НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення для організації онлайн аудіо- та відеозв'язку»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Максим НЕДЬОШЕВ**
Керівник бакалаврської кваліфікаційної роботи д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Виконав	_____ (підпис)	**Максим САУХ**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Сауху Максиму Андрійовичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення для організації онлайн аудіо- та відеозв'язку»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):
Навчальні матеріали для роботи з WebRTC
Перелік питань, що підлягають дослідженню:
Аналіз предметної області організації аудіо- та відеозв'язку
Розробка програмного забезпечення організації онлайн аудіо- та відеозв'язку
Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Максим НЕДЬОШЕВ**

Консультант бакалаврської кваліфікаційної роботи _____ (підпис) **Роман РУДЕНСЬКИЙ**

Завдання взяв до виконання _____ (підпис) **Максим САУХ**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API— програмний інтерфейс додатку (Application Programming Interface).

БД — база даних.

HTTPS — захищений протокол передачі гіпертексту.

ICE — Interactive Connectivity Establishment, механізм встановлення з’єднання WebRTC через NAT.

NAT— перетворення мережевих адрес (Network Address Translation).

P2P — безпосередня взаємодія вузлів (peer-to-peer).

REST — архітектурний стиль взаємодії через HTTP-ресурси.

RLS — політики безпеки на рівні рядків (Row Level Security) у PostgreSQL / Supabase.

SQL— мова структурованих запитів.

STUN — служба визначення зовнішньої адреси для WebRTC.

TLS — транспортний рівень безпеки.

UML — уніфікована мова моделювання.

WebRTC — технологія реального часу у браузері для аудіо/відео та обміну даними.

WebSocket — двосторонній канал поверх TCP для інтерактивних додатків.

SFU — сервер для відеодзвінків, який пересилає медіапотоки між учасниками без їх обробки.

MCU — сервер для відеоконференцій, який об’єднує та обробляє аудіо/відео потоки всіх учасників в один.

STUN — протокол, який допомагає пристрою визначити свою зовнішню IP-адресу для встановлення P2P-з’єднання.

TURN — протокол, який передає трафік через проміжний сервер, якщо пряме з’єднання між пристроями неможливе.

XSS — тип веб-атаки, при якому зловмисник вставляє шкідливий JavaScript-код на вебсторінку для виконання в браузері користувача.

ВСТУП

Сьогодні люди все частіше спілкуються та працюють через інтернет: дистанційна робота, навчання, наради та зустрічі з колегами з різних міст уже звична практика [1, 2]. Для цього потрібні зручні й надійні засоби відеозв’язку, які можна відкрити на комп’ютері чи в браузері без складного встановлення окремих програм. Тому розробка веб-додатків для відеоконференцій залишається важливою й затребуваною темою в галузі інформаційних технологій [3].

Найпоширеніші комерційні сервіси відеозв’язку добре підходять для повсякденного використання, але мають і недоліки: залежність від правил постачальника, обмеження за тарифами й регіонами, а іноді й складність з тим, щоб повністю контролювати налаштування системи. Для навчання та для невеликих проєктів часто цікавіше мати власне рішення, яке можна змінювати, вивчати «зсередини» і адаптувати під конкретні задачі. Саме тому актуальним є створення веб-додатку, який поєднує сучасні підходи до відеозв’язку з нормальною реєстрацією користувачів і базовими соціальними функціями — профіль, друзі, інформація про те, хто зараз онлайн.

Сучасні браузери підтримують технології реального часу для передачі звуку та відео між учасниками. Щоб зустріч відбулася, потрібен не лише сам зв’язок між людьми, а й серверна частина, яка допомагає «познайомити» учасників один з одним у мережі та передає службові повідомлення. Окремо важливо передбачити безпечний вхід до системи та збереження даних користувачів — для цього зручно використовувати хмарні сервіси, які надають готові механізми автентифікації та роботи з базою даних [20].

Актуальність теми полягає в тому, що під час підготовки фахівця з інженерії програмного забезпечення важливо навчитися проєктувати й збирати разом клієнтську частину в сайті, сервер додатку та підключення до хмарного сервісу — це типова сучасна схема розробки.

Мета роботи — спроєктувати й реалізувати веб-систему для відеоконференцій у браузері: кімнати за кодом, відео- та аудіозв’язок між учасниками, текстовий чат у кімнаті, обмеження кількості учасників, а для зареєстрованих користувачів — кабінет з профілем, заявками в друзі та відображенням присутності.

