перегляду текстових, pdf та відео файлів, а також зображень;
- завантаження файлів на власний пристрій;
- можливість додавання реакцій на повідомлення в чаті;
- приєднання до медіаканалу та вихід із медіаканалу;
- передавання аудіо/відео потоку до віддаленого користувача;
- ввімкнення/вимкнення мікрофона та камери;
- режим шумозаглушення;
- створення індикатора звуку для виявлення активних користувачів;
- віртуальний фон із можливістю вибору інтенсивності розмиття, встановлення кольору та завантаження зображень та відео файлів;
- можливість здійснення спільного доступу до екрана користувача;
- створення інтерактивної дошки із такими функціями: виділення частини дошки, малювання пензлем та визначеним набором фігур із встановленням кольору та товщини, введення тексту, вибір інструментів для часткового/повного видалення вмісту дошки, скасування та повторення дії, наближення/віддалення області дошки та додавання сторінок.

Інформаційна система повинна зберігати дані в документо-орієнтованій базі даних.

Перелік вимог до інтерфейсу наступний:

- вебсторінки повинні однаково відображатись незалежно від виду та версії браузера;
- інтерфейс повинен бути простим;
- сторінки повинні використовувати кодування UTF-8;
- інтерфейс повинен бути адаптивним (різне відображення вебсторінок залежно від роздільної здатності екрана пристрою).

Система повинна підтримувати мінімум 50 користувачів (кількість користувачів, що знаходяться в одному аудіо/відео каналі обмежується десятьма), що працюють одночасно, а час відгуку не повинен перевищувати 0,5 секунд для простих і 1,5 секунди для складніших задач.

Вимоги до безпеки для інформаційної системи наступні:

– доступ до системи має здійснюватися лише за допомогою веббраузерів для різних пристроїв;
– доступ до вебсайту повинен бути здійснений за допомогою протоколу HTTPS;

Перелік експлуатаційних вимог такий:

– браузери: Chrome, Safari, Edge, Mozilla Firefox;
– операційні системи: Windows 7/8/10/11, Linux, Mac OS.

1.3 Опис акторів та прецедентів системи

В результаті аналізу вимог було виділено наступних акторів:

– guest: даний актор представляє неавторизованого користувача і має можливість зареєструватися, ввійти в систему з використанням електронної пошти та пароля або облікового запису Google, а також може відновити пароль до облікового запису;
– contact: актор, який є у списку контактів іншого користувача. Він має доступ до того набору функціональних можливостей, які не включають взаємодію в каналі, тобто встановлення статусу (онлайн/офлайн), додавання контактів/каналів, здійснення аудіо/відео викликів до іншого контакту та ін.;
– channelMember: даний актор представляє учасника каналу, який має змогу надсилати повідомлення в цьому каналі, приєднуватися до аудіо/відео комунікацій і т.д.;

– channelAdmin: репрезентує користувача, який створив канал. За здійсненням функціональних можливостей відповідає актору channelMember;
– authorizedUser: узагальнений актор для акторів contact, channelMember та channelAdmin;
– channelUser: узагальнений актор для акторів channelMember та channelAdmin.

Для вищезгаданих акторів наведено відповідні варіанти використання в таблиці 1.1.

Таблиця 1.1 – Варіанти використання системи

Код	Актор	Найменування	Формування
1	2	3	4
G1	guest	Створити акаунт	Передбачає можливість створення облікового запису для неавторизованого користувача
G2	guest	Активувати обліковий запис	Передбачає отримання повідомлення на вказану електронну пошту із посиланням на сторінку для активації акаунта
G3	guest	Отримати вітального листа	Користувач отримує вітальний лист, що приходить на вказану при реєстрації електронну пошту, одразу після створення облікового запису
G4	guest	Відновити пароль	Передбачає можливість відновлення/зміни пароля на основі введеної у визначене поле електронної пошти
G5	guest	Ввійти в акаунт	Неавторизований користувач може здійснити вхід в обліковий запис за допомогою електронної пошти та пароля або з використанням Google акаунта

Продовження таблиці 1.1

1	2	3	4
U1	authori-zedUser	Вийти з облікового запису	Для зазначеного актора є можливість виходу з акаунта
U2	authori-zedUser	Встановити статус	Передбачає встановлення поточного статусу користувача (онлайн/офлайн)
U3	authori-zedUser	Скопіювати власний id	Авторизовані користувачі можуть копіювати власні унікальні ідентифікатори для того, щоб інші користувачі змогли відправити запит на встановлення контакту
U4	channel-User	Скопіювати id каналу	Для вступу в закриті канали необхідно надіслати запит із зазначенням id цього каналу
U5	contact	Обробити запит на додавання контакту	Актор contact може надсилати, отримувати, відкликати, відхиляти та підтверджувати запити щодо додавання контакту
U6	channel-User	Обробити запити вступу та запрошення в канал	Авторизовані користувачі можуть надсилати, отримувати, відкликати, відхиляти та підтверджувати запити щодо вступу та запрошення в канал
U7	authori-zedUser	Приєднатися до каналу	Користувачі можуть приєднатися до відкритого каналу без необхідності відправлення запиту
U8	authori-zedUser	Створити канал	Авторизовані користувачі можуть створювати власні канали, зазначивши назву та тип каналу (відкритий/закритий)

Продовження таблиці 1.1

1	2	3	4
U9	authori-zedUser	Обробити аудіо та відео виклики	Передбачає можливість надсилання, відкликання, відхилення та прийняття аудіо та відеовикликів
U10	contact	Видалити контакт	Передбачає видалення контакту зі списку контактів
U11	channel-User	Залишити канал	Передбачає можливість виходу із певного каналу
U12	authori-zedUser	Надіслати повідомлення	Авторизовані користувачі мають можливість надсилання текстових (в тому числі емодзі) та файлових повідомлень
U13	authori-zedUser	Обробити аудіоповідом-лення	Передбачає запис, попереднє прослухову-вання та надсилання аудіоповідомлень
U14	authori-zedUser	Відкрити файл	Учасники чату мають можливість відкриття та перегляду текстових, pdf та відео файлів, а також зображень
U15	authori-zedUser	Завантажити файл	Учасники чату можуть завантажити файли на власний пристрій
U16	authori-zedUser	Додати реакцію	Учасники чату мають можливість додавання реакцій на повідомлення в чаті
U17	authori-zedUser	Керувати під-ключенням до медіаканалу	Авторизовані користувачі можуть приєднатися та залишити медіаканал
U18	authori-zedUser	Передати аудіо та відео потоки	Передбачає передавання аудіо/відео потоку до віддаленого користувача

Продовження таблиці 1.1

1	2	3	4
U19	authori-zedUser	Керувати мікрофоном та камерою	Передбачає ввімкнення/вимкнення мікрофо-на та камери
U20	authori-zedUser	Увімкнути та вимкнути режим шумо-заглушення	Надає можливість використання режиму шумозаглушення
U21	authori-zedUser	Налаштувати віртуальний фон	Авторизовані користувачі можуть ввімкнути/вимкнути віртуальний фон із можливістю вибору інтенсивності розмиття, встановлення кольору та завантаження зображень та відео файлів
U22	authori-zedUser	Надати спіль-ний доступ до екрана	Передбачає можливість здійснення спільного доступу до екрана користувача
U23	authori-zedUser	Використати інтерактивну дошку з розширеними функціями	Передбачає використання інтерактивної дошки із такими функціями: виділення частини дошки, малювання пензлем та визначеним набором фігур із встановленням кольору та товщини, введення тексту, вибір інструментів для часткового/повного видалення вмісту дошки, скасування та повторення дії, наближення/віддалення області дошки та додавання сторінок

Діаграма варіантів використання, що показана на рис. 1.1, є важливим інструментом для візуалізації взаємодії між різними учасниками системи та основних сценаріїв їхньої взаємодії. Діаграма є результатом застосування

стандарту Unified Modeling Language (UML) [6], який забезпечує уніфікацію та стандартизацію процесу моделювання програмних систем. На діаграмі виділено дві основні сутності: актори та прецеденти. Актори представляють ролі або сутності, які взаємодіють з системою, тоді як варіанти використання представляють конкретні взаємодії або сценарії, які можуть відбуватися між акторами та самою системою.

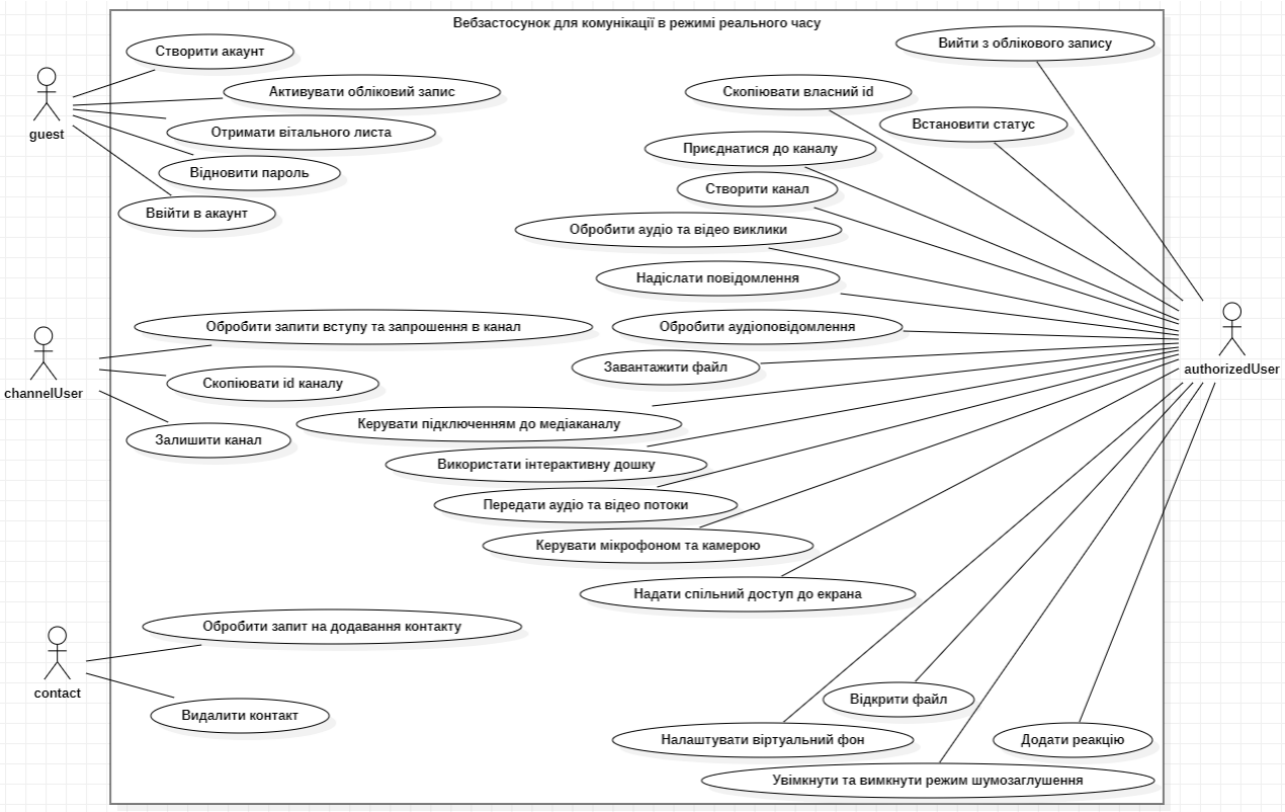

Рисунок 1.1 – Діаграма варіантів використання

1.4. Опис архітектури та технологій системи

Архітектура системи визначає її структуру, компоненти, їхні взаємозв’язки та принципи взаємодії між ними. Проєктована система для комунікації в режимі реального часу, що підтримує текстовий, аудіо та відеочат, побудована на

сучасному технологічному стеку MERN (MongoDB, Express.js, React, Node.js). Нижче наведено детальний опис архітектури та використовуваних технологій.

Система використовує клієнт-серверну архітектуру, яка передбачає чітке розділення обов'язків між клієнтською та серверною частинами. Така архітектура дозволяє досягти високої продуктивності, масштабованості та зручності в обслуговуванні.

На клієнтській стороні вебзастосунка використовуються такі технології:

- React: використовується для створення інтерактивних користувацьких інтерфейсів. React дозволяє будувати компонентну архітектуру, що сприяє підвищенню повторного використання коду та полегшує підтримку [7];
- Redux та Redux Toolkit: забезпечують управління станом додатка. Це дозволяє централізовано зберігати та обробляти дані, спрощуючи передачу даних між компонентами [8];
- TailwindCSS: використовується для швидкої та гнучкої стилізації інтерфейсу. Tailwind надає набір класів, які дозволяють легко застосовувати стилі без написання власного CSS [9];
- TypeScript: доповнює JavaScript статичним типізуванням, що допомагає уникнути багатьох помилок на етапі розробки та робить код більш зрозумілим і підтримуваним [10].

На серверній частині створюваного вебдодатка було використано:

- Node.js: використовується як середовище виконання для сервера. Node.js забезпечує високу продуктивність і масштабованість завдяки асинхронній моделі вводу-виводу [11];
- Express.js: фреймворк для Node.js, що надає широкий набір інструментів для створення вебсерверів та API. Express дозволяє легко налаштовувати маршрутизацію, обробку запитів та відповідей [12];
- Mongoose: бібліотека для роботи з MongoDB, яка надає зручний інтерфейс для взаємодії з базою даних, забезпечуючи валідацію, створення схем та інші функції [13].

Як базу даних було вибрано MongoDB – документо-орієнтовану базу даних, яка зберігає дані у форматі JSON-подібних документів [14]. MongoDB забезпечує високу гнучкість у зберіганні даних та масштабованість.

Для забезпечення високої якості зв’язку та надійності обрано сервіси реального часу Agora RTC, RTM, Chat, Interactive Whiteboard, що надають SDK для інтеграції аудіо та відеозв’язку, текстового чату та інтерактивної дошки [15].

Клієнтська частина надсилає запити до серверної частини через протокол HTTP, забезпечуючи швидкий обмін даними в режимі реального часу. Серверна частина обробляє запити, виконує необхідні операції з базою даних та надає необхідні дані для функціонування клієнта. Взаємодія з сервісами Agora для забезпечення аудіо та відеозв’язку відбувається безпосередньо з клієнта (браузера), що забезпечує високу якість та надійність зв’язку. Серверна частина забезпечує мінімальну взаємодію з сервісом Agora, обмежуючись лише допоміжними функціями. Всі дані, що надходять від користувачів, зберігаються у MongoDB, що забезпечує їхню доступність та цілісність.

Отже, здійснено аналіз предметної області, що включає дослідження сучасних технологій та методів розробки вебзастосунків для комунікації в режимі реального часу. Розглянуто вимоги до системи, як функціональні, так і нефункціональні, що забезпечують високу якість та надійність майбутнього застосунку. Окрім того, проведено аналіз рішень, що існують на ринку, таких як BigBlueButton, Discord та Zoom, що дає змогу визначити переваги та недоліки кожного з них. Це дозволяє краще зрозуміти, які функціональні можливості та особливості повинні бути реалізовані у проєктованій системі, щоб вона була конкурентоспроможною та задовольняла потреби користувачів. Отримані результати аналізу стають основою для подальшого проєктування та розробки вебзастосунку, забезпечуючи чітке розуміння вимог та очікувань від майбутньої системи.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

2.1. Проєктування бази даних та сховища медіафайлів

Проєктування бази даних є важливим етапом у розробці системи, оскільки від нього залежить ефективне зберігання та обробка даних. У даній системі для зберігання даних використовується документо-орієнтована база даних MongoDB, що дозволяє гнучко зберігати структури даних у форматі JSON-подібних документів. Для збереження медіафайлів, таких як зображення та відео, які використовуються лише для віртуального фону, застосовується IndexedDB на стороні клієнта.

