МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ БАГАТОКАНАЛЬНОЇ КОМУНІКАЦІЇ ЗАСОБАМИ RUBY ON RAILS»**

Виконала: студентка 4 курсу, групи 6.1212-пі-2
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
К.Є. Соколова
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Соколовій Каріні Євгенівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку багатоканальної комунікації засобами Ruby on Rails

керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 01.06.2026 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз аналогів, предметної області, вимог та засобів розробки.
2. Моделювання та проєктування системи.
3. Програмна реалізація та тестування системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 09.12.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.12.2025	
2.	Збір вихідних даних.	08.01.2026	
3.	Обробка методичних та теоретичних	30.01.2026	
	джерел.		
4.	Розробка першого та другого розділу.	20.03.2026	
5.	Розробка третього розділу.	11.05.2026	
6.	Оформлення та нормоконтроль	25.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2026	

Студент _____ К.Є. Соколова
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку багатоканальної комунікації засобами Ruby on Rails»: 43 с., 27 рис., 9 джерел.

БАГАТОКАНАЛЬНА КОМУНІКАЦІЯ, ВЕБЗАСТОСУНОК, ВЕБСОКЕТИ, МУЛЬТИМЕДІЙНА КОМУНІКАЦІЯ, CLOUDINARY, POSTGRESQL, RSPEC, RUBY ON RAILS, WEBRTC.

Об’єкт дослідження – процес розробки вебзастосунку багатоканальної комунікації в режимі реального часу.

Предмет дослідження – технології Ruby on Rails, PostgreSQL, Cloudinary.

Мета роботи – створення вебзастосунку для організації зручної та швидкої дистанційної комунікації між користувачами в режимі реального часу.

Методи дослідження – методи об’єктно-орієнтованого програмування, проєктування й конструювання програмного забезпечення, системного аналізу інструментів веброзробки та тестування.

У роботі проведено аналіз сучасних аналогів і предметної області, сформульовано вимоги до системи та спроєктовано структуру та архітектуру вебзастосунку з використанням моделі даних та UML-діаграм. Для реалізації аудіо- та відеодзвінків було використано технологію WebRTC та TURN/STUN-сервери, що забезпечують стабільне з’єднання між користувачами в режимі реального часу. Результатом роботи є реалізація сучасного вебзастосунку багатоканальної комунікації TolskyChat, який забезпечує обмін повідомленнями, передачу мультимедійного контенту, аудіо- та відеозв’язку у режимі реального часу, а також підтримку сповіщень і взаємодії користувачів у групових та приватних бесідах.

SUMMARY

Bachelor’s qualifying paper “Development of a Multichannel Communication Web Application Using Ruby on Rails”: 43 pages, 27 figures, 9 references.

WEB APPLICATION, MULTICHANNEL COMMUNICATION, WEBSOCKETS, MULTIMEDIA COMMUNICATION, RUBY ON RAILS, POSTGRESQL, RSPEC, CLOUDINARY, WEBRTC.

The object of research is the process of developing a real-time multi-channel communication web application.

The subject of research includes Ruby on Rails, PostgreSQL, and Cloudinary technologies.

The aim of the study is to create a web application for convenient and fast real-time remote communication between users.

Research methods include object-oriented programming methods, software design and construction methods, system analysis of web development tools, and application testing.

The work includes an analysis of modern analogs and the subject area, formulates system requirements, and designs the structure and architecture of the web application using a data model and UML diagrams. WebRTC technology and TURN/STUN servers were used to implement audio and video calls, ensuring stable real-time connection between users. The result of the work is the implementation of the modern multi-channel communication web application TolskyChat, which provides messaging, multimedia content sharing, real-time audio and video communication, as well as notifications and user interaction in both group and private conversations.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналіз аналогів, предметної області, вимог та засобів розробки 9
1.1 Актуальність та аналіз систем багатоканальної комунікації 9
1.2 Опис предметної області 10
1.3 Вимоги до проєкту 11
1.4 Опис програмних засобів 14
1.5 Висновки до розділу 1 15
2 Моделювання та проєктування системи..... 16
2.1 Побудова концепції вебзастосунку 16
2.2 Логічне моделювання системи 21
2.3 Проєктування архітектури та компонентів системи 27
2.4 Висновки до розділу 2 29
3 Програмна реалізація та тестування системи..... 30
3.1 Реалізація вебзастосунку..... 30
3.2 Тестування вебзастосунку..... 36
3.3 Загальний огляд системи..... 38
3.4 Висновки до розділу 3 41
Висновки 42
Перелік посилань..... 43

ВСТУП

Зі стрімким розвитком інформаційних технологій відкрились нові можливості спілкування незалежно від місцеперебування співрозмовників, навіть якщо вони перебувають у різних частинах світу. Особливої актуальності дистанційне спілкування набуло з початком пандемії та воєнних подій в Україні. Для розв’язання цього питання важливу роль відіграють вебзастосунки багатоканальної комунікації.

У даній роботі розглядається багатоканальна комунікація як взаємодія користувачів через різні логічні сектори обміну інформацією, такі як приватні діалоги, групові бесіди та системні повідомлення в межах одного застосунку, що реалізується у вигляді чат-орієнтованої системи. Чат є загальновідомим засобом онлайн-комунікацій, який забезпечує обмін інформацією між користувачами в реальному часі. Система надає можливість передавати різні типи повідомлень, зокрема текстові, графічні, аудіо- та відеодані. Багато сучасних вебчатів підтримують аудіо- та відеодзвінки між користувачами.

Метою кваліфікаційної роботи є розробка інформаційної системи «Розробка вебзастосунку багатоканальної комунікації засобами Ruby on Rails».

Для досягнення поставленої мети, необхідно розв’язати такі задачі:

– здійснити аналіз та опис предметної області;
– сформулювати вимоги до проєкту;
– спроєктувати архітектуру інформаційної системи;
– розробити вебзастосунок;
– провести тестування функціональності застосунку.

Кваліфікаційна робота складається з трьох розділів. У першому розділі проведено аналіз предметної області, сформульовано функціональні та нефункціональні вимоги до системи, а також обґрунтовано вибір засобів розробки. У другому розділі спроєктовано структуру та архітектуру

вебзастосунку з використанням моделі даних та UML-діаграм, зокрема діаграм прецедентів, діяльності, потоків даних, послідовності, класів, станів, компонентів і розгортання. У третьому розділі описано програмну реалізацію вебзастосунку засобами Ruby on Rails та його тестування з використанням бібліотеки RSpec.

Основні результати кваліфікаційної роботи апробовано та опубліковано у збірці тез доповідей Сімнадцятої Всеукраїнської, двадцять четвертої регіональної наукової конференції молодих дослідників «Актуальні проблеми математики та інформатики» у публікації на тему «Вебзастосунок для багатоканальної комунікації на Ruby on Rails».

1 АНАЛІЗ АНАЛОГІВ, ПРЕДМЕТНОЇ ОБЛАСТІ, ВИМОГ ТА ЗАСОБІВ РОЗРОБКИ

1.1 Актуальність та аналіз систем багатоканальної комунікації

Актуальність розробки вебзастосунків багатоканальної комунікації викликана стрімким розвитком інформаційних технологій та потребою в обміні інформацією між користувачами. Сучасні месенджери використовуються не тільки для особистого спілкування, але й для організації навчання та робочих моментів. Наявність можливості створення приватних і групових чатів, а також підтримка аудіо- та відеодзвінків розширює функціональність подібних систем. Тому розробка таких застосунків є актуальною задачею, яка спрямована на створення зручного та функціонального середовища для взаємодії користувачів.

На ринку представлені найрізноманітніші сервіси багатоканальних застосунків, які мають своє призначення, унікальні можливості, функції та цільову аудиторію. Попри велику кількість готових рішень, існує потреба в розробці спеціалізованих систем, що будуть гнучкими до налаштувань, обробки медіаконтенту та забезпечуватимуть високу швидкість обміну даними у реальному часі.

Існують такі месенджери, як Telegram та Viber, які використовуються широким колом людей, та корпоративні інструменти Slack, Microsoft Teams та інші. Кожен із цих сервісів має свої переваги та недоліки. Наприклад, Telegram відомий високою швидкістю роботи та відкритим API для створення ботів, але користувач не має контролю над тим, як зберігаються дані на серверах компанії. Viber забезпечує високий рівень безпеки та якісний зв’язок, проте характеризується значним використанням пам’яті пристрою. Slack натомість пропонує зручні інструменти для командної роботи, але для зберігання історії повідомлень без обмежень часу необхідно вносити кошти. Microsoft Teams

безкоштовно інтегрується в екосистему Microsoft 365, що забезпечує ефективну командну роботу. Водночас має підвищену вимогливість до ресурсів комп’ютера, що може сповільнювати систему.

Окрему позицію серед досліджуваних систем займають платформи для організацій відеоконференцій, онлайн-зустрічей та вебінарів, зокрема Zoom і Google Meet. Дані програмні рішення забезпечують проведення дистанційних робочих зустрічей і організацію навчального процесу, проте їх функціональні можливості не дозволяють використовувати їх як повноцінні месенджери.

Розроблювана система «TolskyChat» орієнтована як на широке коло, так і на корпоративний сектор. Даний застосунок містить усі необхідні функції для зручного обміну інформації між користувачами без обмежень. А завдяки хмарному середовищу Cloudinary вирішується проблема надійного зберігання медіафайлів. Водночас на даному етапі розробки система залежить від стабільності сторонніх API-сервісів та швидкості з’єднань з ними.

1.2 Опис предметної області

Вебзастосунки багатоканальної комунікації містять багато дій і процесів, щоб забезпечити користувачів усім необхідним для ефективного обміну повідомленнями. Чати пропонують користувачам різні види комунікації: обмін текстовими, аудіо- та відеоповідомленнями, а також створення аудіо- та відеочатів.

Системою користуються користувачі, ролі яких надсилати та отримувати повідомлення, створювати нові приватні розмови або групові чати та додавати інших користувачів у список контактів.

Цілі створення системи:

– забезпечення обміну повідомленнями між користувачами;
– підтримка різних типів повідомлень(текст, аудіо, відео, фото);
– реалізація аудіодзвінків в реальному часі;

– автоматизація процесів надсилання повідомлень та системних сповіщень;
– оптимізація роботи адміністративної частини(керуванням користувачами в чаті).

Задачі, які система має вирішувати:

– реєстрація та авторизація користувача;
– додавання інших користувачів до списку контактів;
– блокування або розблокування користувача;
– створення приватної розмови, групових бесід;
– запрошення контактів до бесід;
– вилучення учасника з бесіди;
– надання учасникам групи певної ролі;
– надіслання повідомлень іншим користувачам;
– видалення повідомлення;
– здійснення аудіо- та відеодзвінків.