Завдання роботи:

1. Розглянути предметну область і вимоги до системи, описати основні сценарії роботи користувачів і будову програмних компонентів.
2. Спроєктувати дані для облікових функцій і загальну архітектуру рішення, обґрунтувати вибір технологій.
3. Реалізувати програмне забезпечення: сервер додатку, клієнтську частину з відеозв’язком і обміном повідомленнями, підключення до хмарної автентифікації та бази даних, інтерфейс сторінок.
4. Підготувати рекомендації з тестування, вимог до комп’ютера та програм, а також з розгортання системи.
5. Перевірити роботу основних функцій за тестовими сценаріями й підсумувати результати.
6. Запропонувати напрями подальшого розвитку (наприклад, краща підтримка складних мереж, запис зустрічей тощо).

Об’єкт дослідження — процес організації відеозустрічі в браузері з використанням технологій реального часу та обліку користувачів у хмарі.

Предмет дослідження — конкретне програмне забезпечення та підхід до його створення: веб-клієнт, сервер додатку, засоби відеозв’язку та інтеграція з хмарним сервісом автентифікації й даних.

Методи дослідження: аналіз літератури й офіційної документації; опис і моделювання предметної області; порівняння існуючих рішень; проєктування та програмування; перевірка роботи системи на практиці; узагальнення висновків і рекомендацій.

Наукова новизна полягає у поєднанні у відносно простій системі відкритих веб-підходів до відеозв’язку з готовою хмарною платформою для користувачів і соціальних функцій, без розробки всієї підсистеми входу та зберігання даних.

Практична цінність — отриманий програмний продукт і описані рекомендації можуть використовуватися як навчальний приклад повного циклу створення веб-додатку або як основа для подальшого розширення під конкретні потреби за належного налаштування безпеки.

Структура записки: вступ; розділ із аналізом предметної області та вимог; розділ із проєктуванням даних і архітектури; розділ із описом реалізації; розділ із рекомендаціями щодо тестування та впровадження; висновки; список джерел; додатки з діаграмами, фрагментами коду та матеріалами перевірки.

Таким чином, робота показує практичний шлях від постановки задачі до робочого веб-додатку для відеоконференцій і корисна для формування навичок сучасної веб-розробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Синхронна відеокомунікація через Інтернет стала однією з базових інфраструктурних послуг сучасної економіки та суспільства. Відеозустрічі використовують у корпоративному секторі, державному адмініструванні, охороні здоров'я, освіті, креативних індустріях і приватному спілкуванні. Після поширення дистанційних форматів роботи та навчання частка процесів, де відеозв'язок є регулярним інструментом, залишилася структурно високою: організації інвестують у цифрові платформи співпраці, а користувачі формують стійкі очікування щодо доступності сервісу «з будь-якого пристрою» та передбачуваної якості аудіо й відео [1, 2]. Таким чином, предметна область охоплює не лише технічну передачу медіа, а й організаційні та соціальні практики віддаленої взаємодії, норми безпеки даних і типові сценарії відмовостійкості [4].

Економічний вимір галузі багатошаровий. З одного боку, відеоплатформи знижують витрати на відрядження, прискорюють узгодження рішень і підтримують безперервність бізнес-процесів при географічно розподілених командах. З іншого — постачальники комерційних сервісів формують ринок підписок і корпоративних ліцензій, а інтеграція відеокомунікацій у CRM, системи навчання та медичні інформаційні системи створює додану вартість для екосистеми програмного забезпечення [3, 5]. Для малих організацій і навчальних закладів важливим залишається співвідношення вартість / контроль / гнучкість: готові хмарні рішення прискорюють запуск, але обмежують глибину кастомізації та залежність від політики постачальника.

Соціальний аспект пов'язаний із доступністю послуг, цифровою включеністю та якістю життя. Відеозв'язок знижує бар'єри для людей у віддалених населених пунктах, для людей з обмеженою мобільністю, для учасників освітніх програм без релокації. Водночас зростає значення етикету

цифрової присутності, регламентів запису зустрічей, приватності домашнього простору, що потрапляє в кадр, а також питань психологічного навантаження від «безперервної доступності». У контексті України ці фактори поєднуються з завданнями відновлення та стійкості інфраструктури, цифровізації послуг і підтримки дистанційних форматів у освіті та державних комунікаціях, що підсилює попит на зрозумілі, керовані та безпечні інструменти взаємодії.