Для проєктування структури бази даних у MongoDB використовуються моделі документів, які визначають структуру та взаємозв’язки між даними. Основними моделями даних у системі є User, Channel, Contact, Membership, Request та Token.

Структуру вищезгаданих моделей даних подано далі.

User:

- username: унікальне ім’я користувача;
- email: електронна адреса користувача;
- password: хешований пароль користувача;
- active: статус активності облікового запису;
- isOnline: статус користувача.

Channel:

- name: назва каналу;
- admin: ID адміністратора каналу;
- isPrivate: статус приватності каналу;
- chatTargetId: ID чату;
- whiteboardRoomId: ID кімнати інтерактивної дошки.

Contact:

- user1: ID першого користувача;

– user2: ID другого користувача;
– whiteboardRoomId: ID кімнати інтерактивної дошки.

Membership:

– user: ID користувача;
– channel: ID каналу.

Request:

– from: ID користувача, що надсилає запит;
– to: ID користувача, що отримує запит;
– type: тип запиту (встановлення контакту, запрошення до каналу, приєднання);
– channel: ID каналу (опціонально, залежно від типу запиту).

Token:

– userId: ID користувача;
– token: токен для аутентифікації;
– createdAt: дата створення токена.

Зв’язки між цими моделями визначають, як частини даних взаємодіють та взаємозалежні одне від одного. У даній системі користувачі (User) можуть бути пов’язані з багатьма каналами (Channel), членствами (Membership), контактами (Contact), запитами (Request) та токенами (Token). Кожен канал може мати багато учасників та запитів. Це забезпечує гнучку та масштабовану архітектуру для роботи з даними у системі комунікації в режимі реального часу. Графічне представлення зв’язків між моделями даних можна побачити на рис. 2.1.

Для збереження медіафайлів (зображень та відео для віртуального фону) використовується IndexedDB на стороні клієнта. IndexedDB дозволяє зберігати великі обсяги структурованих даних, включаючи файли, безпосередньо в браузері користувача. Це зменшує навантаження на сервер та забезпечує швидкий доступ до медіафайлів під час роботи з додатком.

Проєктування бази даних та сховища медіафайлів з використанням MongoDB та IndexedDB дозволяє забезпечити гнучкість, масштабованість та ефективність зберігання даних. Використання цих технологій відповідає сучасним вимогам до

систем комунікації в режимі реального часу, забезпечуючи надійність та високу продуктивність додатка.

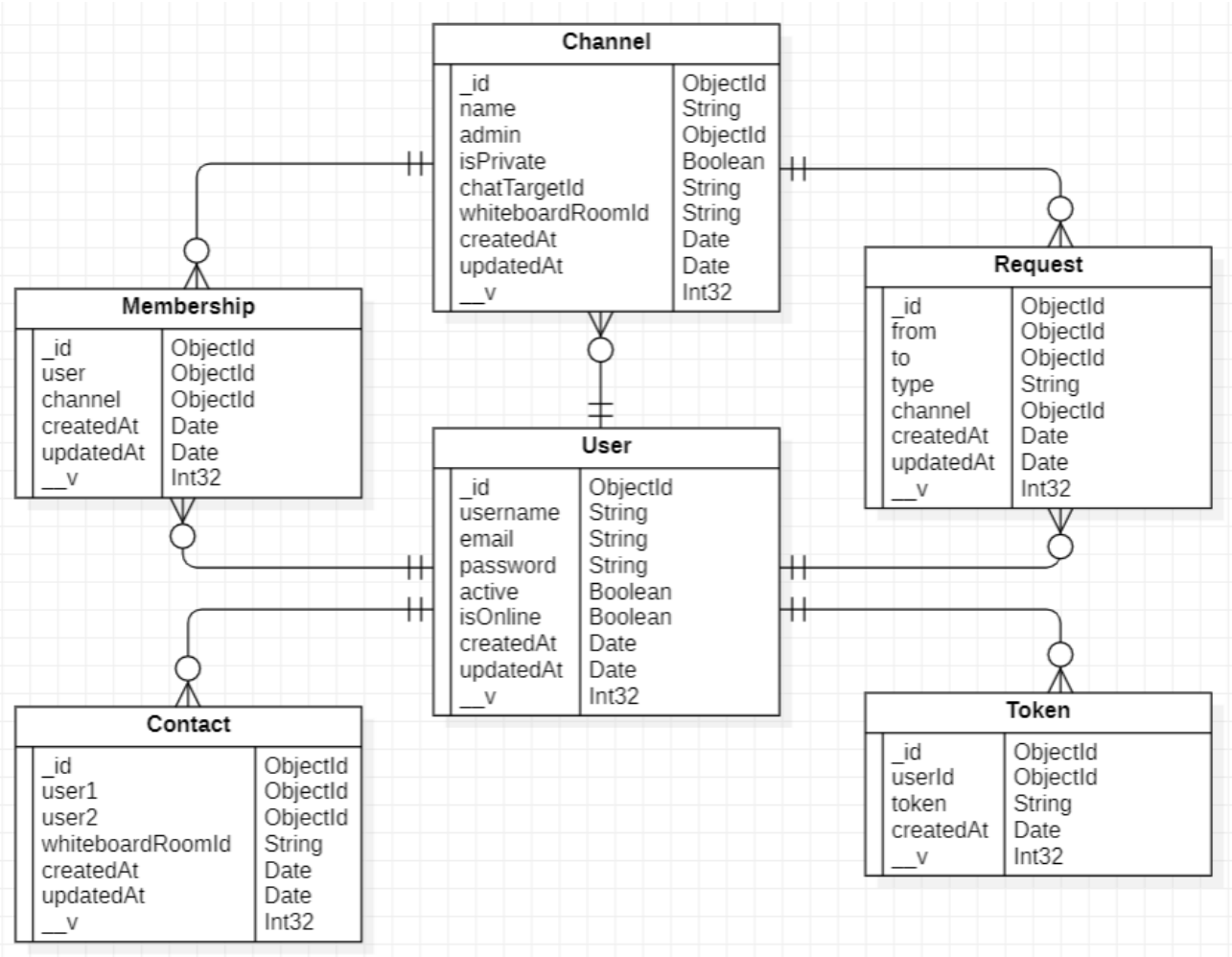

Рисунок 2.1 – Діаграма зв’язків між моделями даних

2.2. Розробка модулів системи

Розробка модулів системи передбачає поділ серверної логіки на окремі компоненти, такі як маршрути (routes), контролери (controllers) та сервіси (services). Такий підхід дозволяє зробити код більш організованим, підтримуваним та розширюваним.

Серверна частина системи реалізована на основі платформи Node.js з використанням фреймворку Express.js для обробки HTTP запитів та бібліотеки Mongoose для взаємодії з базою даних MongoDB. Основні компоненти серверної логіки включають:

- маршрути (routes): визначають, які контролери будуть викликані у відповідь на певні HTTP запити. Вони відповідають за приймання запитів від клієнтів та передачу їх контролерам для подальшої обробки;
- контролери (controllers): обробляють вхідні запити, взаємодіють із сервісами для виконання бізнес-логіки та повертають відповідні відповіді клієнтам. Вони виступають посередниками між маршрутами та сервісами, забезпечуючи чіткий поділ обов'язків;
- сервіси (services): містять бізнес-логіку та взаємодію з базою даних через моделі Mongoose. Вони інкапсулюють складні операції, що дозволяє зменшити дублювання коду та покращити його повторне використання.

У нашій системі визначено наступні маршрути, контролери та сервіси:

- auth: відповідає за аутентифікацію користувачів;
- channel: відповідає за управління каналами;
- chat: відповідає за реєстрацію користувачів у системі чату та генерацію токенів для доступу до чату;
- contact: відповідає за управління контактами користувачів;
- request: відповідає за обробку запитів між користувачами;
- rtc: відповідає за генерацію токенів для налаштування сервісів для спілкування в реальному часі (Real-Time Communication);
- rtm: відповідає за генерацію токенів для управління повідомленнями в режимі реального часу (Real-Time Messaging);
- user: відповідає за управління інформацією про користувачів;
- whiteboard: відповідає за управління інтерактивною дошкою.

Опис ключових функцій для кожного із вищезгаданих сервісів подано далі.

Сервіс auth:

- registerUser: використовується для реєстрації нового користувача;

– verifyAccount: використовується для верифікації акаунта користувача за допомогою токена;
– loginUser: використовується для входу користувача в систему за допомогою електронної пошти та пароля;
– loginUserWithGoogle: використовується для входу користувача в систему за допомогою Google OAuth2;
– requestPasswordReset: використовується для запиту скидання пароля користувача;
– resetPassword: використовується для скидання пароля користувача за допомогою токена скидання пароля.

Сервіс channel:

– createChannel: використовується для створення нового каналу з вказаними адміністратором та учасниками;
– searchChannels: використовується для пошуку каналів за ідентифікатором або назвою;
– joinChannel: використовується для приєднання користувача до каналу за його ідентифікатором;
– leaveChannel: використовується для виходу користувача з каналу за його ідентифікатором;
– getChannelById: використовується для отримання інформації про канал за його ідентифікатором.

Сервіс chat:

– registerChatUser: використовується для реєстрації нового користувача в сервісі Agora Chat;
– generateAppChatToken: використовується для генерації токена додатка для сервісу Agora Chat;
– generateChatToken: використовується для генерації токена користувача для сервісу Agora Chat.

Сервіс contact:

- getContactRelation: використовується для отримання інформації про контактні відносини між двома користувачами;
- getContactById: використовується для отримання інформації про контакт за його ідентифікатором;
- removeContactRelation: використовується для видалення контактних відносин між двома користувачами.

Сервіс request:

- createRequest: використовується для створення нового запиту від одного користувача до іншого, включаючи запити на приєднання до каналу, запрошення до каналу та додавання контакту;
- getUserRequests: використовується для отримання всіх запитів, пов’язаних з певним користувачем;
- declineRequest: використовується для відхилення запиту від одного користувача до іншого;
- acceptRequest: використовується для прийняття запиту від одного користувача до іншого, з можливістю створення контакту або членства у каналі.

Сервіс rtc**:**

- generateRTCToken: використовується для генерації токенів реального часу (RTC) для користувачів з певними ролями (публікатор або глядач) і типами токенів (userAccount або uid).

Сервіс rtm:

- generateRTMToken: використовується для генерації токенів обміну повідомленнями в режимі реального часу (RTM) для користувачів з можливістю задавати час дії токена.

Сервіс user:

- getUserContacts: використовується для отримання контактів користувача за його ідентифікатором;

- getUserChannels: використовується для отримання всіх каналів, у яких користувач є членом, за його ідентифікатором;
- getUserById: використовується для отримання інформації про користувача за його ідентифікатором, без включення пароля;
- updateUserById: використовується для оновлення інформації про користувача за його ідентифікатором.

Сервіс whiteboard:

- createRoom: використовується для створення нової кімнати за допомогою токена SDK;
- disableRoom: використовується для відключення кімнати, що існує, за допомогою UUID кімнати та токена SDK;
- listRooms: використовується для отримання списку всіх кімнат за допомогою токена SDK.

Основні компоненти клієнтської частини додатка включають маршрутизацію, форми автентифікації, сторінку профілю та інші ключові сторінки додатка. Вебзастосунок має наступні сторінки:

- WelcomePage: використовується для привітання нових користувачів та надання можливості входу або реєстрації в системі;
- LoginPage: використовується для входу користувачів у систему, включаючи можливість входу за допомогою Google;
- SignupPage: використовується для реєстрації нових користувачів у системі;
- AccountVerificationPage: використовується для верифікації акаунта користувача за допомогою токена, отриманого з URL, та відображення результату верифікації;
- RequestPasswordResetPage: використовується для запиту скидання пароля користувача та надсилання електронного листа з посиланням для скидання пароля;
- PasswordResetPage: використовується для скидання пароля користувача за допомогою токена, отриманого з URL, та введення нового пароля;

- ProfilePage: використовується для відображення інформації про профіль користувача, зміни онлайн-статусу, копіювання ідентифікатора користувача, а також надання доступу до функцій додавання контактів, приєднання/створення каналів, перегляду запитів та виходу з системи;
- ContactAdditionPage: використовується для додавання нових контактів за допомогою створення запиту на додавання контакту;
- ChannelAdditionPage: використовується для приєднання до каналів, що існують, або створення нових каналів;
- RequestsPage: використовується для перегляду, прийняття, відхилення та відкликання запитів між користувачами в системі, а також для керування цими запитами у вхідних та вихідних запитах;
- ContactChatPage: використовується для відображення інтерфейсу чату з конкретним контактом, перевірки статусу, а також для ініціювання медіавиклику з контактом;
- ChannelChatPage: використовується для відображення інтерфейсу чату групового каналу, включаючи можливість здійснення медіавиклику в межах каналу;
- ChannelPage: використовується для управління відео та аудіо комунікаціями в реальному часі в межах групового або одноосібного чату, включаючи функції віртуального фону та шумозаглушення.

2.3 Опис роботи системи та зв’язків між компонентами

Для кращого розуміння функціональності та взаємодії різних компонентів системи, розглянемо основні варіанти використання. Ці прецеденти демонструють, як користувачі можуть взаємодіяти з системою та як вона забезпечує виконання основних завдань. Нижче наведено п’ять ключових функціональних можливостей, які ілюструють роботу системи:

– додавання нового контакту;
– приєднання до закритого каналу;
– передавання аудіо/відео потоків до віддаленого користувача;
– надсилання повідомлення в чаті;
– запит на скидання пароля.

Ці функціональні особливості даної вебсистеми допомагають показати зв’язки між різними компонентами системи та їхню інтеграцію для забезпечення основних функцій додатка. Графічне представлення варіантів використання подано нижче за допомогою UML-діаграм.

Діаграму діяльності для додавання нового контакту можна побачити на рис. 2.2.