1.3 Вимоги до проєкту

Для створення сучасного вебсервісу для обміну повідомленнями було проведено детальний аналіз потреб користувачів та логіки взаємодії. На основі даного аналізу було сформовано перелік вимог, що забезпечують комфортне користування даним ресурсом.

Розглянемо основні **функціональні вимоги**.

Управління користувачами. Система повинна забезпечувати можливість реєстрації, автентифікації та управління обліковими записами користувачів.

У межах інформаційної системи користувач може:

– зареєструватись за допомогою електронної пошти;
– зареєструватись або увійти через обліковий запис Google;
– підтвердити адресу електронної пошти після реєстрації;

– увійти до системи після підтвердження електронної пошти;

– редагувати власні дані;

– видалити або архівувати власний обліковий запис.

Не підтверджений користувач не повинен мати можливості виконувати вхід у систему.

Управління контактами. Система повинна забезпечувати можливість роботи зі списком контактів.

Користувач повинен мати можливість:

– здійснювати пошук інших користувачів;

– надсилати запит на додавання до списку контактів;

– підтверджувати або відхиляти запит на додавання;

– переглядати список контактів;

– блокувати та розблокувати інших користувачів.

Заблокований користувач не може:

– створювати нові бесіди з користувачем, який його заблокував;

– запрошувати до групових бесід.

Управління бесідами. Система повинна підтримувати різні типи бесід:

– приватні діалоги між двома користувачами;

– групові бесіди;

– публічні бесіди, до яких може приєднатись будь-який користувач;

– приватні групові бесіди, до яких користувачів запрошують учасники.

Користувач повинен мати можливість:

– створювати бесіди;

– видалити або архівувати власні бесіди;

– запрошувати користувачів до приватних бесід;

– виключати користувачів із бесіди;

– вимикати або вмикати сповіщення для окремих бесід.

Ролі користувачів у бесідах. У групових бесідах повинні підтримуватись ролі користувачів: власник бесіди, модератор, учасник.

Власник бесіди:

– може призначати адміністраторів;

– може запрошувати та виключати користувачів;

– змінювати назву бесіди та її приватність.

Модератор:

– може запрошувати користувачів;

– може виключати користувачів;

– змінювати назву бесіди.

Учасник:

– може запрошувати користувачів.

Обмін повідомленнями. Система повинна забезпечувати можливість обміну повідомленнями в режимі реального часу.

Користувач повинен мати можливість:

– надсилати повідомлення;

– видаляти власні повідомлення;

– використовувати емодзі;

– додавати вкладення (зображення та інші файли);

– створювати аудіо- та відеодзвінки.

Повідомлення, які знаходяться у видимій області бесіди, повинні автоматично позначатися як прочитані.

Система сповіщень. Система повинна створювати сповіщення у таких випадках:

– отримано запрошення до бесіди;

– користувач покинув бесіду;

– користувач призначений адміністратором;

– користувача виключено з бесіди;

– користувач має непрочитані повідомлення понад 30 хвилин.

Якщо користувач не прочитав сповіщення протягом 1 години, система повинна надіслати повідомлення на електронну пошту.

Управління архівом. Архів бесід повинен автоматично видалятися через 3 місяці.

Окрім функціональних вимог, до системи висуваються й **нефункціональні вимоги**, розглянемо їх детальніше.

Робота в реальному часі. Система має забезпечувати обмін даними в режимі реального часу без перезавантаження сторінки з використанням WebSocket-з'єднання.

Безпека. Система повинна забезпечувати автентифікацію, авторизацію та розмежування прав доступу користувачів.

1.4 Опис програмних засобів

Для розробки вебзастосунку багатоканальної комунікації були обрані сучасні програмні засоби, за допомогою яких, можна досягти поставленої мети, відповідно до вимог системи.

Основним засобом серверної розробки обрано фреймворк Ruby on Rails, який реалізує архітектурний патерн MVC та надає вбудовані механізми для роботи з базами даних, маршрутизації, обробки HTTP-запитів і побудови вебінтерфейсів. Використання даного фреймворку дозволяє прискорити процес розробки завдяки великої наявності вбудованим компонентам та готових програмних рішень [1].

Для зберігання даних використано систему керування базами даних PostgreSQL, яка забезпечує безпечне зберігання структурованих даних, підтримує складні запити та масштабування дуже складних робочих навантажень з даними [2].

У реалізації користувацького інтерфейсу велику роль відіграє бібліотека Bootstrap, що містить готові компоненти інтерфейсу та дозволяє створювати адаптивний дизайн для різних типів пристроїв.

Обмін повідомленнями в режимі реального часу реалізовано з використанням бібліотеки Action Cable, яка забезпечує двосторонню взаємодію між клієнтом і сервером без перезавантаження сторінки. Тобто дана бібліотека інтегрує в додаток WebSocket [3].

Для реалізації механізмів реєстрації, автентифікації, авторизації користувачів використано бібліотеку Devise [4].

Аудіо- та відеозв’язок реалізовано з використанням технологій WebRTC, яка забезпечує передачу відео та голосу між користувачами в реальному часі. MediaRecorder використано для запису голосових повідомлень[5, 6].

Для зберігання та обробки мультимедійного контенту використано хмарний сервіс CLoudinary, що забезпечує швидкий доступ до медіафайлів.

Для організації аудіо- та відеозв’язку за межами одної мережі використано сервіс Twilio, який забезпечує стабільне з’єднання між користувачами[7].

Також було використано бібліотеку Sidekiq для виконання фонових задач, зокрема надсилання сповіщень та автоматичної обробки системних подій [8].

1.5 Висновки до розділу 1

У першому розділі обґрунтовано актуальність розробки вебзастосунків багатоканальної комунікації та проведено аналіз предметної області на основі дослідження існуючих систем. Також сформульовано вимоги до системи та описано вибір програмних засобів.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Побудова концепції вебзастосунку

Концептуальне проєктування полягає в уточненні розуміння й узгодження деталей вимог. Тому він є невід'ємним етапом розробки програмного забезпечення. На даному етапі можна зрозуміти загальну ідею роботи системи ще до її реалізації та сформулювати основні вимоги користувачів без заглиблення в технічні деталі.

Для представлення взаємодії користувачів із системою та визначення основних функціональних можливостей вебзастосунку було побудовано діаграму прецедентів (див. рис. 2.1). Дана діаграма дозволяє відобразити основні сценарії роботи системи, ролі користувачів та доступні їм функції.

Основним актором системи є користувач, який може здійснювати реєстрацію та авторизацію, керувати власним обліковим записом, створювати бесіди, надсилати повідомлення та взаємодіяти з іншими користувачами системи додавши їх у список контактів. Актори «Модератор» та «Власник бесіди» є розширенням ролі користувача, оскільки мають такі ж самі базові функціональні можливості та додатково мають права на управління бесідою. Зокрема, власник бесіди може запрошувати, виключати користувачів, призначати модератора та змінювати певні налаштування бесіди, а модератор – лише запрошувати та виключати користувачів.

На діаграмі відображено взаємозв'язки між окремими прецедентами, зокрема реєстрації, управління власним обліковим записом, списком контактів та обміном повідомленнями.

На рисунку 2.2 наведено діаграму потоків даних, що відображає процес передачі та обробки інформації в системі TolskyChat. На діаграмі показано взаємодію користувача із системою, основні функціональні процеси, зовнішні сервіси та сховища даних.

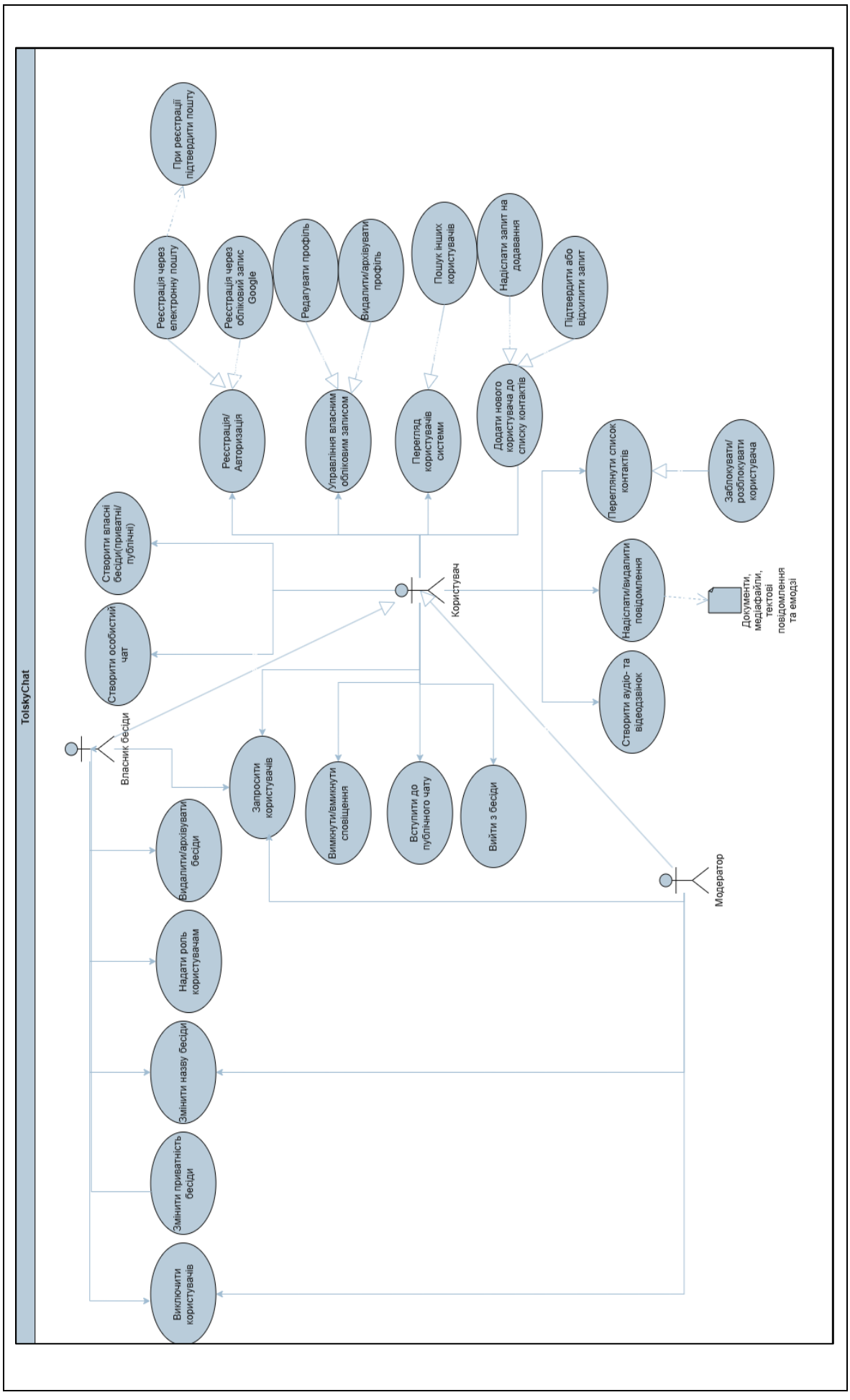


Рисунок 2.1 – Діаграма прецедентів

Користувач передає до системи дані для реєстрації, авторизації, управління профілем, контактами, бесідами, повідомленнями та дзвінками. Для авторизації через обліковий запис Google система взаємодіє із сервісом Google OAuth. Процес аудіо- та відеодзвінків реалізується за допомогою технологій WebRTC та сервісу Twilio, який використовується для обробки сигналів даних під час встановлення з’єднання між користувачами.