На рівні понятєвого апарату предметну область доцільно описувати через кілька взаємопов'язаних елементів. Сесія відеокомунікації — це узгоджений у часі процес обміну медіа та/або даними між двома або більше учасниками. Кімната (канал, зустріч) — логічний контейнер доступу: набір правил, хто може приєднатися, як ідентифікується місце зустрічі. Клієнтські програми можуть бути нативними або реалізованими у веб-браузері; веб-підхід зменшує бар'єр входу для користувача, але накладає вимоги до сумісності стандартів і продуктивності на стороні клієнта. Сигналізація у мережевих архітектурах реального часу — це обмін службовою інформацією для встановлення параметрів з'єднання та проходження мережевих перетворень адрес; без неї «сирі» медіапотоки між браузерами, як правило, не узгоджуються. Медіа-топологія визначає, чи йде трафік безпосередньо між учасниками (типово для малих груп), чи агрегується на медіа-сервері (SFU/MCU) для великих конференцій — це фундаментальна класифікаційна відмінність у галузі, що визначає вартість експлуатації та складність реалізації [2, 3].

Технічні виклики предметної області не зводяться до «швидкого інтернету». Важливі: затримка (latency), втрати пакетів, адаптивне кодування, ехо та шум, синхронізація аудіо/відео, стабільність при зміні мережі. Окремий пласт — проходження NAT і фаєрволів, для якого в інженерній практиці використовують служби визначення адреси та ретрансляцію (STUN/TURN), що впливає на архітектуру і витрати [6, 7, 8]. Паралельно існує вимога захищеного транспорту (TLS/HTTPS), керування ключами, моделі загроз для веб-клієнта (XSS, витоки токенів, неправильні політики доступу до даних) [29].

Інформаційна безпека та приватність у сфері відеоконференцій є самостійним блоком предметної області. До нього належать: хто зберігає метадані (час, учасники, IP), чи є запис зустрічі, де зберігаються файли чату, як реалізовано автентифікацію та авторизацію, чи застосовуються механізми контролю доступу на рівні даних у базах. Регуляторні рамки (зокрема вимоги до персональних даних) впливають на проєктування журналювання, згоди користувача та георозподіл обробки даних [29, 30].

Ринкова структура включає великі комерційні платформи, відкриті рішення з можливістю самостійного розгортання, а також нішеві та навчально-дослідницькі розробки, де метою є демонстрація повного циклу інженерії або адаптація під внутрішні політики організації. Для освітніх і дослідницьких цілей типовим є поєднання відкритих веб-стандартів реального часу з відносно простим сервером координації та зовнішніми сервісами ідентифікації, що знижує вартість підтримки облікових записів без відмови від сучасних практик безпеки.

Тренди галузі включають інтеграцію з календарями та ідентичностями, штучний інтелект для шумопригнічення й мовних сервісів, гібридні формати подій, підвищені вимоги до прозорості архітектури та можливості самостійного аудиту для корпоративних замовників. Для веб-сегмента стабільно важливою залишається опора на стандартизовані технології браузера, що забезпечує кросплатформеність і довгострокову підтримку екосистемою виробників браузерів [1, 10, 12].

Предметна область цієї роботи — це галузь програмних засобів і сервісів синхронної відеокомунікації в мережі Інтернет, зокрема у формі веб-додатків, з усіма супутніми питаннями: ролями учасників і моделлю «зустрічі», мережевими та медіа-обмеженнями, безпекою, економікою постачання та соціальними наслідками масового використання. Подальший аналіз сучасних методів, архітектур і програмних технологій у цій сфері дозволяє обґрунтувати вимоги до конкретної програмної системи та вибір підходів до її реалізації в наступних розділах записки.

1.2 Аналіз вимог до програмної системи

Вимоги до системи відеозв’язку зводяться до того, щоб користувач міг швидко зайти в зустріч, почути й побачити інших учасників і за потреби написати в чат, а сервер при цьому допомагав узгодити з’єднання між браузерами. Окремо варто врахувати безпеку (захищений доступ, обережне зберігання ключів) і простоту інтерфейсу, щоб людині не доводилось довго розбиратися перед першим дзвінком.