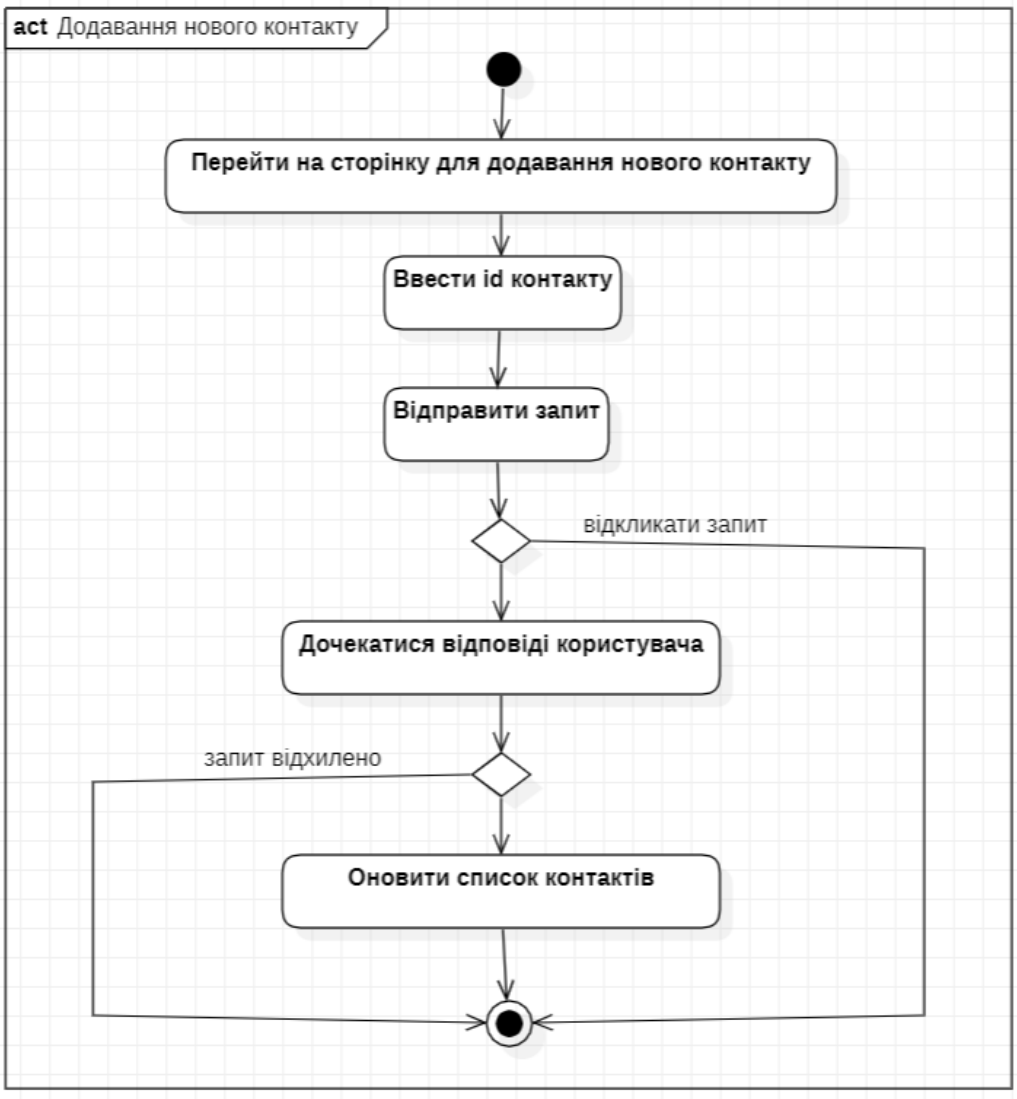

Рисунок 2.2 – Діаграма діяльності для додавання нового контакту

Додавання нового контакту розпочинається із моменту, коли користувач переходить на сторінку для додавання нового контакту, де він вводить ID контакту, який бажає додати. Після цього він відправляє запит на додавання контакту. Якщо запит відкликано до отримання відповіді, варіант використання завершується. В іншому випадку, система чекає на відповідь користувача, до якого було надіслано запит. Якщо запит відхилено, варіант використання також завершується. Якщо запит прийнято, система оновлює список контактів, додаючи нового користувача.

Для приєднання до закритого каналу користувач відправляє запит на приєднання. Після цього канал приймає запит і додає користувача в систему Agora Chat. Після успішного додавання користувача в Agora Chat, канал отримує підтвердження про те, що користувача додано. Далі канал оновлює список учасників у базі даних і отримує підтвердження про успішне оновлення даних. Після цього канал повідомляє користувача про успішне приєднання. Користувач відправляє запит на отримання оновленого списку каналів і отримує оновлені дані від бази даних. Діаграму послідовностей для цього варіанті використання можна побачити на рис. 2.3.

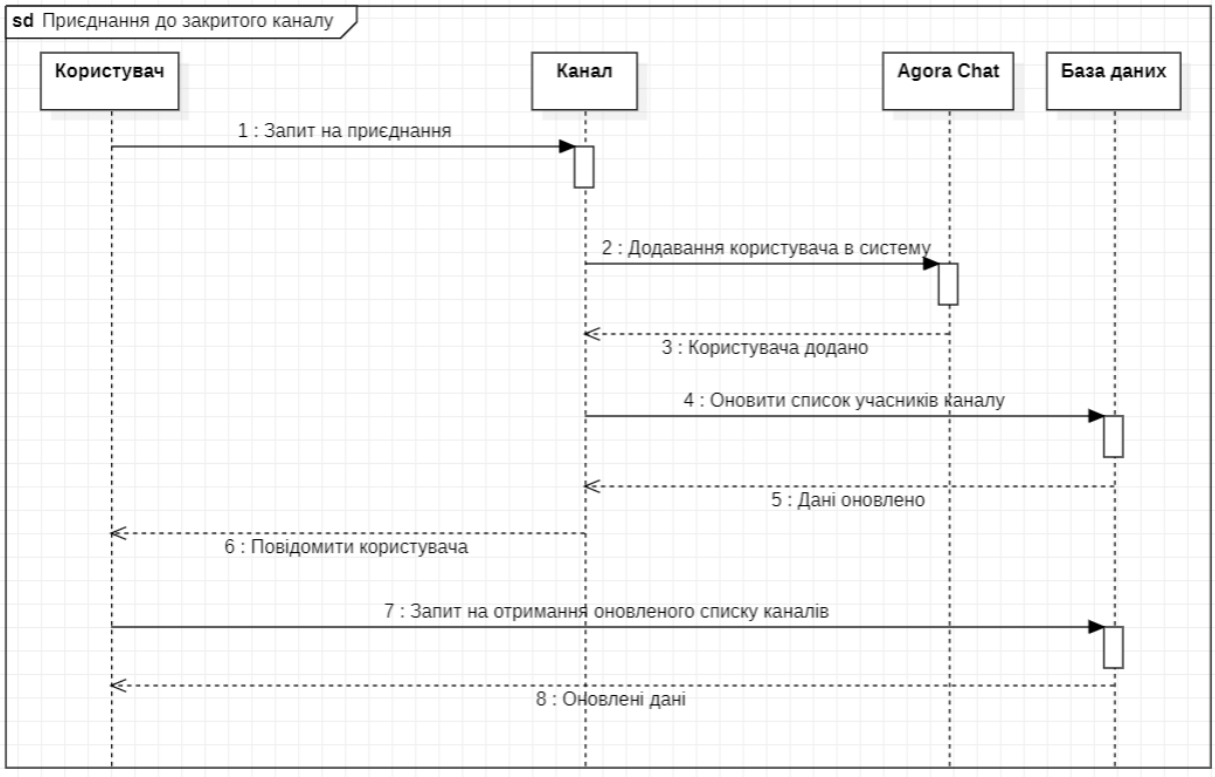

Рисунок 2.3 – Діаграма послідовностей для приєднання до закритого каналу

Для здійснення передачі аудіо та відеопотоків до іншого користувача користувач 1 генерує токен для приєднання до RTC кімнати, надсилаючи запит до сервера вебзастосунка. Сервер вебзастосунка відповідає, повертаючи згенерований токен. Користувач 1 використовує цей токен для приєднання до RTC кімнати через Agora RTC. Після цього користувач 1 отримує запит від Agora RTC на отримання дозволу на використання мікрофона та камери. Користувач 1 надає відповідні дозволи, після чого починає передачу аудіо та відео потоків до користувача 2. Відповідну діаграму послідовностей для цього прецеденту подано на рис. 2.4.

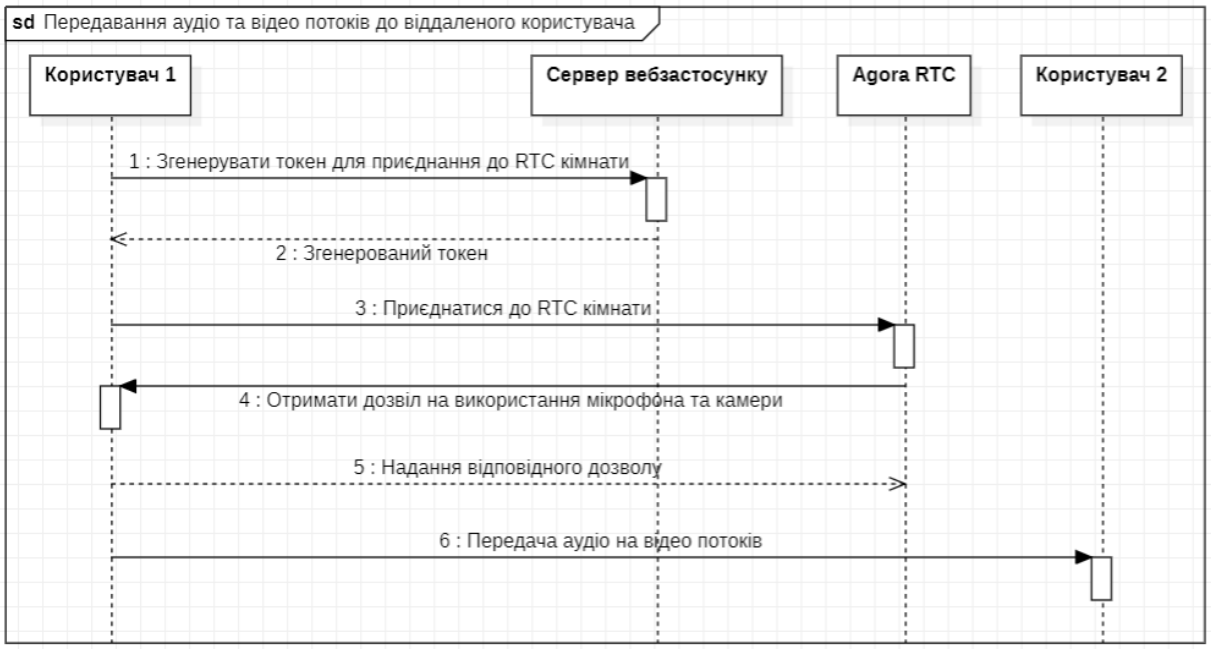

Рисунок 2.4 – Діаграма послідовностей для передавання аудіо та відеопотоків до віддаленого користувача

Для відправлення повідомлення в чаті користувач 1 вводить повідомлення в інтерфейсі чату та натискає кнопку «Відправити». Інтерфейс чату відправляє повідомлення на сервер Agora Chat. Після цього Agora Chat доставляє повідомлення іншим користувачам в чаті, зокрема користувачу 2. Користувач 2 отримує повідомлення у своєму інтерфейсі чату. Діаграму послідовностей для цього варіанту використання подано на рис. 2.5.

Діаграму діяльності для створення запиту на скидання пароля можна побачити на рис. 2.6.

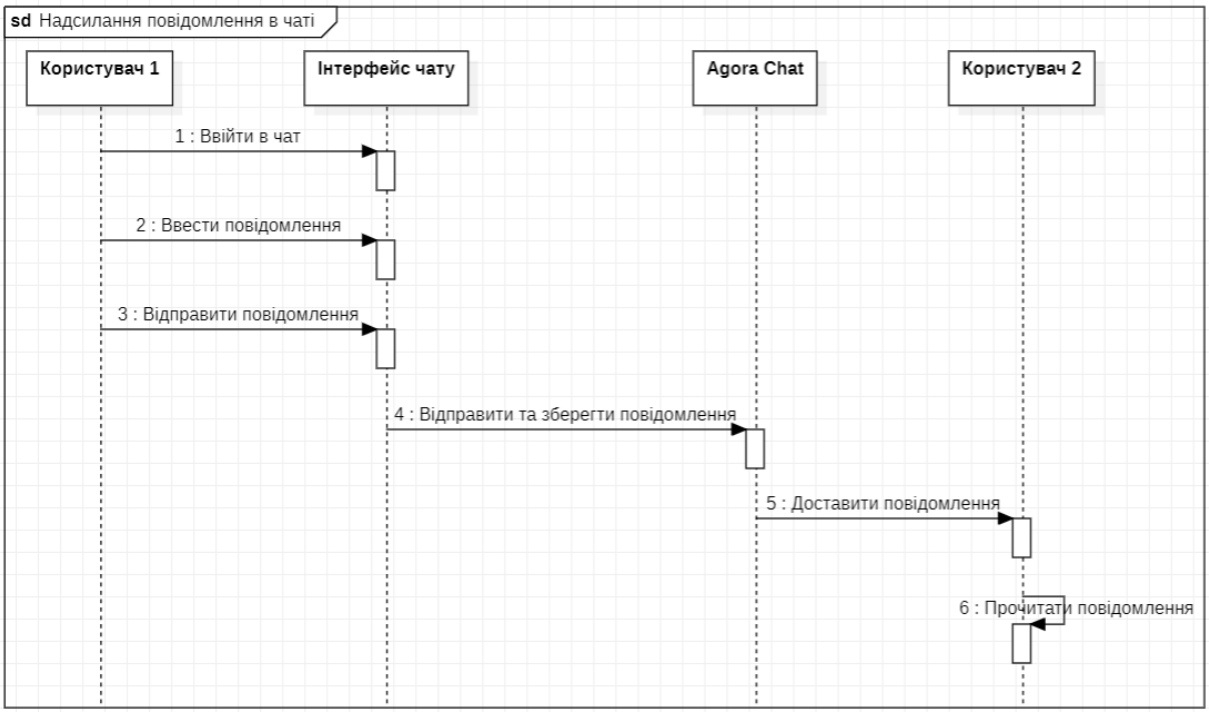

Рисунок 2.5 – Діаграма послідовностей для здійснення надсилання повідомлення в чаті

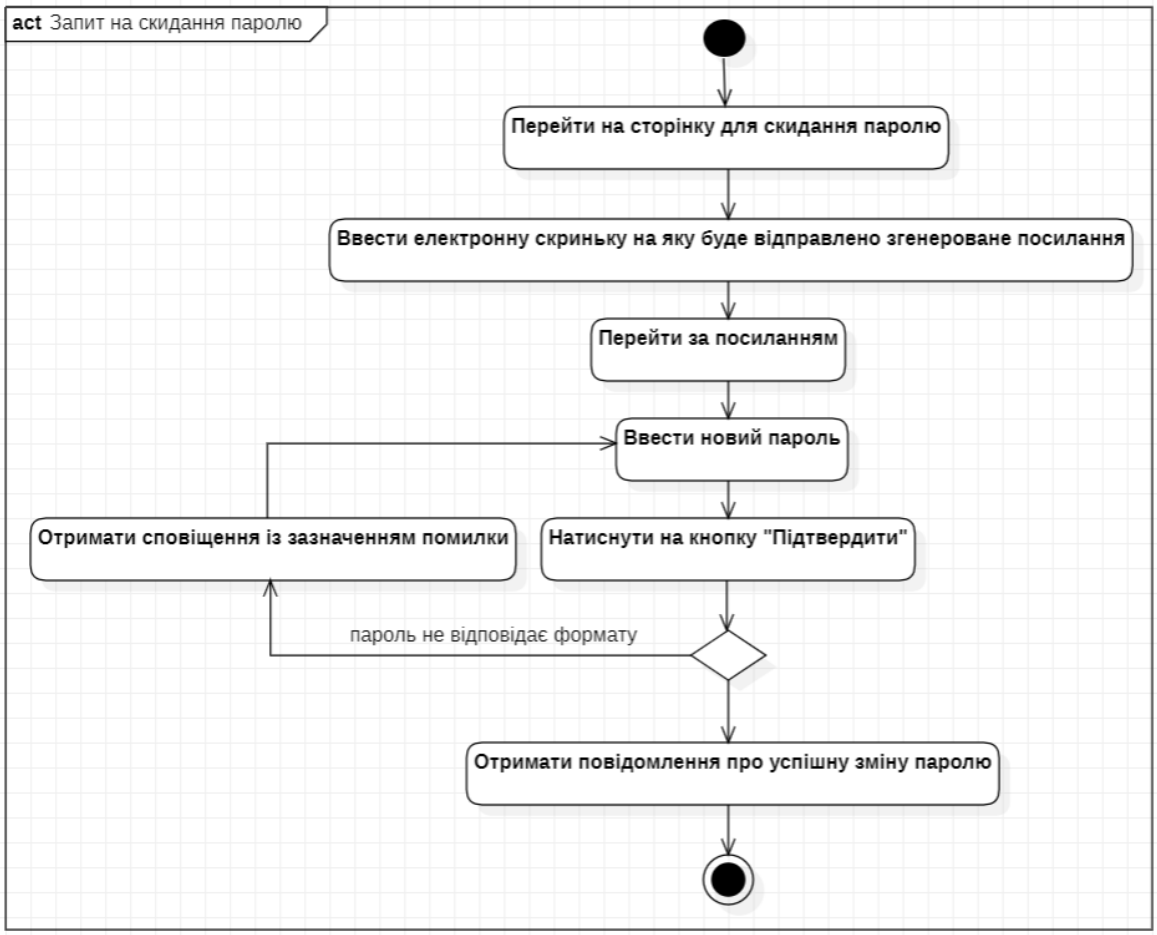

Рисунок 2.6 – Діаграма діяльності для створення запиту на скидання пароля

Для того, щоб створити запит на скидання пароля користувач переходить на відповідну сторінку. Далі він вводить свою електронну скриньку, на яку буде відправлено згенероване посилання для скидання пароля. Після отримання листа, користувач переходить за посиланням, вводить новий пароль та натискає на кнопку «Підтвердити». Якщо пароль не відповідає вимогам формату, користувач отримує сповіщення із зазначенням помилки та повертається до кроку введення нового пароля. У разі успішної зміни пароля користувач отримує відповідне повідомлення.