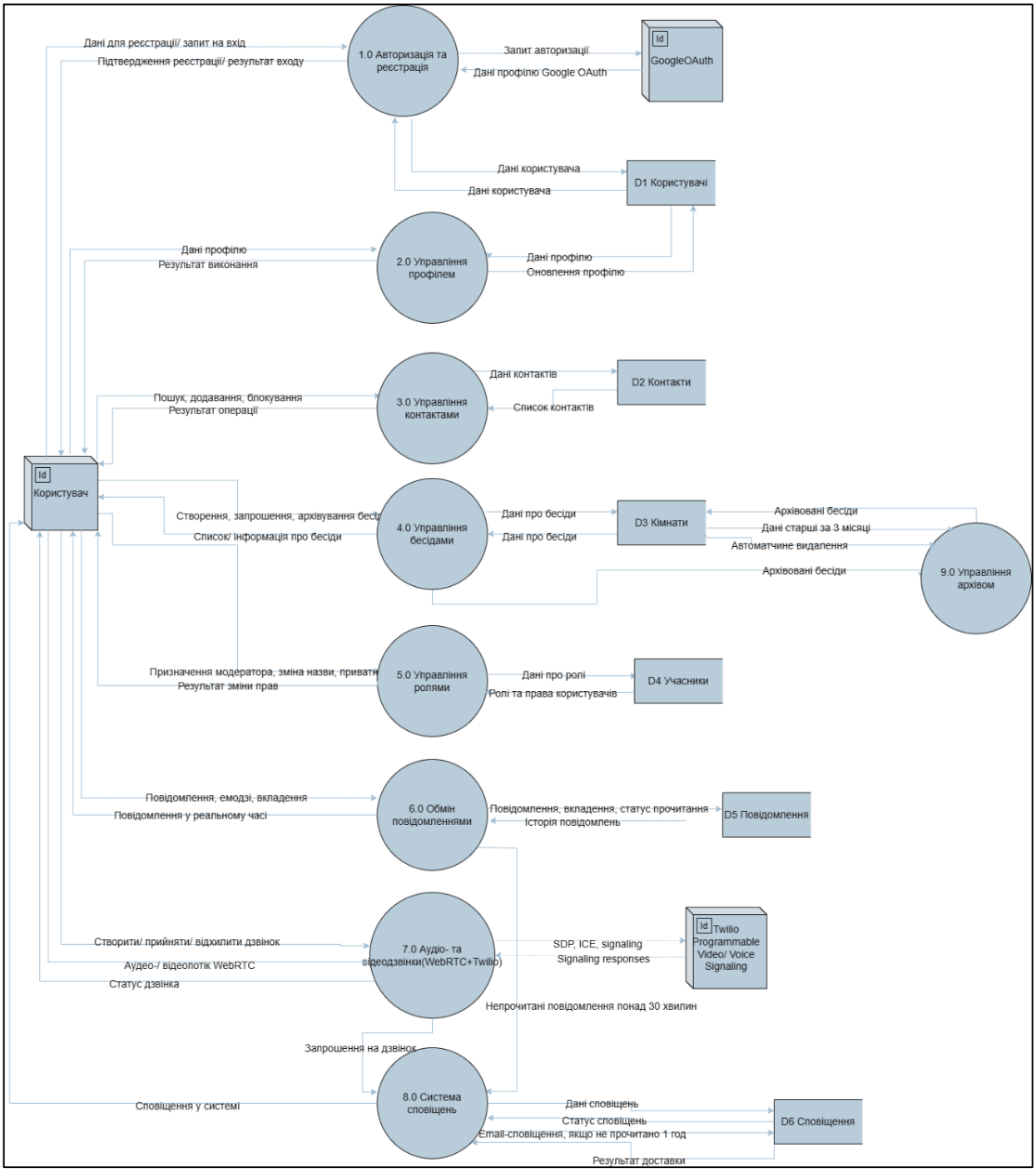


Рисунок 2.2 – Діаграма потоків даних

Для збереження інформації використовуються окремі сховища даних, зокрема сховища користувачів, контактів, кімнат, повідомлень, учасників кімнат та сповіщень.

Також одним із важливих етапів концептуального проєктування є побудова діаграми діяльності. Оскільки вона допомагає виявити основні потоки виконання процесів, спростити аналіз функціональності застосунку та забезпечити більш ефективну реалізацію програмного забезпечення.

На рисунку 2.3 наведено діаграму діяльності, яка відображає послідовність взаємодії користувача із системою TolskyChat та описує основні сценарії роботи застосунку. За допомогою даної діаграми можна простежити логіку виконання дій користувача, реакцію системи на ці дії, а також особливості роботи різних ролей у системі.

На рисунку представлено процес авторизації та реєстрації користувача різними способами, а саме через електронну пошту, що передбачає проходження верифікації, або через обліковий запис Google, який дозволяє виконати вхід до системи без додаткового підтвердження електронної пошти. Також відображено основні функції користувача, зокрема перегляд повідомлень, управління контактами, створення бесід, вступ до публічних чатів та налаштування сповіщень.

Окремо на діаграмі показано додаткові можливості власника бесіди та модератора. Власник бесіди може призначити модераторів, видаляти бесіди та керувати учасниками чату. У той час як модератор, має можливість лише запрошувати та виключати учасників з бесіди.

Діаграма демонструє розподіл відповідальності між користувачем, системою, власником бесіди та модератором за допомогою функціональних доріжок. Це дозволяє наочно показати, які саме дії виконує кожний учасник процесу та які операції обробляються безпосередньо системою.

На діаграмі відображено альтернативні сценарії розвитку подій, які виникають залежно від вибору способу входу до системи, ролі користувача або результатів перевірки введених даних. Це дозволяє врахувати різні варіанти роботи системи ще на етапі проєктування.

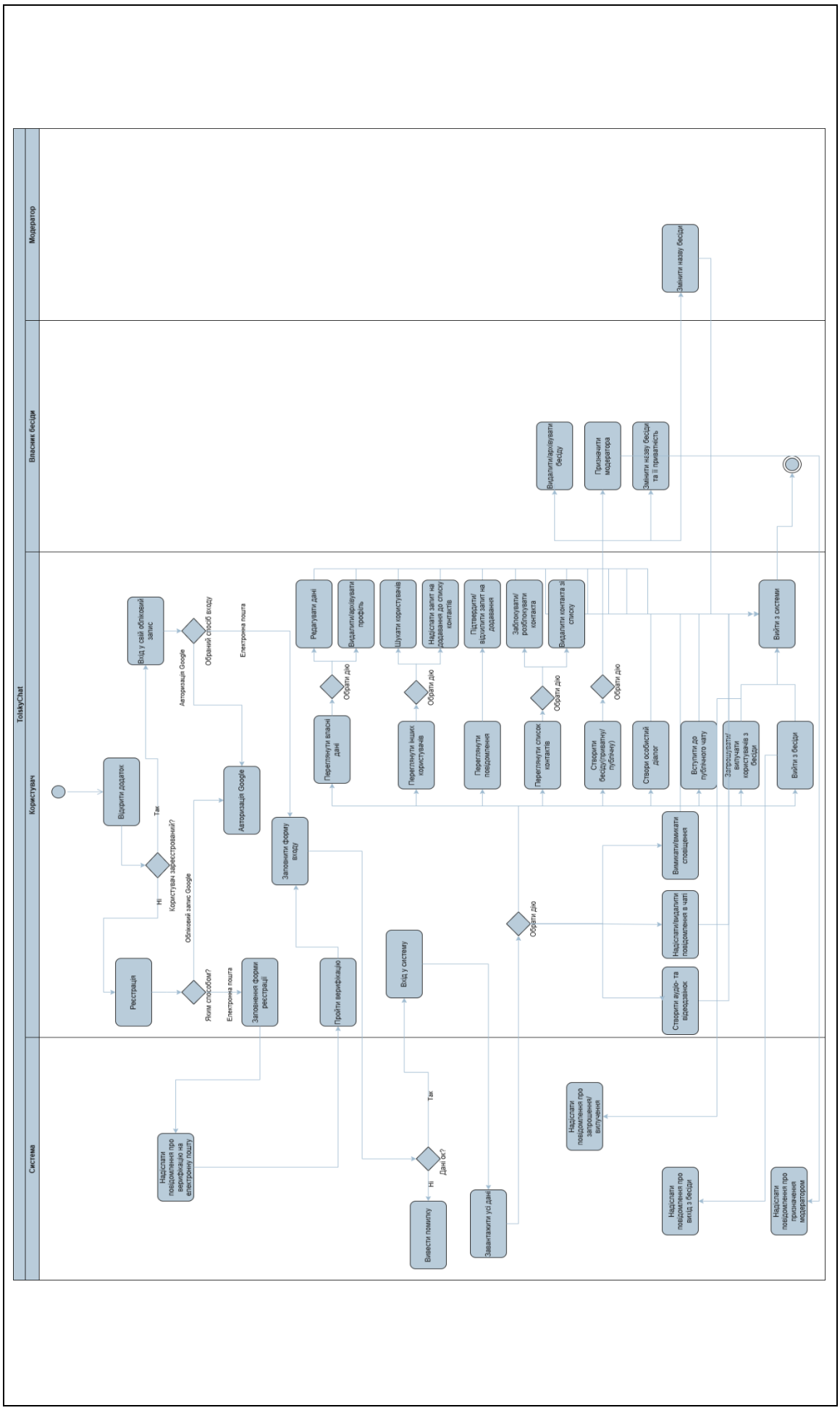


Рисунок 2.3 – Діаграма діяльності

2.2 Логічне моделювання системи

Для того щоб відобразити структуру бази даних інформаційної системи TolskyChat, було спроєктовано ER-діаграму (див. рис. 2.4). Дана діаграма описує основні сутності системи, їх атрибути та зв’язки між ними.