З функціональної точки зору система має показувати головну сторінку і сторінку кімнати. У кімнаті має працювати аудіо та відео між учасниками однієї зустрічі, має бути можливість вмикати й вимикати камеру та мікрофон і бачити, хто зараз говорить. Для узгодження зв’язку між браузерами потрібен обмін службовими повідомленнями через сервер. Текстовий чат у кімнаті має доходити до всіх присутніх через сервер. Кількість учасників у кімнаті має бути обмежена. Бажано показувати ім’я та аватар; для зареєстрованих користувачів — підтримка профілю, входу в систему, друзів і інформації про присутність. Для демонстрації ззовні може бути передбачений безпечний веб-доступ через типовий інструмент тунелювання, якщо це потрібно для показу роботи.

З нефункціональної точки зору інтерфейс має бути зручним і адаптивним під різний розмір вікна. Секретні ключі не повинні зберігатися у відкритому вигляді в клієнтській частині; чутливі параметри — у налаштуваннях сервера або змінних оточення. Слід зазначити, що така система розрахована на невеликі зустрічі; для дуже великої кількості людей зазвичай потрібні інші архітектурні рішення з окремим медіа-сервером. Стабільність при коротких обривах мережі може бути покращена в наступних версіях, якщо в першій реалізації це зроблено не повністю.

1.3 Моделювання предметної області

Моделювання предметної області для програмного забезпечення організації аудіо- та відеозв’язку дозволяє перейти від загального опису процесів спілкування в інтернеті до узгодженої моделі функцій, ролей і послідовностей дій, якими керується система. Мета такого моделювання — зменшити неоднозначність формулювань вимог, показати межі відповідальності учасників (користувач, сервер, інші абоненти) і підготувати основу для проєктування програмних модулів та інтерфейсів. У роботі використано набір діаграм UML та пов’язаних з ними графічних моделей, які взаємно доповнюють одна одну [24, 25].

З функціональної точки зору предметну область доцільно представити як множину прецедентів, тобто цілісних користувацьких задач, які система має підтримувати. Для систем аудіо- та відеозв’язку такі прецеденти зазвичай групуються навколо встановлення з’єднання між абонентами, керування джерелами медіа (звук і відео), завершення сеансу та супутнього текстового обміну. Окремо виділяють системні або сервісні прецеденти, що відображають узгодження учасників на рівні логіки зв’язку. Важливим елементом моделі є відношення <<include>>: воно показує, що загальні дії «налаштувати джерело звуку» та «налаштувати джерело відео» обов’язково передбачають можливість вимкнути звук та вимкнути відео як невід’ємні підзадачі керування потоком. Таким чином, діаграма прецедентів не лише перелічує функції, а й фіксує модульність керування медіа: аудіо- та відеоканали можна розглядати окремо, зберігаючи єдиний сценарій зустрічі.

Рис. 1 - Діаграма прецедентів

Після статичного опису функцій необхідно показати динаміку — як дії учасників і сервера розгортаються в часі. Для цього використовують діаграму послідовностей із життєвими лініями двох користувачів та сервера як центрального координатора сценарію. Типовий ланцюг включає запит на встановлення з’єднання, сповіщення другого абонента про вхідний виклик, прийняття виклику та підтвердження з’єднання обом сторонам. Далі модель відображає етапи налаштування мікрофона та камери та передачі аудіо- і відеопотоку. Окремим шляхом показано надсилання текстового повідомлення через сервер і його передачу адресату, що підкреслює багатоканальність комунікації: поруч із медіа існує текстовий канал, який зазвичай реалізують через серверну доставку. Завершення сценарію відбувається через ініціювання завершення виклику однією зі сторін і повідомлення про завершення та роз’єднання, яке сервер доводить до іншого учасника. Варіанти з помилками, повторними спробами та тайм-аутами можна розширити окремими фрагментами діаграми або коментарем у тексті записки, якщо це передбачено вимогами.