2.4 Реалізація функціональності

У цьому підрозділі наведено фрагменти коду, які ілюструють реалізацію ключових функцій системи на основі основних варіантів використання з попереднього підрозділу. Ці фрагменти коду демонструють, як конкретні дії користувача перетворюються на функціональні можливості додатка. Ці приклади допоможуть зрозуміти, як розроблено та інтегровано різні компоненти системи для забезпечення необхідної функціональності.

Сторінку для відправлення запиту на додавання нового контакту показано на рис. 2.7.

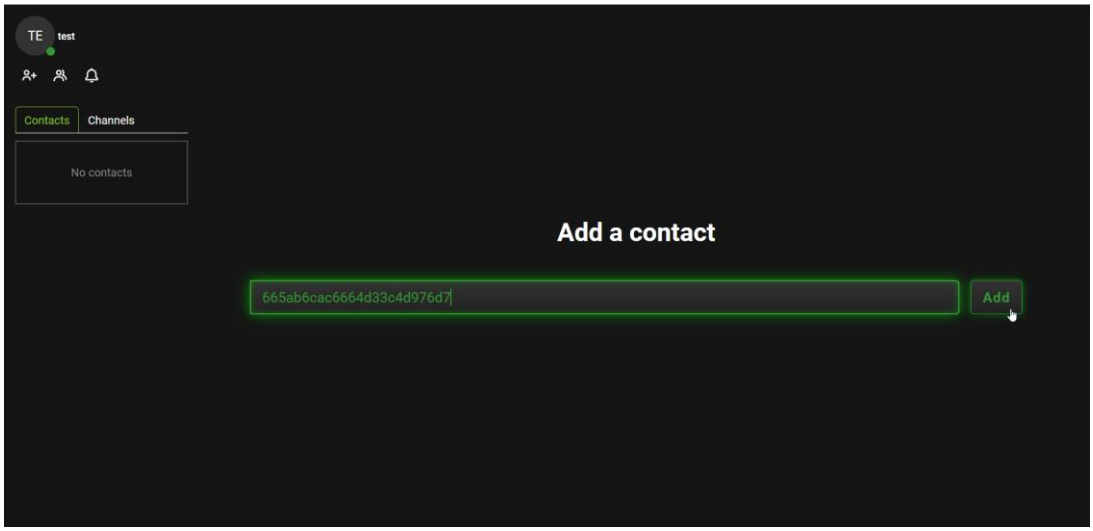

Рисунок 2.7 – Вигляд сторінки для додавання нового контакту

Розглянемо приклад реалізації додавання нового контакту. Фрагмент коду для відправлення запиту на додавання нового контакту за допомогою React та RTK Query можна побачити на рис. 2.8.

```
const handleSubmit: SubmitHandler<{ contactId: string }> = async
(data) => {
  try {
    await createRequest({
      from: userId ?? "",
      to: data.contactId,
      type: RequestTypeEnum.Contact,
    }).unwrap();
    await RTMClient.sendMessageToPeer(
      { text: PeerMessage.RequestCreated },
      data.contactId,
    );
    alert.success("Request created successfully!");
  } catch (err) {
    throw new Error(
      getErrorMessage({
        error: err,
        defaultErrorMessage:
          "Could not send a contact request. Please try again",
      }),
    );
  }
};
```

Рисунок 2.8 – Запит на додавання нового контакту

Функція handleSubmit відправляє запит на додавання нового контакту. Вона використовує createRequest для відправлення запиту до сервера та RTMClient для надсилання повідомлення до одержувача через Agora RTM. Якщо запит успішно відправлено, користувач отримує повідомлення про успішне створення запиту.

Фрагмент серверного програмного коду для обробки запитів, який створений за допомогою Node.js, Express.js та бібліотеки Mongoose, подано на рис. 2.9.

Функція createRequest (див. рис. 2.9) перевіряє, чи існує користувач, який відправляє запит, та чи є тип запиту дійсним. Залежно від типу запиту, вона викликає відповідну функцію для обробки запиту, наприклад handleContactRequest для запитів на додавання контакту.

```
const createRequest = async (
  from: string,
  to: string | null,
  type: string,
  channelId?: string,
) => {
  if (!mongoose.Types.ObjectId.isValid(from))
    throw new Error("Invalid user ID for 'from'");

  // Check if the 'from' user exists
  const fromUser = await User.findById(from);
  if (!fromUser) throw new Error("User 'from' not found");

  // Validate the request type
  if (!Object.values(RequestTypeEnum).includes(type as
RequestTypeEnum))
    throw new Error("Invalid request type");

  if (type === RequestTypeEnum.Join) return
handleJoinRequest(from, channelId);
  else if (type === RequestTypeEnum.Invite)
    return handleInviteRequest(from, to, channelId);
  else if (type === RequestTypeEnum.Contact)
    return handleContactRequest(from, to);
  else throw new Error("Invalid request type");
};
```

Рисунок 2.9 – Функція для створення запитів

Вигляд сторінки з вхідними/вихідними запитами подано на рис. 2.10. Якщо отримувач підтвердить запит на встановлення контакту, тоді буде створено новий документ в базі даних MongoDB. Фрагмент коду для підтвердження запитів різних типів можна побачити на рис. 2.11.

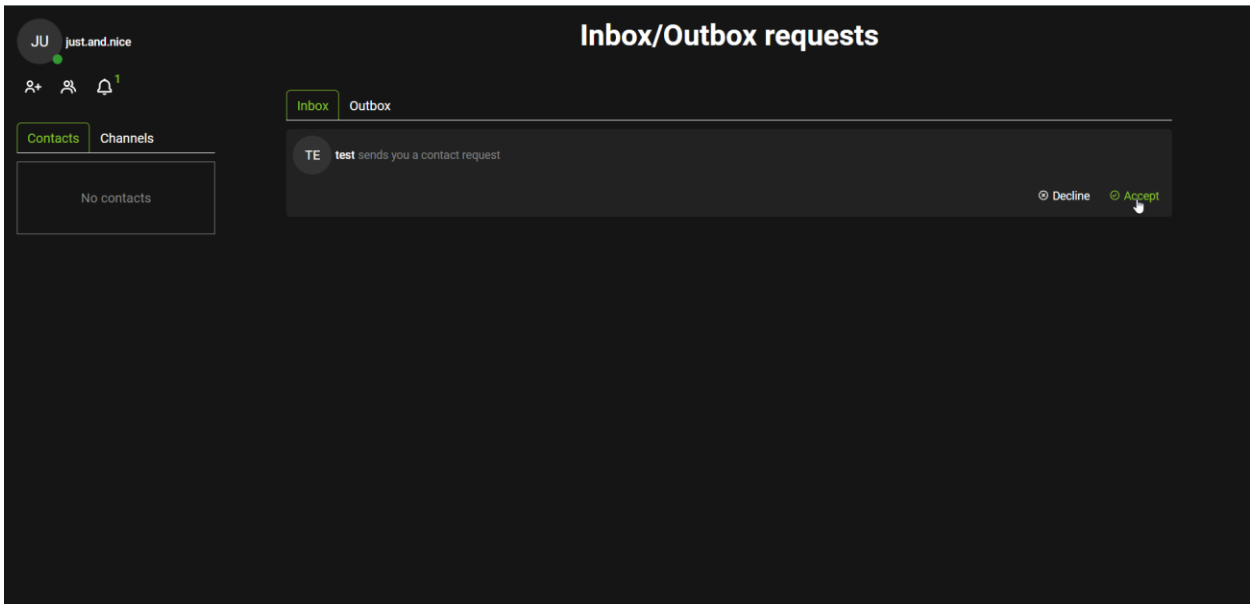

Рисунок 2.10 – Сторінка з вхідними/вихідними запитами

```
const acceptRequest = async (
  requestId: string,
  userId: string,
  whiteboardRoomId?: string,
) => {
  const request = await Request.findById(requestId);

  if (!request) throw new Error("Request not found");
  if (request.from.toString() !== userId &&
request.to.toString() !== userId)
    throw new Error("Unauthorized to accept this request");

  if (request.type === RequestTypeEnum.Contact &&
!whiteboardRoomId)
    throw new Error("Whiteboard id is required for creating a
contact");

  // Start a session for atomic updates
  const session = await mongoose.startSession();
  session.startTransaction();

  try {
    switch (request.type) {
      case RequestTypeEnum.Contact:
        const [user1, user2] =
          request.from < request.to
            ? [request.from, request.to]
            : [request.to, request.from];
        await Contact.findOneAndUpdate(
          { user1, user2, whiteboardRoomId },
          {},
          { upsert: true, new: true, setDefaultsOnInsert: true
},
        );
        break;
      case RequestTypeEnum.Invite:
        await Membership.findOneAndUpdate(
          { user: request.to, channel: request.channel },
          {},
          { upsert: true, new: true, setDefaultsOnInsert: true
},
        );
        break;
      case RequestTypeEnum.Join:
        await Membership.findOneAndUpdate(
          { user: request.from, channel: request.channel },
          {},
{ upsert: true, new: true, setDefaultsOnInsert: true },
        );
        break;
    }

    // Remove the request from the database
    await request.deleteOne();

    // Commit the transaction
    await session.commitTransaction();
    await session.endSession();

    return { success: true };
  } catch (err) {
    // Rollback the transaction in case of error
    await session.abortTransaction();
    await session.endSession();
    throw new Error("Could not accept the request. Please try
again.");
  }
};
```

Рисунок 2.11 – Функція для підтвердження запитів

Функція acceptRequest обробляє запити різних типів. Вона знаходить запит за його ID та перевіряє, чи запит дійсний, чи користувач авторизований для виконання цієї дії. Якщо запит є типу Contact, Invite або Join, відповідно оновлюється база даних, створюючи новий контакт або учасника каналу. Всі операції виконуються в межах транзакції для забезпечення цілісності даних. У разі успішної обробки запиту, запит видаляється з бази даних і транзакція підтверджується. У разі помилки транзакція скасовується.

Для створення запиту на приєднання до закритого каналу користувачеві необхідно перейти на відповідну сторінку (рис. 2.12). Програмний код клієнтської частини вебсистеми для приєднання до закритого каналу подано на рис. 2.13. З лістингу (див. рис. 2.13) можна побачити, що спочатку викликається fetchChannel для отримання інформації про канал за його ідентифікатором. Потім, за допомогою createRequest, створюється запит на приєднання до каналу, а через chatConnection.joinGroup користувач приєднується до групового чату каналу. Після цього користувачу, який є адміністратором каналу, відправляється повідомлення за допомогою RTMClient.sendMessageToPeer, інформуючи його про створений запит. Якщо запит буде підтверджено, то відправник отримає відповідне повідомлення, а інформацію в БД буде оновлено.

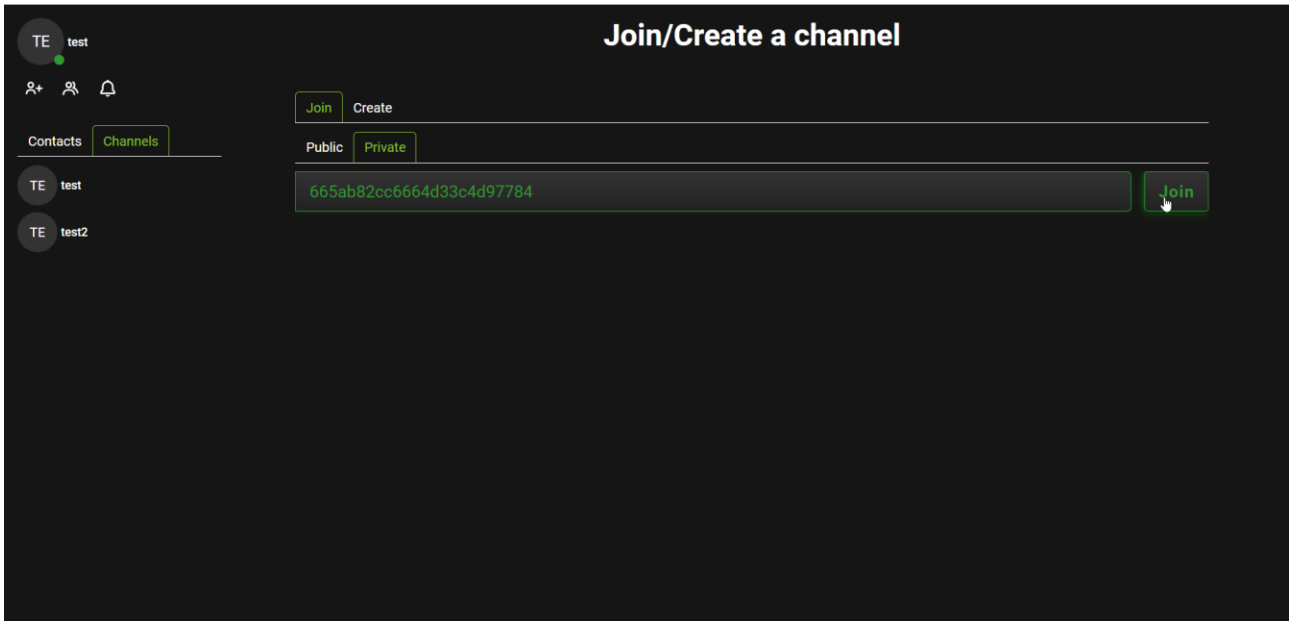

Рисунок 2.12 – Сторінка для створення запиту на вступ в закритий канал

```
const handlePrivateJoiningFormSubmit: SubmitHandler<{
  privateSearchValue: string;
}> = async (data) => {
  try {
    const channel = await
fetchChannel(data.privateSearchValue).unwrap();

    await Promise.all([
      createRequest({
        from: userId ?? "",
        to: null,
        type: RequestTypeEnum.Join,
        channel: channel._id,
      }).unwrap(),
      chatConnection.joinGroup({
        groupId: channel.chatTargetId,
        message: "",
      }),
    ]);
    await RTMClient.sendMessageToPeer(
      { text: PeerMessage.RequestCreated },
      channel.admin.toString(),
    );
    alert.success("Request created successfully!");
  } catch (err) {
    throw new Error(
      getErrorMessage({
        error: err,
        defaultErrorMessage:
          "Could not send a joining request. Please try again",
      }),
    );
  }
};
```

Рисунок 2.13 – Приєднання до закритого каналу

Для здійснення передавання аудіо та відео потоків необхідно згенерувати спеціальний токен для приєднання до RTC каналу. Фрагмент коду для генерування такого токена подано на рис. 2.14. Функція generateRTCToken (див. рис. 2.14) створює RTC токен для користувача. Спочатку вона визначає роль користувача як видавець або глядач. Потім встановлюється час дії токена, за замовчуванням 3600 секунд (1 година), якщо не вказано інше. Потім обчислюється час закінчення дії привілеїв токена. Токен може бути згенерований на основі облікового запису користувача або UID. Токен повертається для подальшого використання при підключенні до RTC кімнати. Після цього користувач під’єднується до RTC кімнати, надає дозвіл на використання мікрофона та камери, після чого аудіо та відео потоки передаються до віддалених користувачів.

```
const generateRTCToken = (
  channel: string,
  role: string,
  tokentype: string,
  uid: string,
  expiry?: string,
) => {
  // get role
  let userRole;

  if (role === "publisher") userRole = RtcRole.PUBLISHER;
  else if (role === "audience") userRole = RtcRole.SUBSCRIBER;
  else throw new Error("Role is incorrect");

  // get the expire time
  let expireTime;

  if (!expiry || expiry === "") expireTime = 3600;
  else {
    try {
      expireTime = parseInt(expiry, 10);
    } catch (err) {
      throw new Error(`Expiry is incorrect
${JSON.stringify(err)}`);
    }
  }

  // calculate privilege expire time
  const currentTime = Math.floor(Date.now() / 1000);
  const privilegeExpireTime = currentTime + expireTime;

  // build the token
  let token;
  if (tokentype === "userAccount")
    token = RtcTokenBuilder.buildTokenWithAccount(
      APP_ID,
      APP_CERTIFICATE,
      channel,
      uid,
      userRole,
      privilegeExpireTime,
    );
  else if (tokentype === "uid")
    token = RtcTokenBuilder.buildTokenWithUid(
      APP_ID,
      APP_CERTIFICATE,
      channel,
      parseInt(uid, 10),
      userRole,
      privilegeExpireTime,
    );
  else throw new Error("Token type is invalid");

  return token;
};
```

Рисунок 2.14 – Генерування RTC токена

Після отримання відповідного токена користувач може приєднатися до медіаканалу (рис. 2.15). Після приєднання до медіаканалу користувача має можливість користуватися різноманітними функціями, такими як віртуальний фон та чат (див. рис. 2.15), шумозаглушення, спільне використання екрана тощо. Вигляд сторінки після активування інтерактивної дошки можна побачити на рис. 2.16.