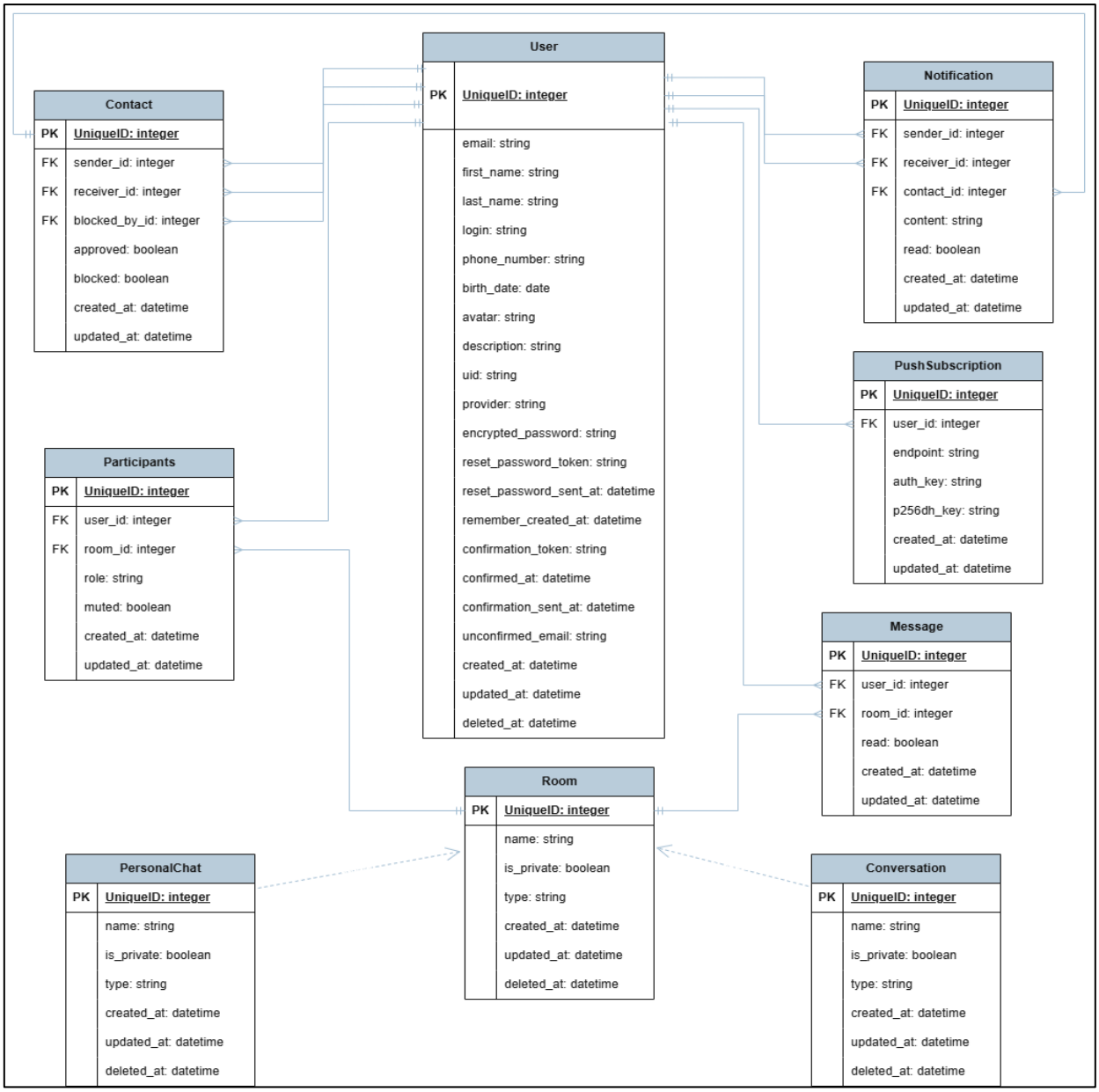


Рисунок 2.4 – ER-діаграма

Центральною сутність є таблиця User, яка містить основну інформацію про користувачів системи, параметри авторизації та підтвердження облікового запису.

Таблиця Contact використовується для реалізації списку контактів та зберігання інформації про взаємозв’язки між користувачами. Зв’язок між користувачами та бесідами реалізовано через таблицю Participants, яка також містить роль користувача в бесіді.

Сутність Room є базовою моделлю бесіди, від якої наслідуються Conversation та PersonalChat для групових і особистих чатів відповідно.

Таблиця Message забезпечує зберігання повідомлень для статусів їх прочитання, а Notification – системних сповіщень. Для підтримки push-сповіщень використовується таблиця PushSubscription.

На рисунку 2.5 наведено діаграму класів, яка відображає архітектуру інформаційної системи TolskyChat, основні класи застосунку та зв’язки між ними. На діаграмі представлено моделі даних, контролери, сервіси та WebSocket-канали, що забезпечують роботу системи обміну повідомлення в режимі реального часу.

Центральними сутностями є класи User, Room, Message, Participant, Contact та Notification, які відповідають за управління користувачами, бесідами, повідомленнями, контактами та сповіщеннями. Контролери реалізують обробку HTTP-запитів, сервіси містять бізнес-логіку системи, а канали Action Cable забезпечують обмін даними в реальному часі, зокрема для чатів, дзвінків та сповіщень.

На рисунку 2.6 наведено діаграму послідовностей, що відображає процес автентифікації користувача через Google OAuth. Користувач ініціює вхід через сторінку авторизації, після чого система перенаправляє його до сервісу Google OAuth для вибору облікового запису та підтвердження входу.

Після отримання OAuth-даних система перевіряє наявність користувача в базі даних PostgreSQL. Якщо користувач уже існує, виконується створення сесії, а якщо ні – створюється новий обліковий запис і після цього відкривається сесія користувача.

Після успішно відкритої сесії користувача перенаправляють на головну сторінку застосунку.



Рисунок 2.5 – Діаграма класів

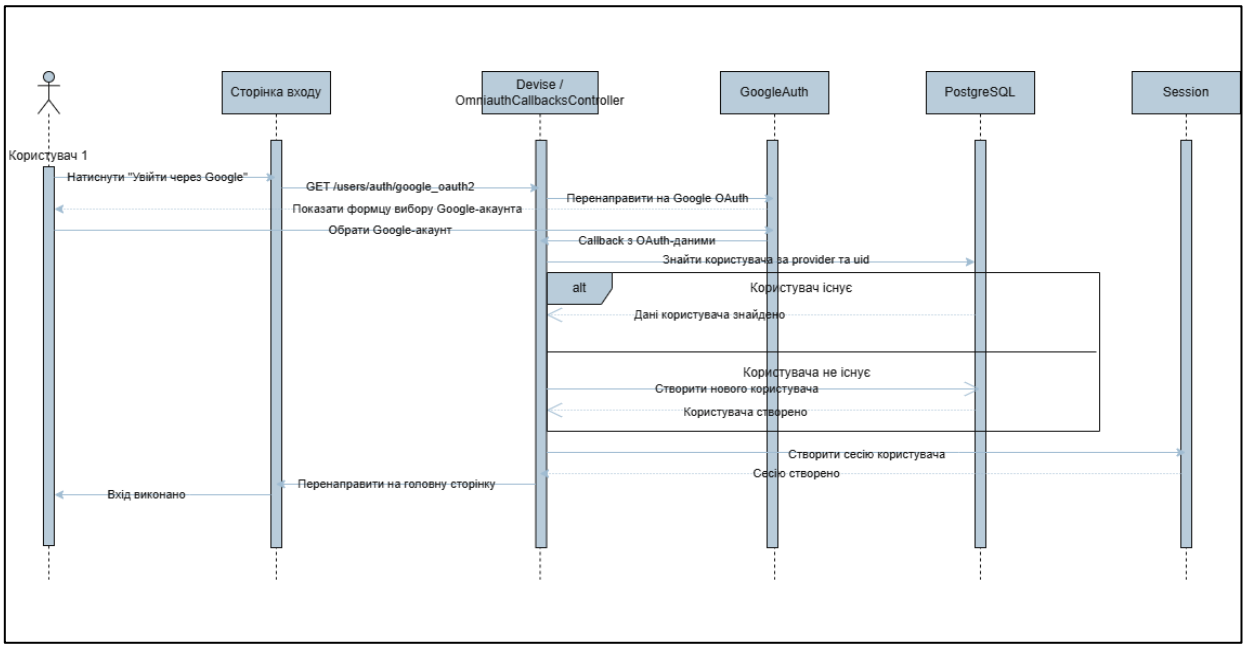


Рисунок 2.6 – Діаграма послідовностей прецеденту «Увійти через обліковий запис Google»

На рисунку 2.7 наведено діаграму послідовностей, яка описує процес надсилання повідомлення в чаті в режимі реального часу. Користувач вводить повідомлення на сторінці чату та надсилає його до MessagesController через HTTP-запит. Контролер зберігає повідомлення в базі даних PostgreSQL, після чого передає його через WebSocket-з’єднання ActionCable. У результаті інший користувач миттєво отримує нове повідомлення без перезавантаження сторінки.