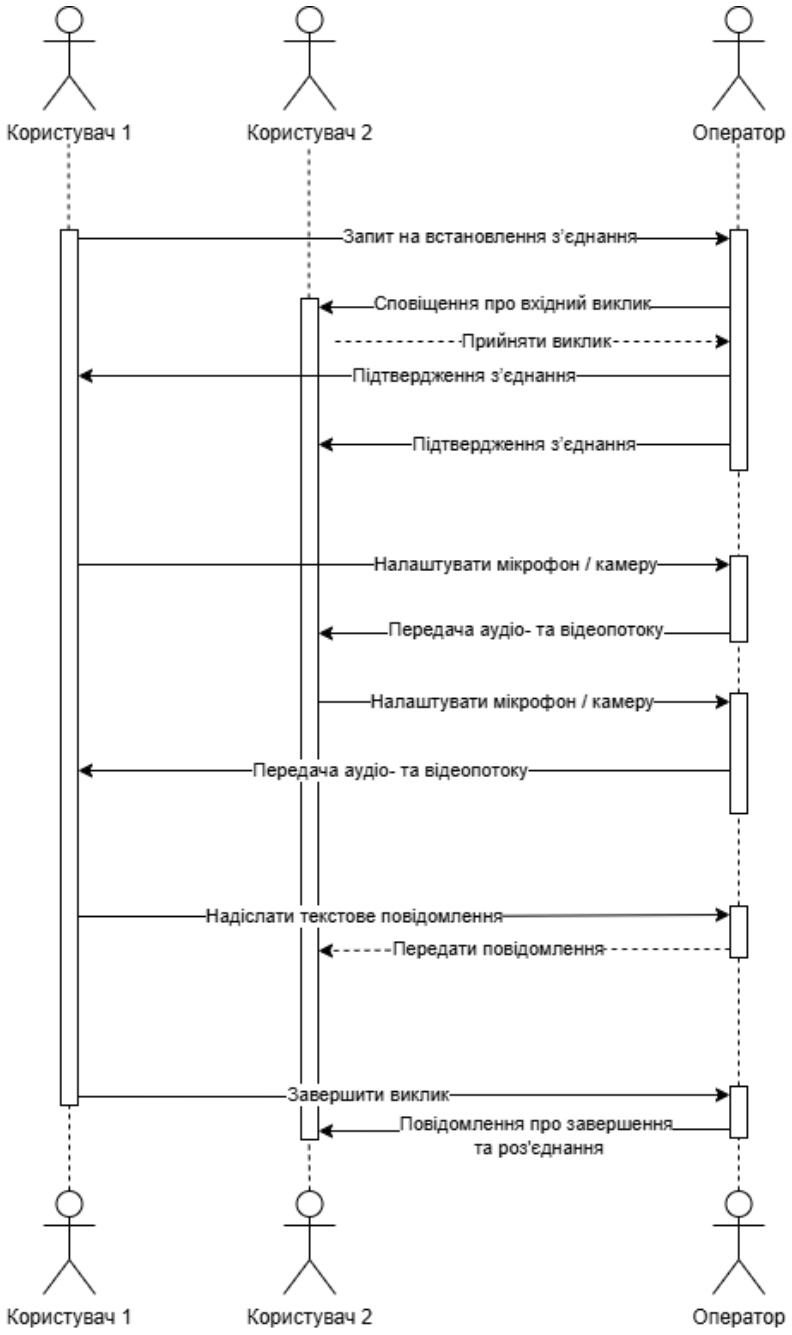

Рис. 2 - Діаграма послідовностей

Окрім послідовності повідомлень, предметну область корисно описати як процес із розвилками та циклами, що відповідає реальній ненадійності мережі та людським рішенням. На діаграмі діяльності (блок-схемі життєвого циклу сеансу) відображено етап ініціації сеансу, перевірку ідентифікаторів абонентів (узагальнено як контроль коректності учасників), запит на встановлення з'єднання та ключове рішення щодо успіху з'єднання. Якщо з'єднання не встановлено, модель допускає повторну спробу або припинення ініціації, що

формально фіксує вимогу до поведінки системи в умовах відмови. У разі успіху процес переходить до обміну службовими даними (параметрами зв’язку), після чого паралельно відображаються передача відеосигналу та передача аудіосигналу, а сеанс перебуває у стані підтримки зв’язку до моменту, поки не надійде сигнал завершення. Після ініціювання завершення виконується припинення передачі даних і розірвання з’єднання. Такий опис узгоджується з інженерною практикою: навіть якщо на нижньому рівні використовують спеціалізовані протоколи реального часу, на рівні предметної області важливо явно виділити фази узгодження, робочого обміну та коректного закриття ресурсів.