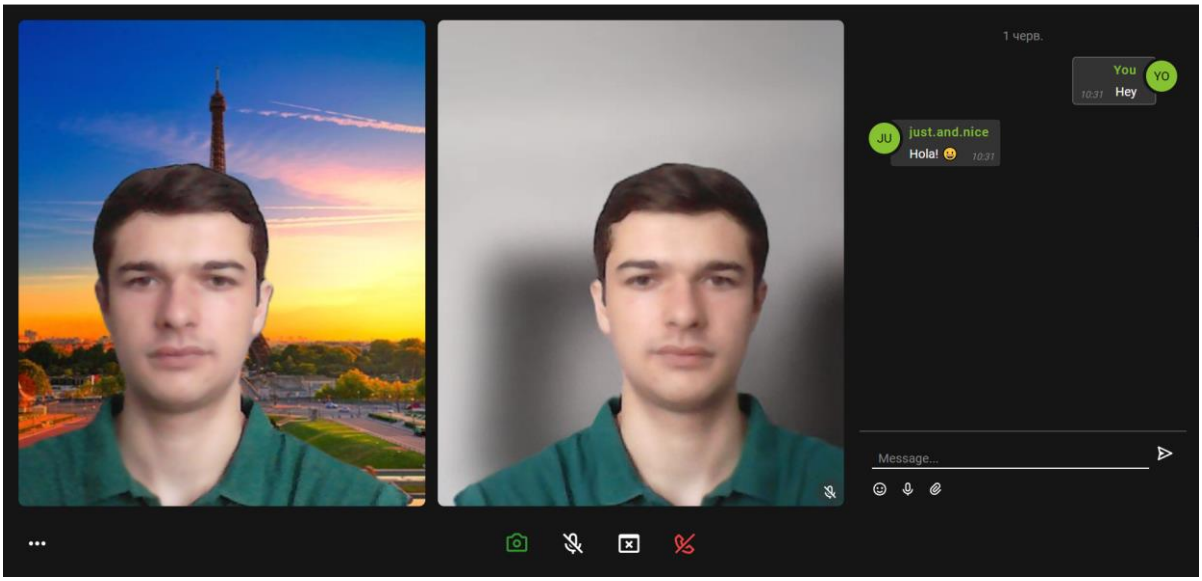

Рисунок 2.15 – Приєднання до медіаканалу

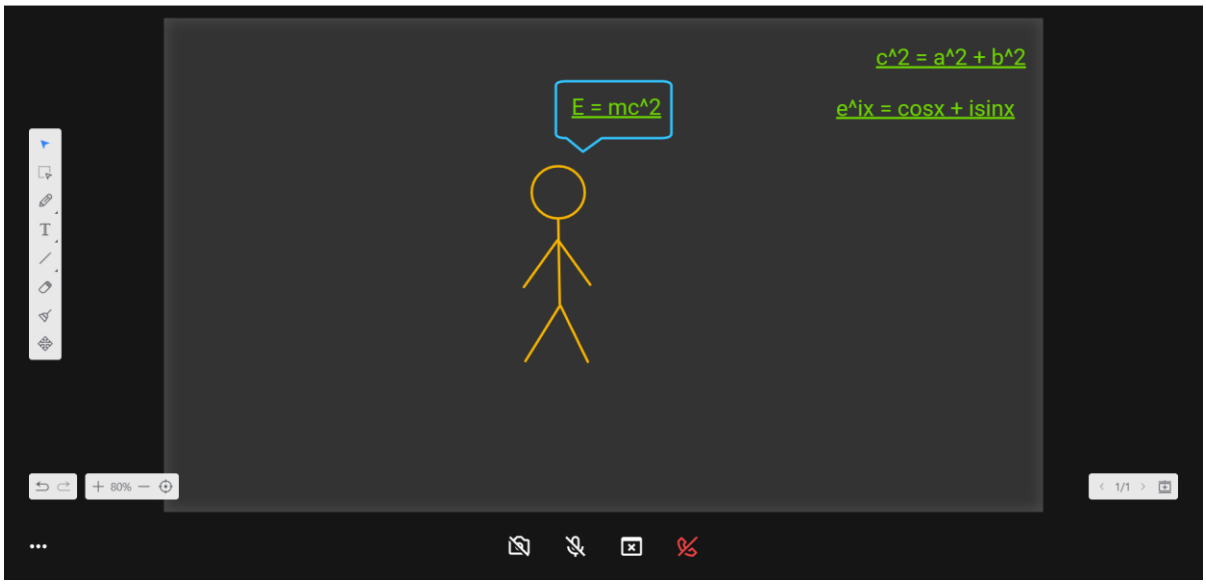

Рисунок 2.16 – Сторінка медіаканалу з увімкненою інтерактивною дошкою

Користувач може використовувати такі інструменти для взаємодії з інтерактивною дошкою як курсор, виділення певної області, олівець, текст, фігури,

гумка, очищення вмісту сторінки, зміщення сторінки; кнопки для скасування/повторення дії, збільшення/зменшення області перегляду, наближення області перегляду за замовчуванням та додавання нових сторінок.

Сторінку для надсилання повідомлень в чаті можна побачити на рис. 2.17. Розглянемо приклад надсилання файлового повідомлення в чаті. Програмний код подано на рис. 2.18.

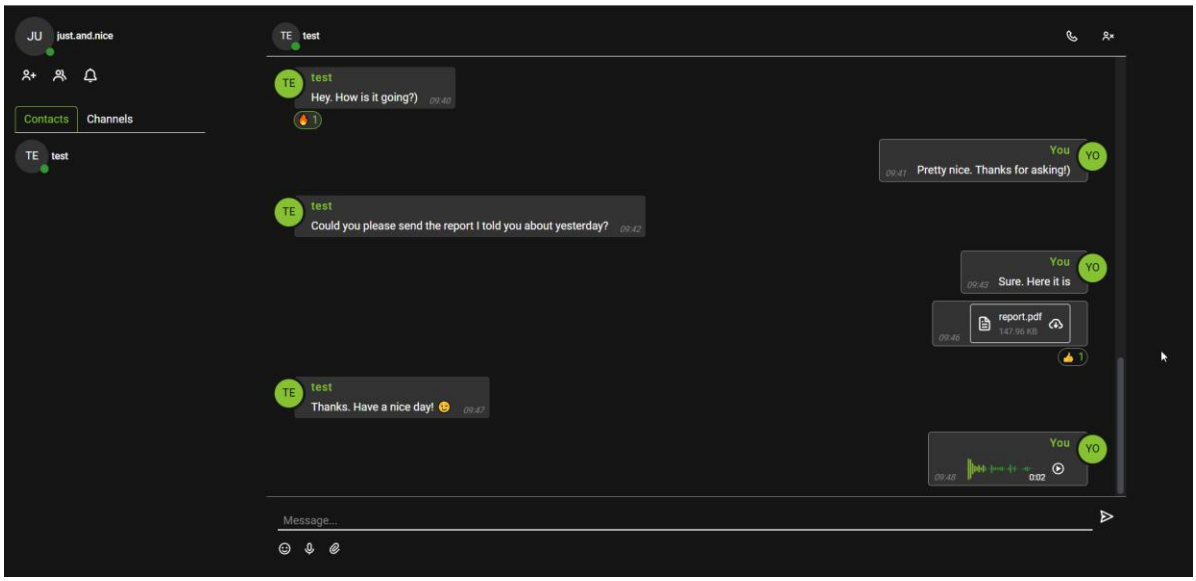

Рисунок 2.17 – Вигляд сторінки текстового чату

```
const handleFileMessageSend = async (
  type: "img" | "audio" | "video" | "file",
  file: AgoraChat.FileObj,
) => {
  const options: AgoraChat.CreateMsgType = {
    type,
    file,
    filename: file.filename,
    to: targetId,
    chatType,
    ext: {
      senderUsername: username ?? "unknown",
      fileType: file.filetype,
      fileName: file.filename,
      fileSize: file.data.size,
    },
    onFileUploadComplete: (event: AgoraChat.UploadFileResult) =>
{
      file.url = event.url;
    },
  };

  try {
    const msg = AC.message.create(options);
    const { serverMsgId } = await connection.send(msg);
    setMessages((prev) => [
```

Рисунок 2.18 – Надсилання файлового повідомлення, аркуш 1

```
        ...prev,
        {
          id: serverMsgId,
          type,
          file,
          fileType: file.filetype,
          fileName: file.filename,
          fileSize: file.data.size,
          url: file.url,
          senderId: userId ?? "",
          senderUsername: username ?? "unknown",
          recipientId: msg.to,
          time: Date.now(),
        },
      ]);
    } catch (err) {
      alert.error(
        `An error occurred while sending ${type} message. Please
try again`,
      );
      console.log(err);
    }
};
```

Рисунок 2.18, аркуш 2

Функція handleFileMessageDispatch обробляє надсилання повідомлення з файлом у чаті. Вона створює об’єкт повідомлення з відповідними параметрами, такими як тип файлу, ім’я файлу, розмір файлу тощо. Потім повідомлення відправляється за допомогою connection.send(msg). Якщо відправлення успішне, повідомлення додається до списку повідомлень у чаті. Якщо виникає помилка, користувачу показується відповідне повідомлення.

Сторінку для здійснення запиту на скидання пароля показано на рис. 2.19.

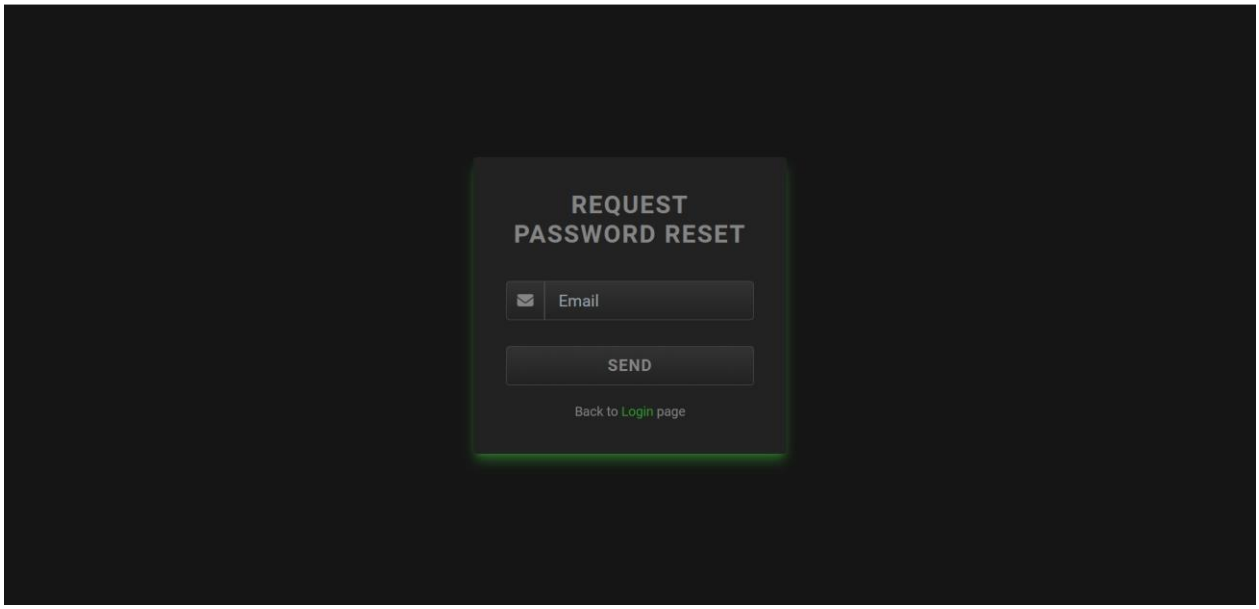

Рисунок 2.19 – Вигляд сторінки для запиту на скидання пароля

Для здійснення запиту на скидання пароля розглянемо фрагмент програмного коду, що поданий на рис. 2.20.

```
const onSubmit: SubmitHandler<FormValues> = async (data) => {
  if (countdown > 0 || isLoading) return;
  if (countdown === 0) setCountdown(60);

  try {
    data.email = data.email.toLowerCase();
    await requestResetPassword(data).unwrap();
    alert.success("Email with a reset password link is sent");
  } catch (err) {
    console.log(err);

    if (isFetchBaseQueryError(err)) {
      alert.error(err.data as string);
    } else if (isErrorWithMessage(err)) {
      alert.error(err.message);
    }

    setCountdown(0);
  }
};
```

Рисунок 2.20 – Запит на скидання пароля

Функція onSubmit обробляє запит на скидання пароля. Вона перевіряє, чи не триває вже зворотний відлік або завантаження, і якщо ні, то ініціалізує зворотний відлік на 60 секунд. Потім надсилає запит на скидання пароля за допомогою функції requestResetPassword, яка відправляє електронного листа із посиланням на сторінку для скидання пароля. Якщо запит успішний, відображається повідомлення про успіх. У разі помилки відображається відповідне повідомлення про помилку.

Отриманий електронний лист зображено на рис. 2.21. У цьому листі міститься посилання для відновлення пароля, яке користувач може використовувати, щоб перейти на спеціальну сторінку. Вигляд сторінки для задання нового пароля можна побачити на рис. 2.22. Ця сторінка включає поля для введення нового пароля та його підтвердження, що забезпечує надійність процесу відновлення пароля. Для обробки відповідного запиту викликається функція, що подана на рис. 2.23.