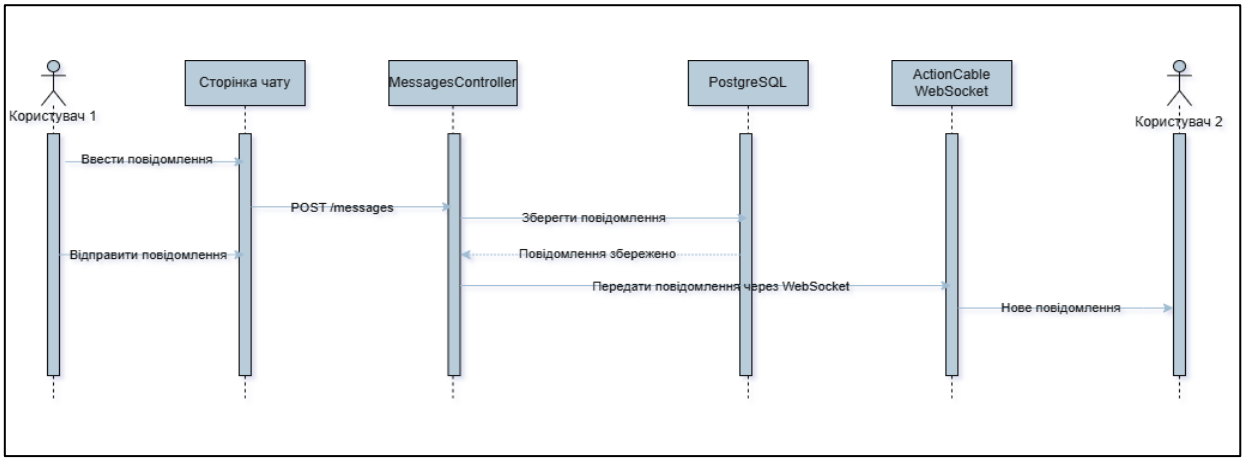


Рисунок 2.7 – Діаграма послідовностей прецеденту «Надіслати повідомлення»

На рисунку 2.8 наведено діаграму послідовностей прецеденту для створення аудіо- та відеодзвінка. Дана діаграма відображає процес встановлення аудіо- або відеодзвінка між двома користувачами за допомогою WebRTC.

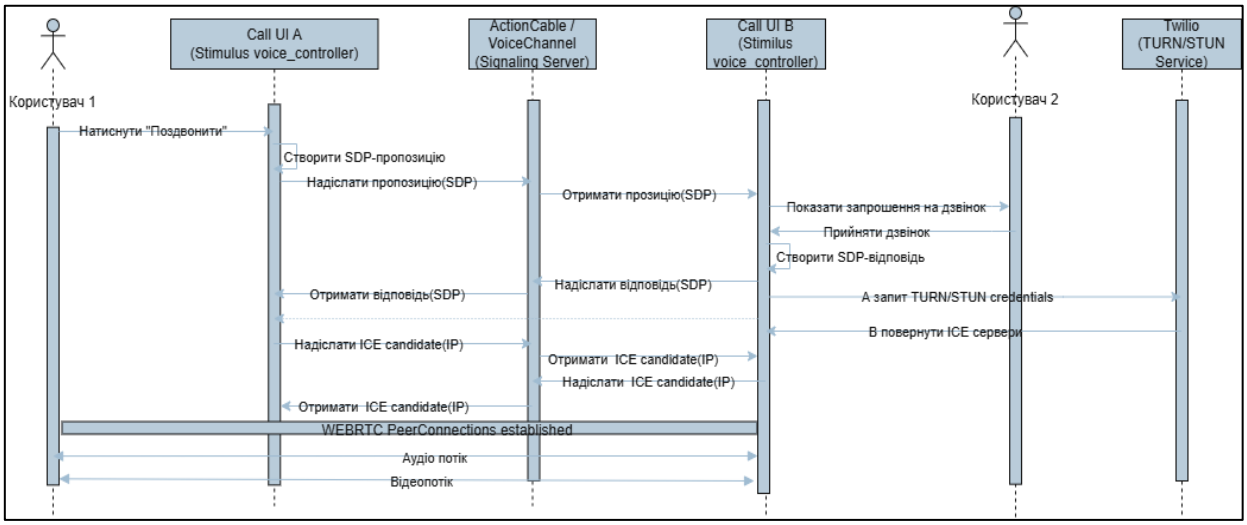


Рисунок 2.8 – Діаграма послідовностей прецеденту «Створити дзвінок»

Користувач 1 ініціює дзвінок, після чого через Action Cable та VoiceChannel передається SDP-пропозиція іншому користувачу.

Користувач 2 приймає виклик, створює SDP-відповідь та надсилає її назад. Далі сторони обмінюються ICE-кандидатами та отримують облікові дані TURN/STUN для встановлення прямого P2P-з’єднання.

Після успішного встановлення WebRTC-з’єднання між користувачами починається передача аудіо- та відеопотоку в режимі реального часу. Для забезпечення стабільного з’єднання система використовує STUN/TURN-сервери, які допомагають встановити зв’язок між користувачами навіть у випадку мережевих обмежень або NAT.

Для відображення життєвого циклу основних сутностей інформаційної системи TolskyChat та можливих переходів між їхніми станами було побудовано діаграму станів (див. рис. 2.9). На діаграмі представлено процеси управління користувачами, контактами, повідомленнями, дзвінками, сповіщеннями та бесідами.

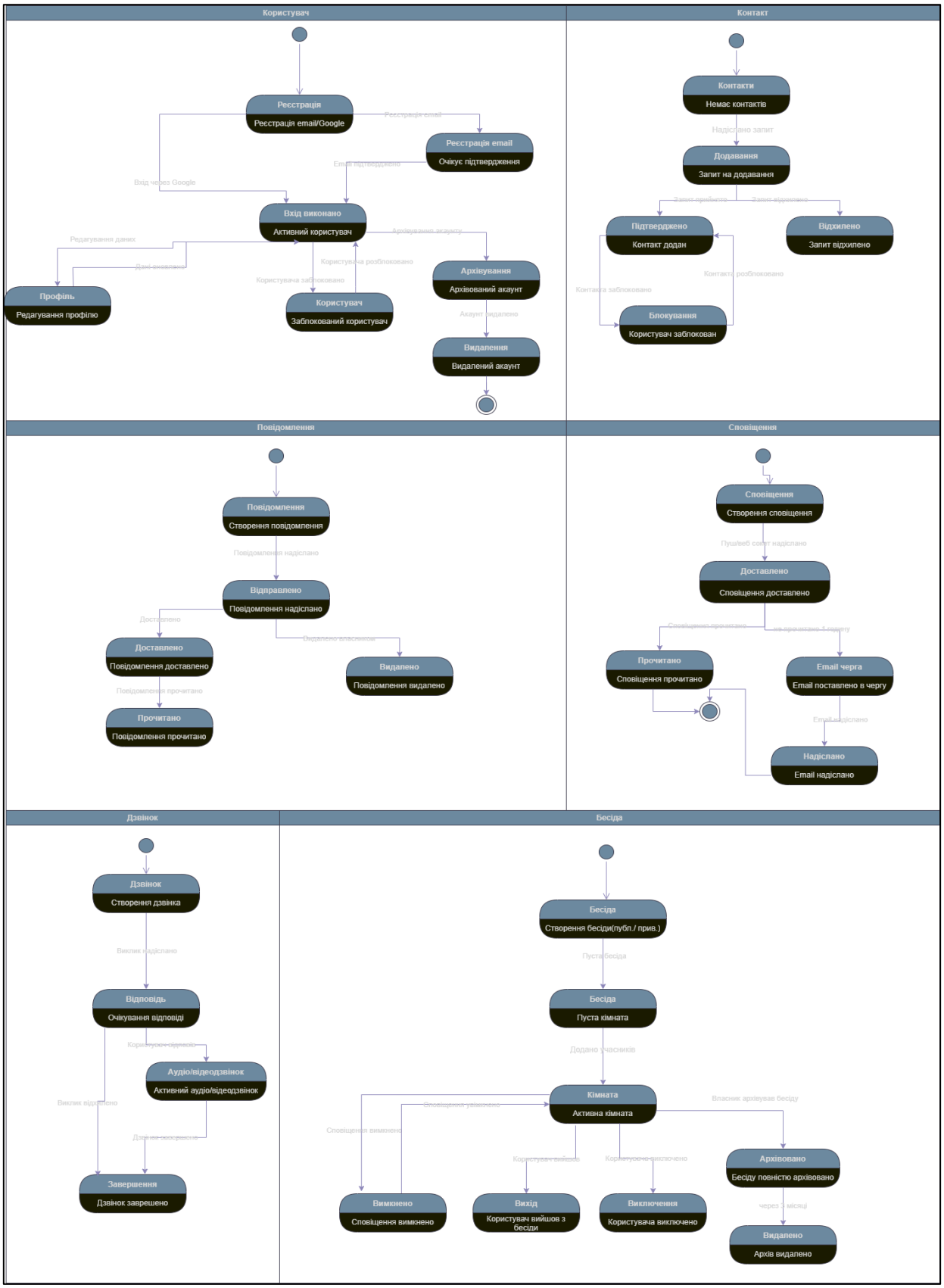


Рисунок 2.9 – Діаграма станів

Для користувача показано процес реєстрації, підтвердження електронної пошти, редагування профілю, блокування та архівування облікового запису.

Для контактів відображено життєвий цикл запиту на додавання, його підтвердження, відхилення та блокування користувача.

У частині повідомлень продемонстровано етапи створення, надсилання, доставлення, прочитання та видалення повідомлення. Також показано процес встановлення аудіо- та відеодзвінка, який включає створення виклику, очікування відповіді та активне з'єднання дзвінка, після чого його завершення.

Окремо відображено роботу системи сповіщень, зокрема створення сповіщення, його доставлення, прочитання та надсилання email-сповіщення у випадку тривалої відсутності перегляду. Для бесід показано процес створення кімнати, переходу до активного стану, вимкнення сповіщень, виходу або виключення користувача з бесіди, а також архівування та автоматичне видалення архіву.

2.3 Проєктування архітектури та компонентів системи

Фізичне проєктування є важливим етапом розробки системи, оскільки дозволяє визначити структуру програмних компонентів, способи їх взаємодії та необхідні зовнішні сервіси для забезпечення стабільної роботи застосунку. На цьому етапі формується загальна архітектура системи та визначаються технології, що використовуються для реалізації її функціональних можливостей.

На рисунку 2.10 наведено діаграму компонентів, що відображає структуру програмної системи TolskyChat та взаємозв'язки між її основними модулями. На діаграмі представлено клієнтську частину застосунку, серверну частину на Ruby on Rails, бази даних та зовнішні сервіси. Клієнтська частина включає Web-інтерфейс, Stimulus controller та модуль WebRTC для реалізації аудіо- та відеодзвінків. Серверна частина містить контролери, моделі, сервіси, Action Cable канали, механізми автентифікації Devise/Google OAuth, а також Active Storage та Sidekiq для обробки файлів і фонових задач. Для зберігання даних використовується PostgreSQL, а Redis забезпечує роботу WebSockets-

з’єднань та фонових процесів. Окремо показано взаємодію системи із зовнішніми сервісами, зокрема Google OAuth, TURN/STUN server, Cloudinary та Web Push.

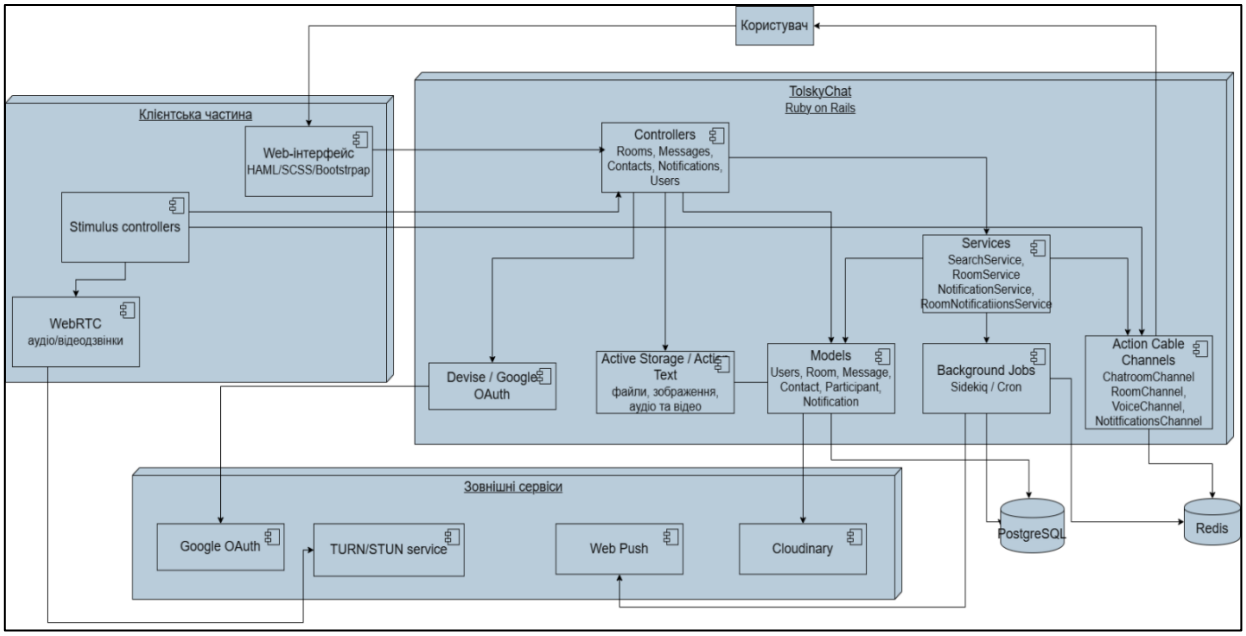


Рисунок 2.10 – Діаграма компонентів

На рисунку 2.11 наведено діаграму розгортання, яка відображає фізичну архітектуру інформаційної системи TolskyChat та взаємодію її основних компонентів у середовищі виконання.

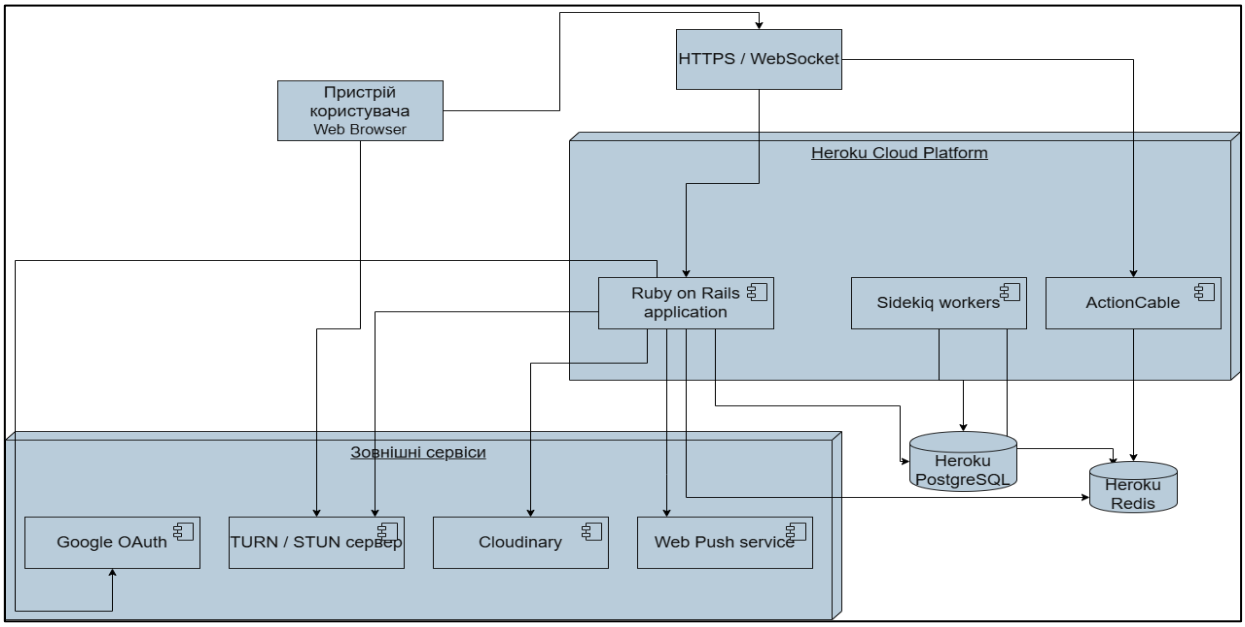


Рисунок 2.11 – Діаграма розгортання

На діаграмі показано клієнтський пристрій користувача, серверну частину застосунку, бази даних та зовнішні сервіси, що забезпечують роботу системи.

Користувач взаємодіє із застосунком через Web Browser, використовуючи HTTPS та WebSocket-з’єднання. Серверна частина системи розгорнута на платформі Heroku та включає Ruby on Rails вебзастосунок, Action Cable для роботи в режимі реального часу та Sidekiq workers для виконання фонових задач. Для зберігання даних використовується Heroku PostgreSQL, а Heroku Redis забезпечує роботу WebSocket-з’єднань та черг фонових процесів.

Окремо на діаграмі представлено Google OAuth для автентифікації користувачів, TURN/STUN сервер для підтримки WebRTC-з’єднань, Cloudinary для зберігання медіафайлів та Web Push service для надсилання пуш-сповіщень.

2.4 Висновки до розділу 2

У другому розділі було здійснено концептуальне, логічне та фізичне проєктування інформаційної системи TolskyChat. У процесі проєктування було визначено основні функціональні можливості системи, структуру бази даних, взаємодію між компонентами застосунку та особливості розгортання програмного забезпечення.

Результати проєктування стали основою для подальшої реалізації вебзастосунку та визначили архітектурні рішення, необхідні для забезпечення роботи системи в режимі реального часу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація вебзастосунку

Програмна реалізація інформаційної системи TolskyChat виконана з використанням фреймворку Ruby on Rails. Застосунок побудований відповідно до архітектурного шаблону MVC, що дозволяє розділити логіку роботи системи на моделі, контролери та представлення.

Для зберігання даних використовується PostgreSQL, а для реалізації обміну даними в режимі реального часу застосовується Action Cable та WebSocket-з'єднання. У проєкті також використовується Devise для автентифікації користувачів, Google OAuth для входу через обліковий запис Google, Active Storage та Action Text для роботи з файлами й повідомленнями, Redis, Sidekiq, Cloudinary та Web Push. Це підтверджується залежностями, зазначеними у Gemfile проєкту[9].

На рисунку 3.1 наведено фрагмент реалізації моделі User, яка відповідає за керування користувачами.

```
class User < ApplicationRecord
  has_many: participants, dependent: :destroy
  has_many: messages, dependent: : destroy
  has_many: push_subscriptions, dependent: : destroy
  validates:login, presence: true, uniqueness: true
  mount_uploader: avatar, AvatarUploader
  devise: database_authenticatable, : registerable,
        : recoverable, : rememberable, :validatable,
        : confirmable, : omniauthable,
        omniauth_providers: [: google_oauth2]
end
```

Рисунок 3.1 – Фрагмент реалізації моделі користувача

Основна структура вебзастосунку складається з моделей користувачів, бесід, повідомлень, контактів, учасників бесід та сповіщень. Модель User відповідає за збереження даних користувача, його автентифікацію, підтвердження електронної пошти, фото профілю та зв’язки з іншими сутностями системи.

У моделі використовується Devise з необхідними модулями, що забезпечують стандартну реєстрацію, вхід, відновлення пароля, підтверджений email та авторизацію через Google OAuth.

На рисунку 3.2 наведено фрагмент реалізації входу через Google OAuth. У моделі користувача створено метод from_omniauth, який перевіряє, чи існує користувач із відповідним провайдером або електронною поштою. Якщо користувач існує, система оновлює його OAuth-дані, а якщо ні – створює новий обліковий запис.

```
def from_omniauth(auth)
    user = find_by(provider: auth.provider, uid: auth.uid) || find_by(email: auth.info.email)
    if user
      user.provider = auth.provider if user.provider.blank?
      user.uid = auth.uid if user.uid.blank?
    else
      user = new
      user.provider = auth.provider
      user.uid = auth.uid
      user.email = auth.info.email
      user.password = Devise.friendly_tokenp[0, 20]
      user.login = auth.info.email.split(‘@’).first
    end
    user.skip_confirmation!
    user.save!
    user
  end
```

Рисунок 3.2 – Метод для реалізації входу через Google OAuth

Для визначення ролей користувачів у групових бесідах використовується модель Participant. Вона забезпечує зв’язок між користувачем і бесідою, а також зберігає інформацію про роль учасника в конкретному чаті. Такий підхід дозволяє реалізувати механізм розмежування прав доступу в межах групових бесід.