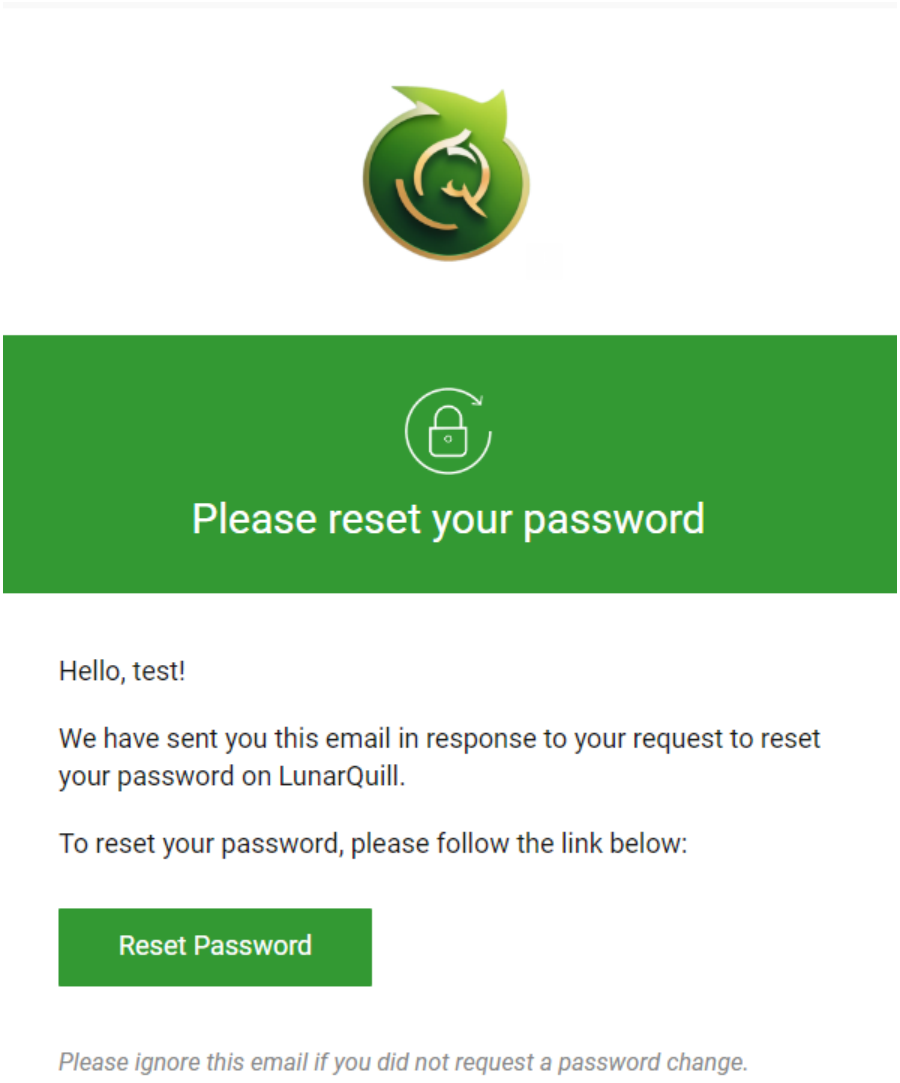

Рисунок 2.21 – Електронний лист із посиланням на сторінку для скидання пароля

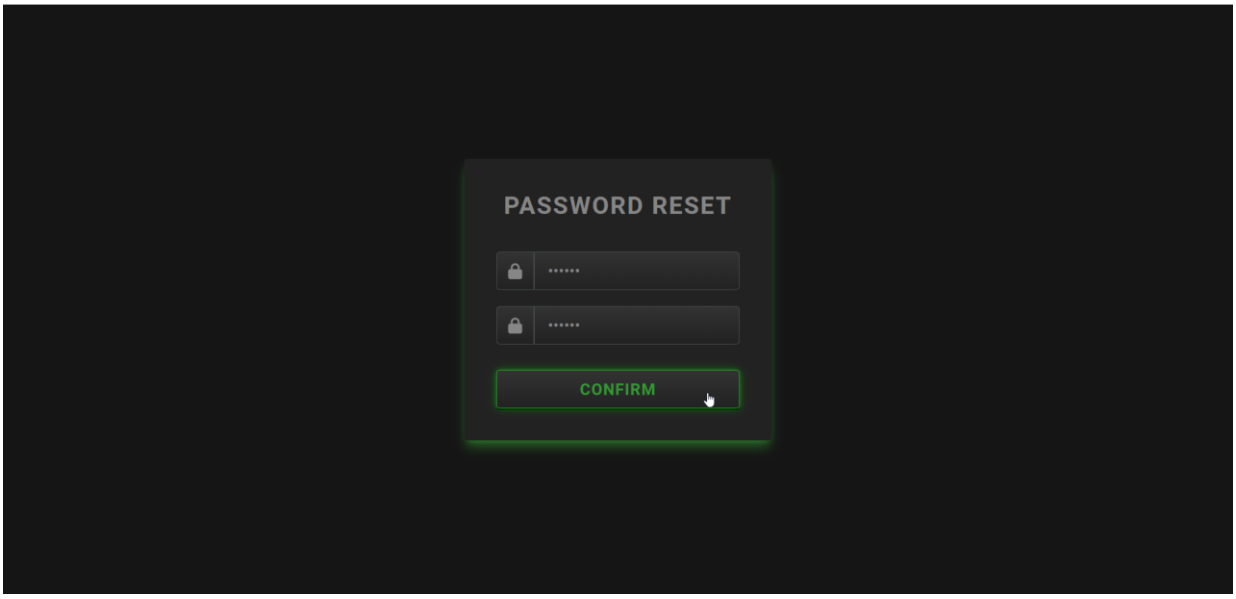

Рисунок 2.22 – Сторінка для скидання пароля

```
const requestPasswordReset = async (email: string) => {
  // Form validation
  const { error, isValid } = validateRequestPasswordResetInput({
    email,
  });

  // Check validation
  if (!isValid) throw new Error(error);

  const user = await User.findOne({ email });

  if (!user) throw new Error("Email does not exist");

  // Check if there's an existing token within the last minute
  const lastToken = await Token.findOne({
    userId: user._id,
    createdAt: { $gte: Date.now() - 60000 }, // Within the last
minute
  });

  if (lastToken) {
    const timeRemaining = Math.ceil(
      (lastToken.createdAt.getTime() + 60000 - Date.now()) /
1000,
    );

    throw new Error(
      `Please wait ${timeRemaining} seconds before requesting
another password reset`,
    );
  }

  const token = await Token.findOne({ userId: user._id });

  if (token) await token.deleteOne();

  const resetToken = createToken();
  const hash = await hashString(resetToken);

  await new Token({
    userId: user._id,
    token: hash,
    createdAt: Date.now(),
  }).save();

  const link = `${clientURL}/password-
reset?token=${resetToken}&id=${user._id}`;

  sendEmail(
    user.email,
    "Password Reset Request",
    {
      username: user.username,
      link,
    },
    "./templates/requestPasswordReset.handlebars",
  );

  return { link };
};
```

Рисунок 2.23 – Обробка запиту на скидання пароля

Функція requestPasswordReset обробляє запит на скидання пароля. Вона перевіряє, чи введена електронна адреса відповідає необхідним вимогам, шукає користувача з цією електронною адресою в базі даних і перевіряє, чи не було створено токен для цього користувача протягом останньої хвилини. Якщо токен вже існує, він видаляється. Потім створюється новий токен для скидання пароля і зберігається в базі даних. Користувачеві відправляється електронний лист зі скидання пароля, що містить посилання на відповідну сторінку.

Отже, у цьому розділі було здійснено проєктування та розробку вебзастосунку для комунікації в режимі реального часу. Було детально розглянуто архітектуру системи, що базується на клієнт-серверній моделі з використанням технологічного стека MERN (MongoDB, Express, React, Node). Проведено проєктування бази даних, розроблено структуру клієнтської та серверної частин додатку, а також описано основні модулі та їх функціональність. Кожен компонент системи був ретельно спроєктований з урахуванням вимог до продуктивності, безпеки та зручності використання.

Окремо було розглянуто реалізацію ключових варіантів використання, що демонструють зв’язки між різними компонентами системи, а також забезпечують надійну та ефективну роботу вебзастосунку. Графічні представлення у вигляді UML-діаграм допомогли візуалізувати процеси та взаємодії між компонентами.

В результаті проєктування та розробки було створено функціональний вебзастосунок, який відповідає сучасним вимогам та стандартам, забезпечуючи високу якість та зручність використання для кінцевих користувачів.

3 ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЯ ВИМОГ

Тестування та верифікація вимог є важливими етапами розробки програмного забезпечення, оскільки вони забезпечують якість та надійність кінцевого продукту. Під час цього процесу проводиться ретельна перевірка як серверної, так і клієнтської частин системи.

3.1 Тестування серверної частини вебсистеми

Тестування серверної частини є критичним, оскільки сервер забезпечує життєво необхідну частину логіки роботи застосунку, управління базою даних та взаємодію з зовнішніми сервісами. Недоліки в серверній частині можуть призвести до серйозних проблем, таких як втрата даних, збої в роботі системи або порушення безпеки.

Для здійснення тестування серверної частини інформаційної системи створено п’ять тест-кейсів (таблиця 3.1) та використано інструмент для тестування API Postman.

Postman є популярним інструментом для тестування API, що дозволяє розробникам створювати, надсилати та аналізувати HTTP-запити [16]. Використання Postman має кілька ключових переваг: зручний інтерфейс для налаштування запитів, підтримка автоматизації тестування через колекції запитів, можливість аналізувати відповіді сервера, вбудований редактор скриптів для додавання тестів, та інтеграція з системами безперервної інтеграції та доставки. Завдяки цим можливостям Postman є незамінним інструментом для тестування та верифікації API, забезпечуючи високу якість та надійність серверної частини програмного забезпечення.

Таблиця 3.1 – Тест-кейси для тестування серверної частини вебсистеми

ID	Тип	Опис	Очікуваний результат
1	positive	Перевірка сервісної функції для запиту на скидання пароля	Посилання на сторінку для скидання пароля
2	positive	Перевірка сервісної функції для отримання списку відкритих каналів	Список відкритих каналів, що відповідають пошуковому запиту
3	negative	Перевірка сервісної функції для створення каналу (здійснення запиту без задання токена для авторизації)	Повідомлення із зазначенням помилки
4	positive	Перевірка сервісної функції для генерування токена для приєднання до медіаканалу	Токен для приєднання до медіаканалу
5	positive	Перевірка сервісної функції для оновлення даних про користувача	Отримання оновлених даних для заданого користувача

Тест-кейс 1. Для перевірки цієї сервісної функції було введено електронну адресу, на яку прийде повідомлення із посиланням на сторінку для зміни пароля. Також це посилання буде надано як відповідь на здійснений запит. Результат виконання запиту можна побачити на рис. 3.1.

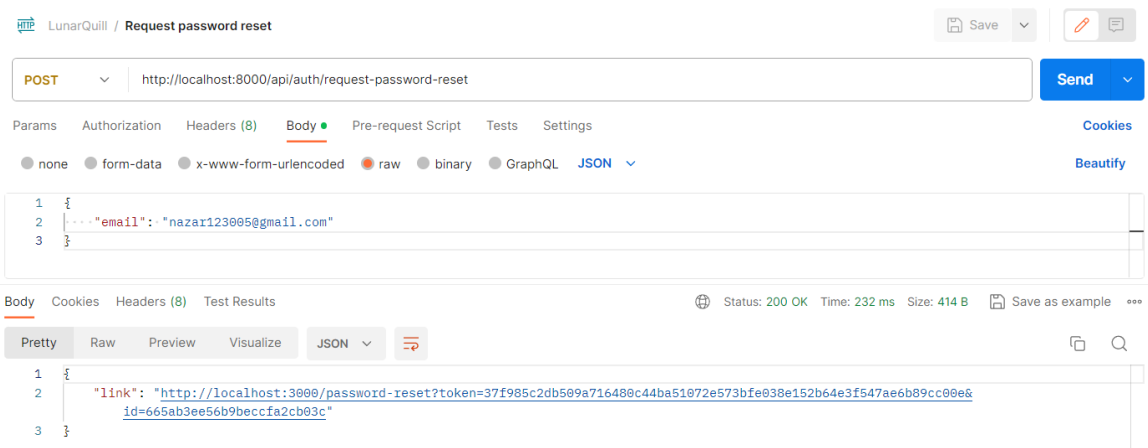

Рисунок 3.1 – Результат запиту на отримання посилання для зміни пароля

Як видно з рис. 3.1 запит є успішним (код 200). Як результат було отримано відповідний лист на зазначену електронну пошту та посилання для переходу на сторінку для скидання пароля.

Тест-кейс 2. Для отримання списку каналів нам необхідно задати пошукову стрічку та токен для здійснення авторизації, оскільки дана сервісна функція доступна лише авторизованим користувачам. Результат виконання запиту подано на рис. 3.2.

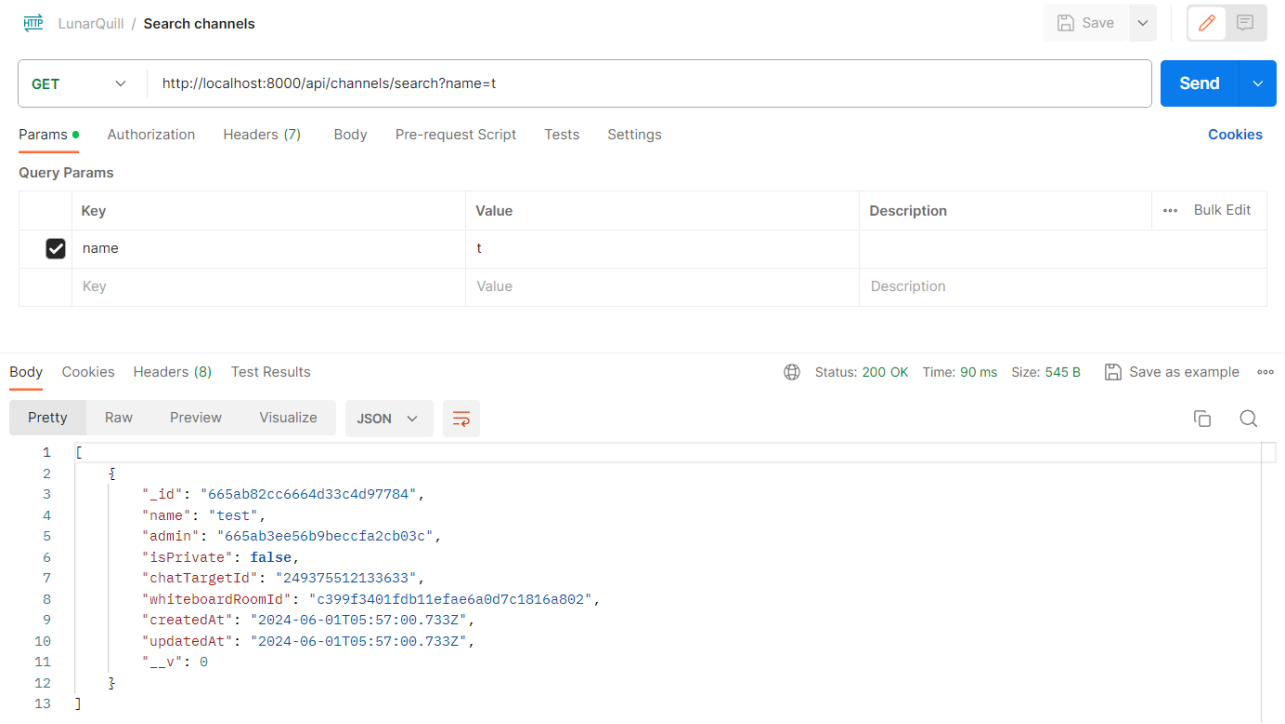

Рисунок 3.2 – Результат виконання запиту на отримання списку відкритих каналів за введеною пошуковою стрічкою

Результатом виконання запиту є отримання одного документа, в якому знаходиться інформація про відповідний відкритий канал.