У системі передбачено три ролі: звичайний учасник, власник та модератор. Кожна з ролей має власний набір дозволених дій, зокрема керування учасниками бесіди, запрошення користувачів та редагування параметрів чату (див. рис. 3.3).

```
class Participant < ApplicationRecord
  belongs_to :user

  belongs_to :room

  enum :role, {
    member: 'Member',
    owner: 'Owner',
    moderator: 'Moderator'
  }

end
```

Рисунок 3.3 – Модель для учасників кімнат

На рисунку 3.4 наведено фрагмент надсилання повідомлень, який реалізовано у контролері для повідомлень. Після його створення система зберігає його в базі даних, створює сповіщення для інших учасників бесіди та транслює нове повідомлення через канал Action Cable. Завдяки цьому повідомлення з’являються у чаті в режимі реального часу без перезавантаження сторінки.

```
def create
    @message = @room.messages.new(message_params.merge(user: current_user))
    @message.read = recipients_viewing_room_now?
```

```
    return head :unprocessable_entity unless @message.save
    create_nofitication
    broadcast_message
    head :no_content
  end

private
def broadcast_message
    ChatroomChannel.broadcast_to(
      @room,
      sender_id: current_user.id,
      html: render_to_string(
        partial: 'messages/message',
        formats: [:html],
        locals: { message: @message }
      )
    )
  end
```

Рисунок 3.4 – Фрагмент створення та надсилання повідомлення

На рисунку 3.5 наведено реалізацію VoiceChannel. Для реалізації аудіо- та відеодзвінків у системі використовується технологія WebRTC, яка забезпечує встановлення прямого P2P-з’єднання між користувачами. Для передачі сигнальних повідомлень використовується VoiceChannel, який працює на основі Action Cable та WebSocket-з’єднань. Канал отримує сигнальні дані від одного користувача та пересилає їх іншому учаснику дзвінка. Це дозволяє обмінюватися SDP-пропозиціями, SDP-відповідями та ICE-кандидатами, необхідними для встановлення та підтримки WebRTC-з’єднань. Такий підхід забезпечує передачу аудіо- та відеопотоку між користувачами в режимі реального часу без необхідності постійної обробки медіаданих сервером.

```
class VoiceChannel < ApplicationCable::Channel
  def subscribed
    stream_for current_user
  end

  def signal(data)
    receiver = User.find_by(id: data['receiver_id'])
    return unless receiver

    VoiceChannel.broadcast_to(receiver, data)
  end
end
```

Рисунок 3.5 – Реалізація VoiceChannel

На клієнтській стороні логіка дзвінків реалізована у Stimulus-контролері voice_controller (див. рис. 3.6). У ньому створюється об’єкт RTCPeerConnection, обробляються локальні та віддалені медіапотоки, виконується обмін ICE-кандидатами, а також реалізовано керування мікрофоном, камерою, таймером дзвінка та відновленням стану після оновлення сторінки.

Крім цього, контролер відповідає за обробку вхідних викликів, зміну стану інтерфейсу під час дзвінка та взаємодію з VoiceChannel для передачі сигнальних повідомлень між користувачами.

На рисунку 3.7 наведено контролер TurnCredentialsController який використовується для отримання TURN/STUN-даних. Він виконує запит до сервісу Twilio, отримує список ICE-серверів і повертає їх клієнтській частині у форматі JSON. Це потрібно для стабільнішого встановлення WebRTC-з’єднання, особливо коли користувачі знаходяться за NAT або мають мережеві обмеження.

```
async initPeer() {
   const iceServers = await this.resolveIceServers()

   this.peer = new RTCPeerConnection({
    iceServers
   })

   this.peer.onicecandidate = (e) => {
    if (e.candidate) {
     this.channel?.perform('signal', {
       receiver_id: this.activePeerId,
       candidate: e.candidate,
       caller_id: this.currentUserIdValue
      })
    }
   }

   this.peer.ontrack = (e) => {
    this.setStatus(“Connected”)
    this.showModal(“activeCallModal)
    }
}
```

Рисунок 3.6 – Фрагмент реалізації дзвінка на клієнтській стороні

```
class TurnCredentialsController < ApplicationController
 before_action :authenticate_user!

 def show
   sid = ENV['TWILIO_ACCOUNT_SID']
   token = ENV['TWILIO_AUTH_TOKEN']

   uri = URI("https://api.twilio.com/2010-04-01/Accounts/#{sid}/Tokens.json")
   req = Net::HTTP::Post.new(uri)
   req.basic_auth(sid, token)

   res = Net::HTTP.start(uri.host, uri.port, use_ssl: true) do |http|
     http.request(req)
   end

   body = JSON.parse(res.body)
   render json: { ice_servers: body["ice_servers"] }
end
```

Рисунок 3.7 – Контролер TurnCredentialsController

3.2 Тестування вебзастосунку

Для перевірки коректності роботи багатоканального вебзастосунку TolskyChat було проведено тестування основних функціональностей системи. У проєкті використовується фреймворк RSpec, а тестові файли розміщені в каталозі spec. Структура тестів охоплює перевірку моделей, HTTP-запитів, сервісів, WebSocket-каналів та фонових задач Sidekiq.

Основну увагу було приділено тестуванню тих частин системи, які відповідають за авторизацію користувачів, роботу з повідомленнями, контактами, бесідами, сповіщеннями та дзвінками. Для моделей перевірялися валідації, зв’язки між сутностями та окремі методи бізнес-логіки.

На рисунку 3.8 наведено фрагмент тестування моделі User, в якому перевіряється створення підтвердженого користувача через Google OAuth, а також пошук користувачів за логіном. На рисунку 3.9 наведено фрагмент, на якому окремо було протестовано роботу повідомлень.

```
it 'creates a confirmed user from auth data' do
    expect(from_omniauth).to be_confirmed
end
it 'returns users matching login' do
    expect(search.pluck(:login)).to contain_exactly('alice')
end
```

Рисунок 3.8 – Фрагмент тестування входу через Google OAuth

```
it 'creates a message' do
    expect { create_message }.to change(Message, :count).by(1)
end
it 'forbids deleting another user message' do
    destroy_message
    expect(response).to have_http_status(:forbidden)
end
```

Рисунок 3.9 – Фрагмент тестування роботи повідомлень

У request-тестах перевіряється створення нового повідомлення в бесіді, а також обмеження доступу до видалення чужих повідомлень. Це дозволяє переконатися, що користувач може створювати власні повідомлення, але не має права видаляти повідомлення інших учасників.

На рисунку 3.10 наведено фрагмент тестування для перевірки роботи WebSocket-каналу для аудіо- та відеодзвінків. У тесті VoiceChannel перевіряється, що сигнальні дані WebRTC передаються потрібному отримувачу. Це важливо для коректної роботи механізму встановлення дзвінка між двома користувачами.

```
it 'broadcasts signal data to the receiver' do
    signal
    expect(described_class).to                    have_received(:broadcast_to).with(receiver,
hash_including('type' => 'offer'))
end
```

Рисунок 3.10 – Фрагмент тестування VoiceChannel

На рисунку 3.11 наведено фрагмент тестування фонових задач Sidekiq. Наприклад, задача ArchiveCleanupJob перевіряє автоматичне видалення бесід, які були архівовані більше ніж три місяці тому. У тесті створюються дві бесіди: стара архівована бесіда та нещодавно архівована бесіда. Після виконання задачі видаляється лише та бесіда, яка перевищує встановлений термін зберігання.

```
it 'removes rooms archived for more than three months' do
    expect { perform }.to change(Room, :count).by(-1)
end
```

Рисунок 3.11 – Фрагмент тестування Sidekiq-задачі

У результаті виконання тестування всі перевірки були успішно пройдені. Отримані результати підтвердили коректність роботи основних

функціональних можливостей вебзастосунку, зокрема системи автентифікації, обміну повідомленнями, WebSocket-з’єднань, аудіо- та відеодзвінків, а також фонових процесів (див. рис. 3.12).

```
Top 4 slowest example groups:
  Messages
    0.6313 seconds average (1.26 seconds / 2 examples) ./spec/requests/messages_spec.rb:3
  VoiceChannel
    0.08474 seconds average (0.08474 seconds / 1 example) ./spec/channels/voice_channel_spec.rb:3
  User
    0.02185 seconds average (0.0437 seconds / 2 examples) ./spec/models/user_spec.rb:3
  ArchiveCleanupJob
    0.01824 seconds average (0.01824 seconds / 1 example) ./spec/sidekiq/archive_cleanup_job_spec.rb:3

Finished in 1.41 seconds (files took 1.79 seconds to load)
6 examples, 0 failures

Randomized with seed 63532
```

Рисунок 3.12 – Результати тестування наведених прикладів

3.3 Загальний огляд системи

Після проведеної роботи застосунок було завершено, тому нижче наведено загальний огляд інтерфейсу вебзастосунку та приклади роботи основних функціональних можливостей системи. Представлені рисунки демонструють реалізований функціонал застосунку та особливості взаємодії користувача із системою. На рисунку 3.13 наведено сторінку автентифікації.

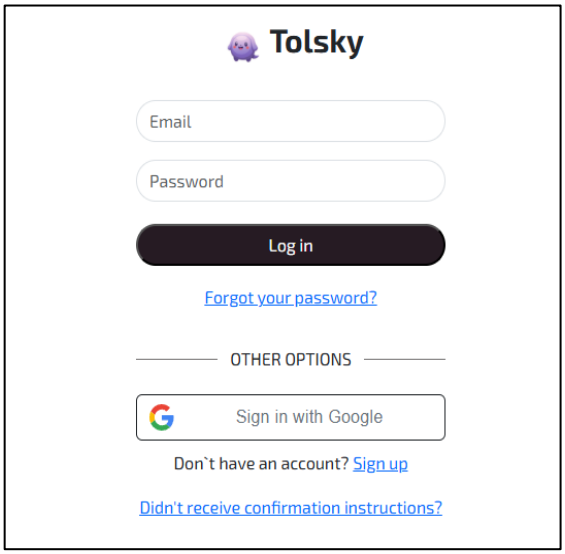


Рисунок 3.13 – Сторінка автентифікації

Користувач може виконати вхід до системи, зареєструвати новий обліковий запис, відновити пароль або скористатися авторизацією через Google OAuth. Інтерфейс сторінки забезпечує зручну взаємодію користувача із системою та надає доступ до основних механізмів автентифікації вебзастосунку.

На рисунку 3.14 продемонстровано сторінку Google OAuth, де користувач може обрати обліковий запис.

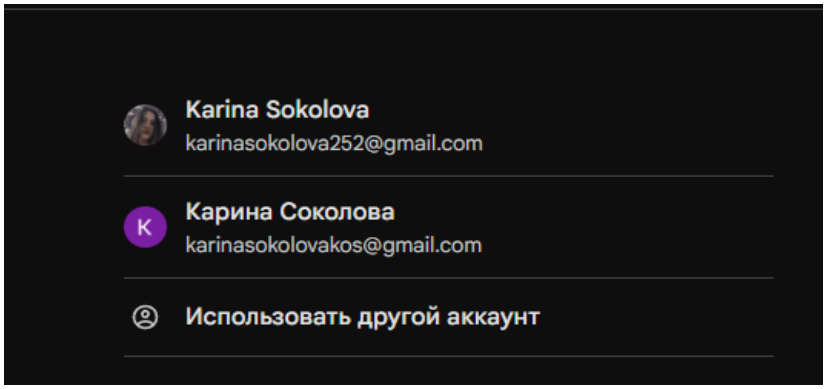


Рисунок 3.14 – Сторінка Google OAuth

На рисунку 3.15 наведено інтерфейс сторінки кімнати вебзастосунку TolskyChat.