Тест-кейс 3. Запит на створення каналу здійснюється, як зазначено в табл. 3.1, без задання токена для авторизації, а отже, не може бути успішно виконаним. Результат здійснення відповідного запиту показано на рис. 3.3.

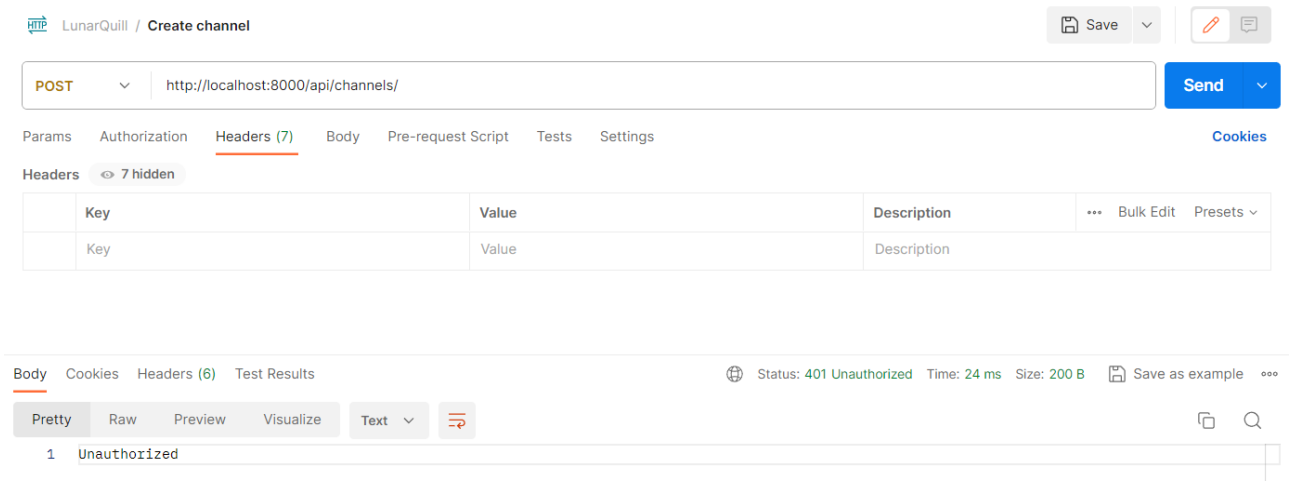

Рисунок 3.3 – Запит на створення каналу без задання відповідних даних для авторизації

Як видно з рис. 3.3, отримано повідомлення про помилку авторизації, що свідчить про правильну поведінку системи в таких випадках.

Тест-кейс 4. Для здійснення цього запиту сформовано відповідну URL адресу, а також задано необхідний токен для авторизації. Результат виконання запиту можна побачити на рис. 3.4.

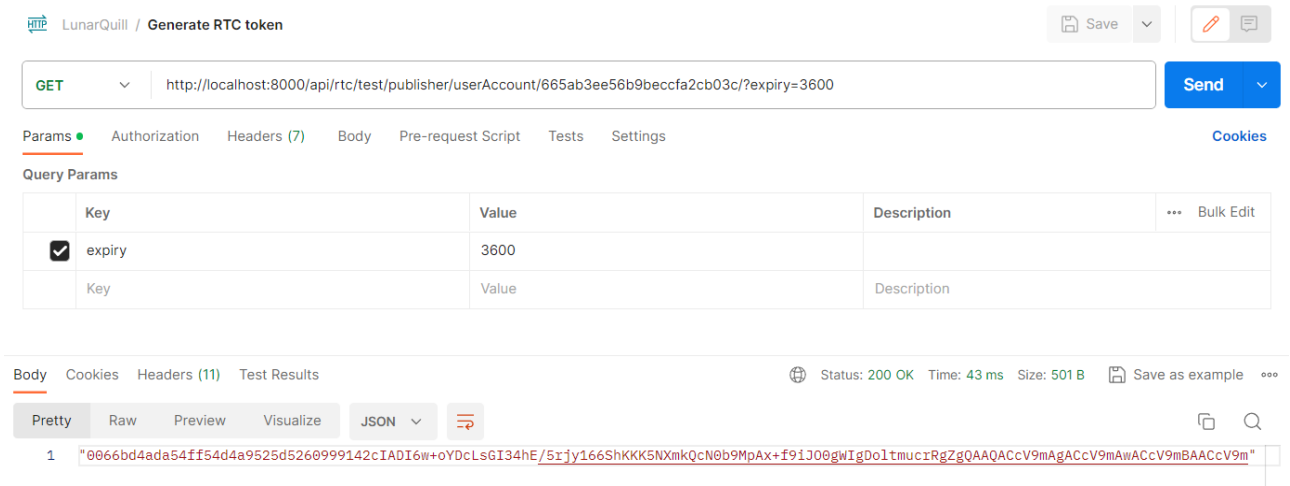

Рисунок 3.4 – Отримання токена для приєднання до медіаканалу

З рис. 3.4 видно, що відповідь від сервісної функції як результат здійснення відповідного запиту є позитивною (код 200) та отримано необхідний токен.

Тест-кейс 5. Задано унікальний ідентифікатор користувача в URL адресі, токен для авторизації та частину документа користувача, що повинна бути оновленою. Результат здійснення запиту показано на рис. 3.5.

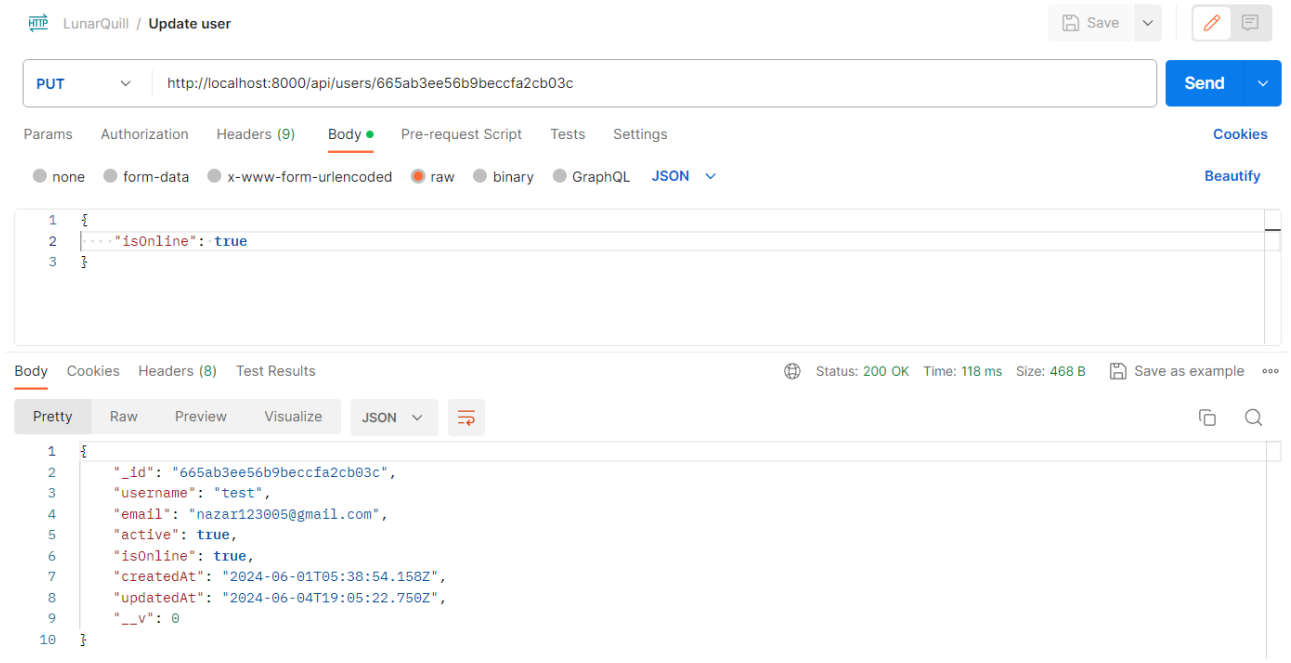

Рисунок 3.5 – Результат виконання запиту на оновлення даних про користувача

Як видно з рис. 3.5 ми отримали оновлені дані про заданого користувача, що свідчить про правильну роботу цієї сервісної функції та підтверджує виконання заданого тест-кейсу.

3.2 Тестування клієнтської частини вебсистеми

Тестування клієнтської частини є не менш важливим, оскільки користувацький інтерфейс є обличчям застосунку для кінцевого користувача. Важливо забезпечити зручність використання, коректне відображення та функціонування інтерфейсу на різних пристроях та браузерах. Це включає

перевірку всіх інтерактивних елементів, форм, навігації та інших компонентів інтерфейсу.

Для здійснення тестування клієнтської частини вебсистеми створено алгоритм тестування та використано Selenium IDE плагін для браузера Chrome [17]. Додатково перевірено сторінки на адаптивний дизайн.

Selenium IDE дозволяє автоматизувати тестування вебдодатків. Цей інструмент забезпечує простий спосіб запису, редагування та відтворення тестових сценаріїв. Основні переваги Selenium IDE включають можливість швидкого створення тестів без написання коду, підтримку різних команд для взаємодії з елементами на вебсторінці, а також можливість експорту тестів у різні мови програмування для подальшої інтеграції з автоматизованими тестовими фреймворками. Використання Selenium IDE допомагає забезпечити якість і стабільність вебдодатків за допомогою автоматизованого тестування.

Тестування здійснено за таким алгоритмом:

1. Ввійти в обліковий запис.
2. Встановити онлайн-статус.
3. Додати контакт.
4. Написати повідомлення до контакту.
5. Здійснити медіавиклик.
6. Розпочати демонстрацію екрана.
7. Завершити демонстрацію екрана.
8. Здійснити операцію на інтерактивній дошці.
9. Завершити медіавиклик.

Результат виконання тестування клієнтської частини інформації системи показано на рис. 3.6.

Як видно з рис. 3.6 здійснення тестування системи за вищеподаним алгоритмом є успішним, що вказує на верифікацію, виконання та валідацію вимог.

Вигляд головної сторінки для неавторизованих користувачів та сторінки з текстовим чатом типу один-до-одного з адаптивним дизайном можна побачити на рис. 3.7 та 3.8 відповідно.

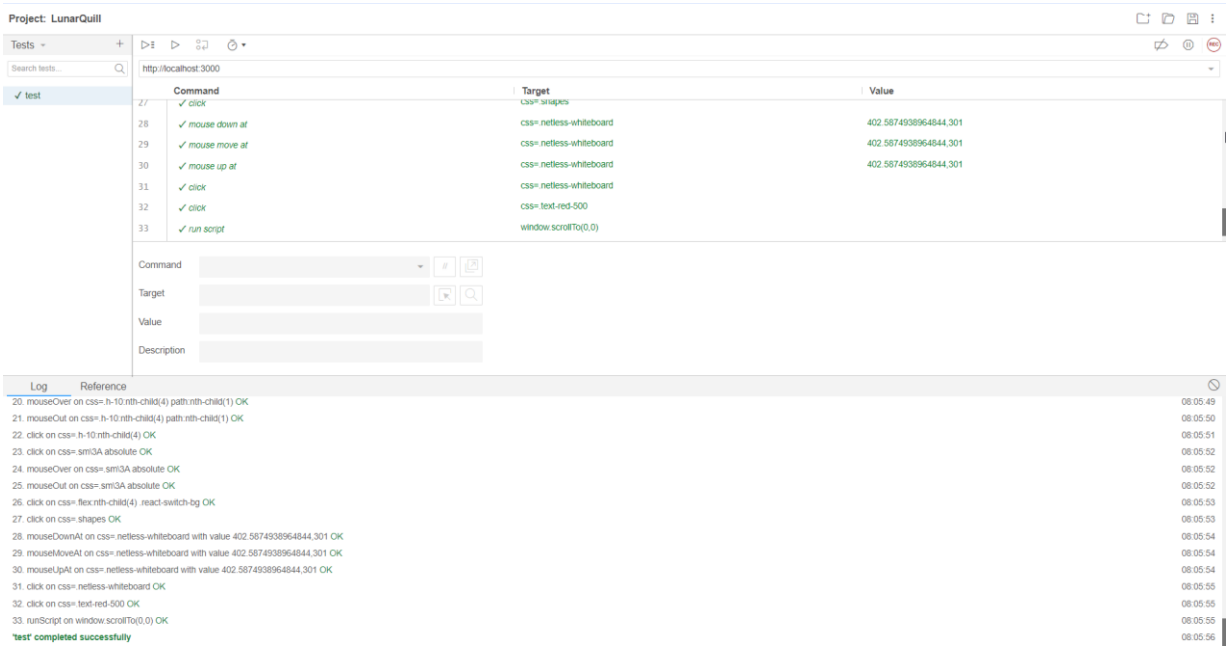

Рисунок 3.6 – Результат тестування клієнтської частини вебсистеми

Рисунок 3.7 – Вигляд головної сторінки для мобільних користувачів

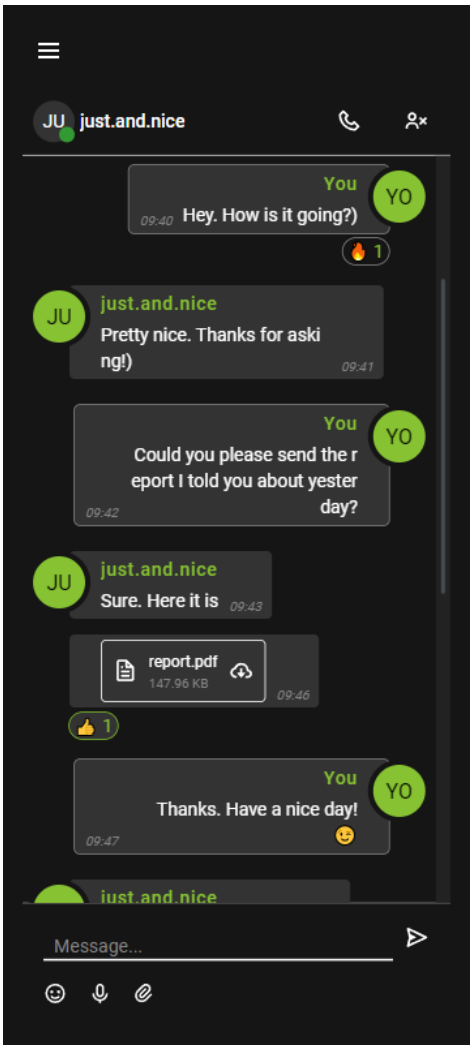

Рисунок 3.8 – Сторінка з текстовим чатом для мобільних користувачів

Отже, як можна побачити з рис. 3.7 та 3.8, в інформаційній системі створено адаптивний дизайн, який забезпечує зручність користування на різних пристроях відповідно до нефункціональних вимог. Система автоматично підлаштовується під розмір екрана користувача, що підвищує її доступність та зручність незалежно від типу пристрою.

3.3 Верифікація та валідація вимог

У цьому підрозділі розглянемо результати тестування інформаційної системи, її відповідність вимогам та аналіз результатів.

Верифікація та валідація вимог є важливим етапом процесу розробки програмного забезпечення, який дозволяє підтвердити, що система відповідає всім встановленим вимогам та очікуванням користувачів. Під час тестування проводиться детальна перевірка як функціональних, так і нефункціональних вимог, що дає можливість виявити та усунути будь-які недоліки або невідповідності.