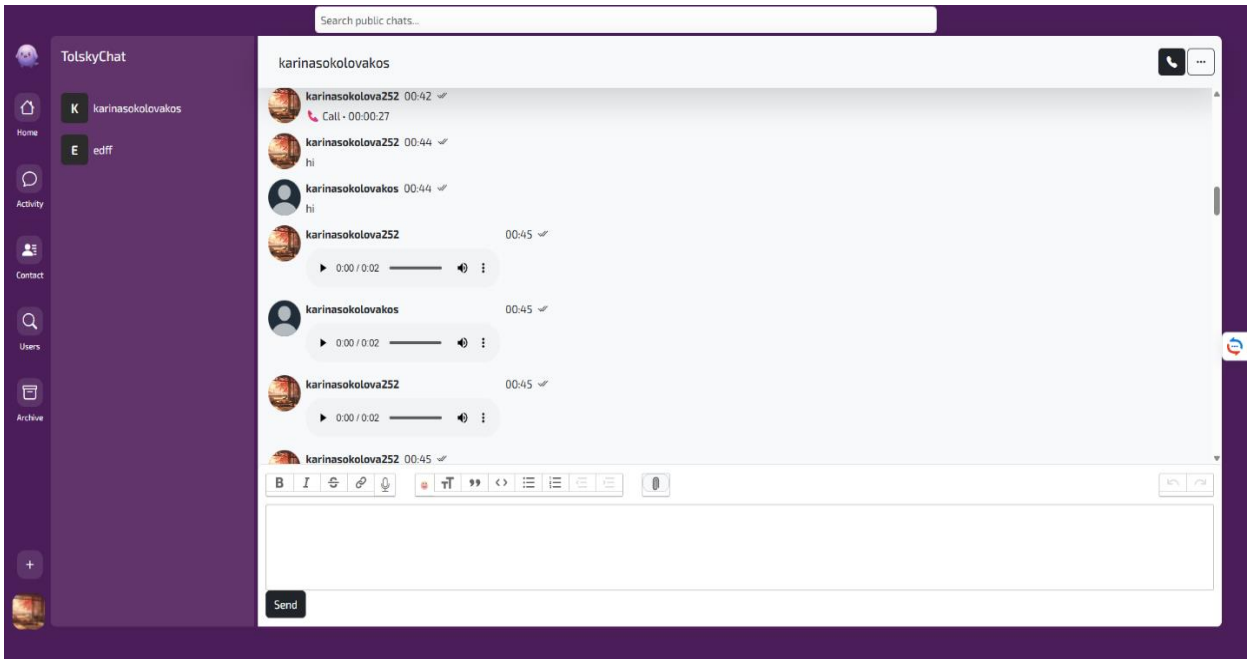


Рисунок 3.15 – Інтерфейс кімнати

На даній сторінці реалізовано основний функціонал системи обміну повідомленнями, зокрема надсилання текстових повідомлень, мультимедійних файлів, використання емодзі, а також підтримку аудіо- та відеодзвінків у режимі реального часу.

На рисунку 3.16 наведено інтерфейс аудіо- та відеодзвінка. Для реалізації аудіо- та відеодзвінків використовується WebRTC-з’єднання, що забезпечує передачу медіапотоку між користувачами у режимі реального часу.

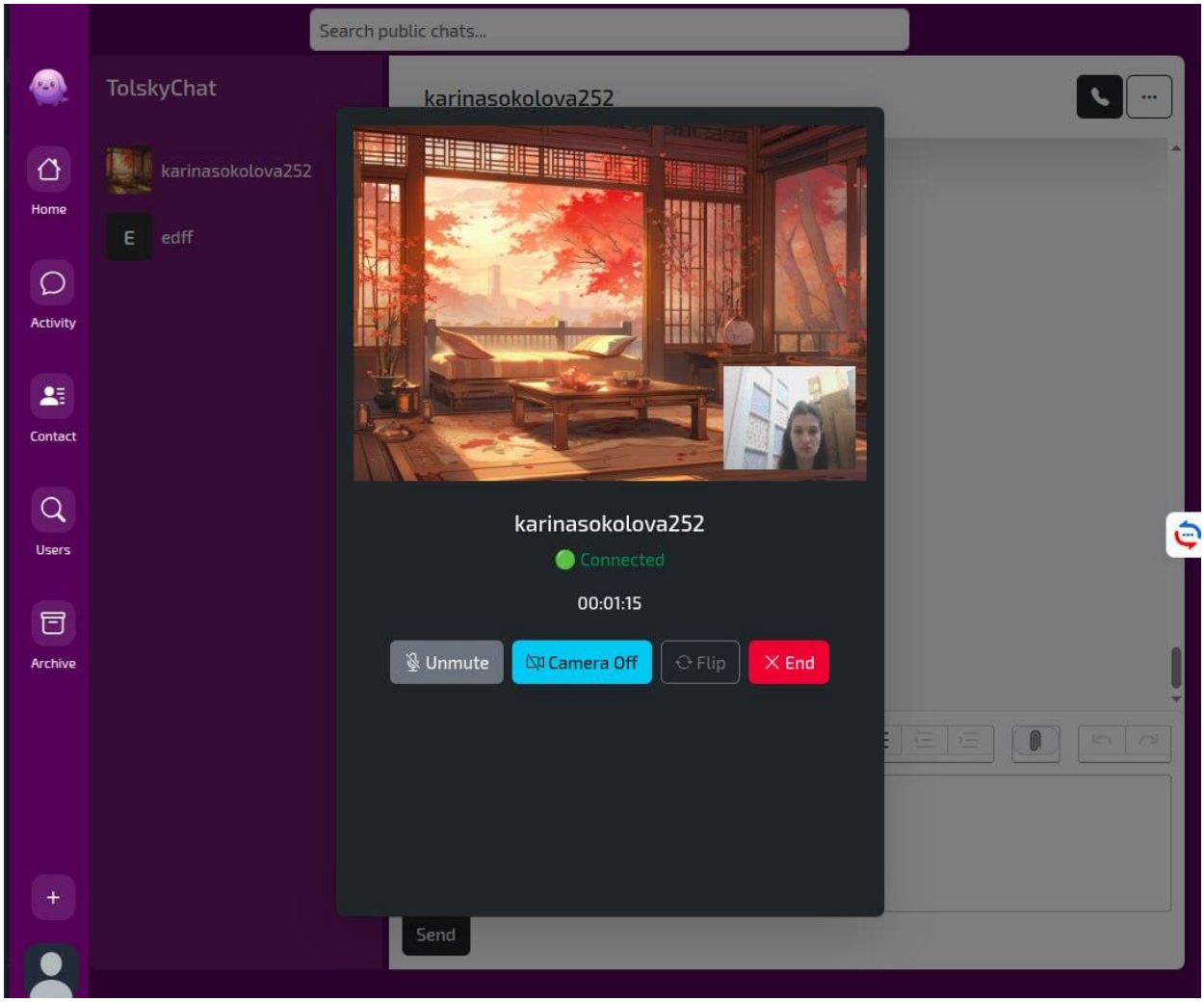


Рисунок 3.16 – Інтерфейс аудіо- та відеодзвінка

Інтерфейс дзвінка надає можливість керування мікрофоном, камерою та завершенням виклику, а також відображає стан активного з’єднання між користувачами.

3.4 Висновки до розділу 3

У третьому розділі було виконано програмну реалізацію та тестування вебзастосуноку багатоканальної комунікації. У процесі розробки реалізовано основні функціональні можливості системи, зокрема автентифікацію користувачів, обмін повідомленнями в режимі реального часу, систему сповіщень, роботу з контактами та підтримку аудіо- та відеодзвінків. Також було проведено тестування основних компонентів системи та наведено загальний огляд інтерфейсу вебзастосунку й реалізованого функціоналу.

ВИСНОВКИ

У кваліфікаційній роботі розроблено вебзастосунок багатоканальної комунікації засобами фреймворку Ruby on Rails та системою керування базами даних PostgreSQL. У процесі розробки було налаштовано систему автентифікації користувачів, реалізовано механізм обміну повідомленнями в режимі реального часу, підтримку аудіо- та відеодзвінків, а також проведено тестування основних функціональних можливостей системи.

У першому розділі здійснено аналіз предметної області через дослідження аналогічних застосунків. На основі результатів огляду сформульовано функціональні та нефункціональні вимоги до системи, а також описано обрані програмні засоби.

Другий розділ охоплює процес концептуального, логічного та фізичного проєктування вебзастосунку призначеного для багатоканальної комунікації. У результаті визначено ключові функціональні можливості системи, сформовано структуру бази даних, архітектуру застосунку та описано взаємодію між компонентами застосунку.

Третій розділ містить фрагменти програмної реалізації системи TolskyChat засобами фреймворка Ruby on Rails. Для реалізації було використано систему керування базами даних PostgreSQL, автентифікацію користувачів налаштовано засобами Devise та Google OAuth. Для підтримки аудіо- та відеодзвінків використано сервіс WebRTC та Twilio.

Таким чином розроблено вебзастосунок багатоканальної комунікації та реалізовано його основні функціональні можливості. Подальший розвиток системи може бути спрямований на розширення функціоналу аудіо- та відеодзвінків, зокрема реалізацію механізму групових дзвінків.

ПЕРЕЛІК ПОСИЛАНЬ

1. Getting Started with Rails. URL: https://guides.rubyonrails.org/getting_started.html (дата звернення: 18.02.2026).

2. What is PostgreSQL. URL: https://www.postgresql.org/about/ (дата звернення: 20.02.2026).

3. Action Cable Overview. URL: https://guides.rubyonrails.org/action_cable_overview.html (дата звернення: 20.02.2026).

4. Devise. Flexible authentication solution for Rails with Warden. URL: https://github.com/heartcombo/devise (дата звернення: 23.02.2026).

5. WebRTC. URL: https://webrtc.org/?hl=en (дата звернення: 25.02.2026).

6. MediaRecorder – WebAPIs. URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder (дата звернення: 25.02.2026).

7. Conversational AI and APIs for SMS, Email, Voice|Twilio. URL: https://www.twilio.com/en-us (дата звернення: 26.02.2026).

8. Sidekiq. Simple, efficient background processing for Ruby. URL: https://github.com/sidekiq/sidekiq (дата звернення: 27.02.2026).

9. TolskyChat. URL: https://github.com/Sokolova2/TolskyChat (дата звернення: 23.05.2026).