На основі результатів тестування, описаних у попередніх підрозділах, можна зробити висновки щодо відповідності системи вимогам:

– відповідність функціональним вимогам: система успішно реалізує основні функції, такі як додавання контакту, створення каналу, передавання аудіо/відео потоків, надсилання повідомлень в чаті та запит на скидання пароля. Це було підтверджено відповідними тестами, результати яких були позитивними. Використання інструментів для тестування API (Postman) та автоматизації тестування клієнтської частини (Selenium IDE) допомогло забезпечити високу якість виконання цих функцій;
– відповідність нефункціональним вимогам: система демонструє належну продуктивність та безпеку, відповідаючи всім визначеним нефункціональним вимогам. Тестування серверної частини показало, що всі сервісні функції працюють коректно і швидко реагують на запити користувачів. Інтерфейс вебдодатка є адаптивним і коректно відображається на різних пристроях, що було підтверджено тестами клієнтської частини;
– аналіз результатів: усі тест-кейси, включаючи позитивні та негативні сценарії, були успішно виконані, що свідчить про високу якість реалізації системи та її відповідність визначеним вимогам. Позитивні результати тестування підтверджують, що система є надійною, функціональною та готовою до використання кінцевими користувачами.

Отже, на основі проведеного тестування та верифікації вимог, можна зробити висновок, що розроблена інформаційна система повністю відповідає встановленим вимогам та готова до впровадження та використання у реальних умовах.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Роль центральної нервової системи в трудовій діяльності людини

Центральна нервова система (ЦНС) відіграє ключову роль у забезпеченні функціонування людського організму, особливо в контексті трудової діяльності. ЦНС складається з головного і спинного мозку, які забезпечують управління всіма аспектами життєдіяльності людини, включаючи рухову активність, сенсорну обробку інформації, когнітивні функції та емоційні реакції. У трудовій діяльності, що включає різноманітні фізичні й розумові завдання, роль ЦНС є критично важливою для ефективного виконання робочих обов'язків.

Однією з головних функцій ЦНС у трудовій діяльності є координація рухів. Головний мозок отримує сенсорну інформацію від органів чуття, аналізує її та надсилає команди до м'язів через спинний мозок. Це забезпечує точність і злагодженість рухів, необхідних для виконання різних завдань, від простих фізичних дій до складних маніпуляцій з технічними пристроями. Наприклад, робота на заводі вимагає швидкої і точної реакції на зміну умов, що можливо завдяки ефективному функціонуванню ЦНС. Забезпечення безпечних і нешкідливих умов праці також є критично важливим для підтримання нормального функціонування ЦНС та загального здоров'я працівників. Відповідно до Закону України «Про охорону праці», на всіх підприємствах повинні створюватися умови, що відповідають вимогам охорони праці, включаючи санітарно-гігієнічні заходи для запобігання професійним захворюванням і травматизму [18].

Крім координації рухів, ЦНС також відповідає за когнітивні процеси, що є основою для розумової діяльності. Когнітивні функції включають пам'ять, увагу, логічне мислення, планування і прийняття рішень. У багатьох професіях, особливо тих, що вимагають високої кваліфікації, ці процеси є вирішальними для успішного виконання робочих обов'язків. Наприклад, у професії програміста важливо вміти швидко аналізувати інформацію, розробляти алгоритми й приймати рішення в умовах обмеженого часу. Для деяких категорій працівників, таких як молодь або

ті, що зайняті на шкідливих роботах, законодавством встановлена скорочена тривалість робочого часу, що сприяє збереженню їх здоров'я. Скорочена тривалість робочого часу встановлюється, наприклад, для працівників віком від 16 до 18 років – 36 годин на тиждень, а для осіб віком від 15 до 16 років – 24 години на тиждень [19].

Емоційна регуляція, якою також займається ЦНС, відіграє важливу роль у трудовій діяльності. Емоції можуть впливати на продуктивність праці, мотивацію і здатність до співпраці в команді. Гіпоталамус і лімбічна система, що є частинами ЦНС, відповідають за регуляцію емоційних станів. Позитивні емоції, такі як задоволення від виконаної роботи, можуть підвищити продуктивність і мотивацію, тоді як стрес і негативні емоції можуть мати зворотний ефект. Важливо також надавати працівникам можливість для перерви і відпочинку протягом робочого дня, що сприяє зниженню стресу і покращенню загального стану працівників. Згідно з Кодексом законів про працю України, працівникам надається перерва для відпочинку і харчування тривалістю не більше двох годин, яка, як правило, надається через чотири години після початку роботи [19].

Ще одним важливим аспектом є роль ЦНС у навчанні та адаптації до нових умов. Пластичність мозку дозволяє людині пристосовуватися до змін в робочому середовищі, освоювати нові навички й знання. Цей процес є особливо важливим в умовах швидких технологічних змін, коли постійне навчання стає необхідністю для підтримання конкурентоспроможності на ринку праці.

Забезпечення медичного захисту та санітарного й епідемічного добробуту населення включає надання медичної допомоги та проведення медико-психологічної реабілітації, що є важливими заходами для підтримки здоров'я працівників у надзвичайних умовах. Це включає надання медичної допомоги постраждалим внаслідок надзвичайних ситуацій, рятувальникам та іншим особам, які залучалися до виконання аварійно-рятувальних та інших невідкладних робіт [20]. Підтримка здоров'я ЦНС в таких умовах є критично важливою для забезпечення здатності працівників ефективно виконувати свої обов'язки.

Отже, центральна нервова система відіграє критичну роль у всіх аспектах трудової діяльності людини. Вона забезпечує координацію рухів, підтримує когнітивні функції, регулює емоційні стани та сприяє адаптації до нових умов. Розуміння функцій ЦНС і їх впливу на трудову діяльність дозволяє ефективніше організовувати робочі процеси та створювати сприятливі умови для підвищення продуктивності праці. Крім того, забезпечення безпечних і нешкідливих умов праці є обов'язком роботодавця, що передбачає створення таких умов праці, які відповідають вимогам нормативних актів про охорону праці [19].

4.2 Соціальне значення охорони праці

Соціальне значення охорони праці полягає у підвищенні ефективності суспільного виробництва через безперервне поліпшення та оптимізацію умов праці, підвищення рівня безпеки на робочих місцях, а також у зменшенні випадків виробничого травматизму і професійних захворювань [21]. Охорона праці сприяє підвищенню продуктивності, збереженню трудових ресурсів та збільшенню сукупного національного продукту, що є основними показниками розвитку суспільного виробництва.

Одним з ключових аспектів соціального значення охорони праці є підвищення продуктивності праці. Це досягається збільшенням робочого часу завдяки зменшенню внутрішньозмінних простоїв, усуненню мікротравм або зниженню їхньої чисельності, введенням оптимальних режимів праці та відпочинку, запобіганню передчасній втомі шляхом оптимізації умов праці, а також сукупності інших заходів, що підвищують ефективність розподілу робочого часу.

На підприємствах харчової промисловості, де кожна мікротравма може супроводжуватися втратою до двох годин робочого часу, заходи з охорони праці мають особливе значення. Крім того, скорочення цілоденних втрат робочого часу

досягається зниженням рівня або ліквідацією тимчасової непрацездатності через виробничий травматизм, професійну та загальну захворюваність. Для харчових виробництв це особливо актуально, оскільки кожна травма може призводити до втрати працездатності на понад 26 днів.

Зменшення кількості травм та захворювань також сприяє зниженню витрат на медичне обслуговування та соціальне забезпечення, що додатково покращує економічні показники. Таким чином, охорона праці не тільки сприяє підвищенню продуктивності, але й економить значні кошти для підприємств і держави.

Охорона праці також має значний вплив на збереження трудових ресурсів і підвищення професійної активності працівників. Поліпшення стану здоров'я працівників і збільшення середньої тривалості їх життя досягається шляхом покращення умов праці, що також супроводжується збільшенням виробничого стажу працівників при їх високій трудовій активності. Збереження здоров'я працівників дозволяє їм довше залишатися в професії, що сприяє накопиченню досвіду і підвищенню кваліфікації.

Підвищення професійного рівня працівників також відбувається завдяки зростанню їхньої кваліфікації та майстерності як результат збільшення стажу роботи. Залучення пенсіонерів та інвалідів до виконання доступних для них завдань, використовуючи їхню залишкову трудову активність, багатий практичний досвід та професійні знання, є важливим аспектом збереження трудових ресурсів. Забезпечення відповідних умов праці для цих категорій працівників дозволяє їм продовжувати робочу діяльність та робити свій внесок у розвиток виробництва.

Окрім цього, підвищення безпеки праці сприяє зменшенню стресових факторів на робочому місці. Це має позитивний вплив на психологічний стан працівників, знижуючи ризики емоційного вигорання та інших психічних розладів. Стабільний емоційний стан, зі свого боку, підвищує загальну продуктивність та знижує кількість випадків непрацездатності через психічні розлади.

Додатково, покращені умови праці можуть залучити більше кваліфікованих працівників, оскільки люди зацікавлені працювати в безпечному і здоровому

середовищі. Це підвищує конкуренцію серед працівників за робочі місця, що стимулює їх до підвищення своєї кваліфікації та професійного рівня.

Зрештою, охорона праці сприяє збільшенню сукупного національного продукту внаслідок поліпшення вищезазначених показників та їхніх складових компонентів. Зростання продуктивності праці, збереження трудових ресурсів і підвищення професійної активності працівників призводять до більш ефективного використання робочого часу і ресурсів, що в результаті сприяє економічному зростанню.

Важливим аспектом є те, що охорона праці також сприяє зниженню витрат на медичне обслуговування та соціальне забезпечення, оскільки зменшується кількість випадків виробничих травм та професійних захворювань. Це позитивно впливає на економічні показники не тільки окремих підприємств, але й держави в цілому.

Соціальне значення охорони праці проявляється у впливі на зміну таких основних показників, що характеризують рівень розвитку суспільного виробництва як зростання продуктивності праці, збереження трудових ресурсів і збільшення сукупного національного продукту. Це підкреслює важливість інвестування в охорону праці як на рівні окремих підприємств, так і на рівні держави.

Забезпечення безпечних умов праці не лише підвищує якість життя працівників, але й сприяє створенню стабільного, продуктивного і розвинутого суспільства, де кожен може реалізувати свій потенціал і зробити свій внесок у спільний добробут. У результаті, охорона праці стає не просто вимогою закону, а стратегічною інвестицією у майбутнє будь-якої нації, що прагне до сталого розвитку та процвітання.

Підсумовуючи, соціальне значення охорони праці охоплює широкий спектр впливу на економіку та суспільство. Воно сприяє підвищенню продуктивності, збереженню здоров’я працівників, зниженню витрат на медичне обслуговування, підвищенню професійного рівня та загальному економічному зростанню. Це підкреслює необхідність продовження розвитку та вдосконалення заходів з охорони праці на всіх рівнях управління та виробництва.

ВИСНОВКИ

В результаті виконання даної кваліфікаційної роботи було розроблено вебзастосунок для комунікації в режимі реального часу, який дозволяє здійснювати обмін текстовими повідомленнями, аудіо та відеодзвінками, а також використовувати інтерактивну дошку для спільної роботи.

У першому розділі було проведено детальний аналіз предметної області, визначено основні вимоги до системи, розглянуто акторів та прецеденти, а також виконано аналіз схожих рішень на ринку. Це дозволило сформулювати чітке розуміння необхідної функціональності та технологій, що повинні бути використані для досягнення поставлених цілей.

У другому розділі здійснено проєктування та розробку системи. Було детально розглянуто архітектуру, створено моделі бази даних, розроблено серверну та клієнтську частини додатка. Крім того, були реалізовані ключові варіанти використання, що забезпечують основну функціональність системи, включаючи додавання нових контактів, приєднання до каналів, передачу аудіо/відео потоків та інші.

Третій розділ присвячено тестуванню та верифікації розробленої системи. Було використано різні методи тестування, включаючи функціональне, інтеграційне та автоматизоване тестування, для перевірки відповідності системи вимогам, її надійності та продуктивності. Верифікація вимог показала, що розроблена система відповідає визначеним критеріям якості та готова до використання.

В результаті виконання роботи було створено функціональний та надійний вебзастосунок, який відповідає сучасним стандартам та вимогам. Система забезпечує високу якість комунікації та зручність використання для кінцевих користувачів, що дозволяє ефективно взаємодіяти незалежно від їх місцеперебування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Slack Statistics 2024: Users, Revenue, Growth. *DemandSage*. URL: https://www.demandsage.com/slack-statistics/ (дата звернення: 15.04.2024).

2. Microsoft Teams Revenue and Usage Statistics (2024). *Business of Apps*. URL: https://www.businessofapps.com/data/microsoft-teams-statistics/ (дата звернення: 15.04.2024).

3. Zoom daily meeting participants worldwide 2020 | Statista. *Statista*. URL: https://www.statista.com/statistics/1253972/zoom-daily-meeting-participants-global/ (дата звернення: 15.04.2024).

4. Discord Revenue and Usage Statistics (2024). *Business of Apps*. URL: https://www.businessofapps.com/data/discord-statistics/ (дата звернення: 15.04.2024).

5. Web Real-Time Communication Market Size & Forecast 2032. *Market Research Company, Reports and Consulting Services | IMARC*. URL: https://www.imarcgroup.com/web-real-time-communication-market (дата звернення: 15.04.2024).

6. Welcome To UML Web Site!. *Welcome To UML Web Site!*. URL: https://www.uml.org/ (дата звернення: 25.04.2024).

7. React. *React*. URL: https://react.dev/ (дата звернення: 27.04.2024).

8. Redux - A JS library for predictable and maintainable global state management | Redux. *Redux - A JS library for predictable and maintainable global state management | Redux*. URL: https://redux.js.org/ (дата звернення: 27.04.2024).

9. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/ (дата звернення: 27.04.2024).

10. JavaScript With Syntax For Types. *TypeScript: JavaScript With Syntax For Types.* URL: https://www.typescriptlang.org/ (дата звернення: 27.04.2024).

11. Node.js – Run JavaScript Everywhere. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/en (дата звернення: 27.04.2024).

12. Express - Node.js web application framework. *Express - Node.js web application framework*. URL: https://expressjs.com/ (дата звернення: 27.04.2024).

13. Mongoose ODM v8.4.1. *Mongoose ODM v8.4.1*. URL: https://mongoosejs.com/ (дата звернення: 27.04.2024).

14. MongoDB: The Developer Data Platform. *MongoDB*. URL: https://mongodb.com/ (дата звернення: 27.04.2024).

15. Agora.io Real-Time Voice and Video Engagement. *Agora*. URL: https://www.agora.io/en/ (дата звернення: 27.04.2024).

16. Postman documentation overview | Postman Learning Center. *Postman Learning Center*. URL: https://learning.postman.com/docs/introduction/overview/ (дата звернення: 10.05.2024).

17. Selenium IDE · Open source record and playback test automation for the web. *Selenium.* URL: https://www.selenium.dev/selenium-ide/ (дата звернення: 10.05.2024).

18. Про охорону праці. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 07.06.2024).

19. Кодекс законів про працю України. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 07.06.2024).

20. Кодекс цивільного захисту України. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/5403-17#Text (дата звернення: 07.06.2024).

21. Конспект лекцій з дисципліни «Соціально-економічні основи охорони праці» (для студентів 4-го курсу денної та 3-го курсу заочної форм навчання, напряму підготовки 6.170202 – Охорона праці) / О. Ю. Нікітченко; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2015. – С. 43.

ДОДАТКИ

ДОДАТОК А – Диск із кваліфікаційною роботою бакалавра