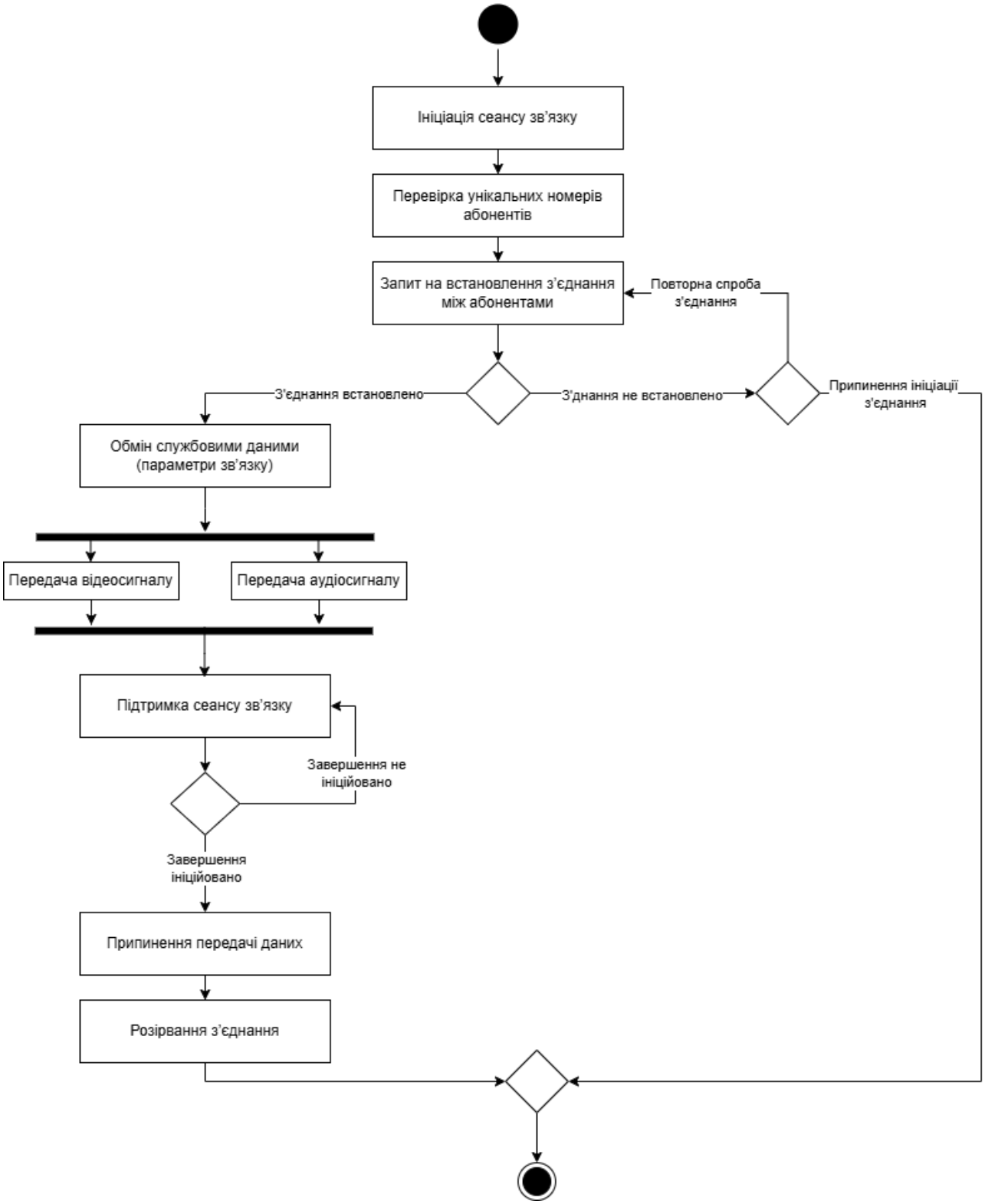

Рис. 3 - Діаграма діяльності життєвого циклу сеансу

Для узгодження «людина — пристрій — дані» доцільно застосувати концептуальну модель абстракцій, де виділено базові сутності предметної області та їхні зв'язки. У представленій моделі користувач виступає ініціатором

і споживачем послуги зв’язку; оператор (у сенсі керуючого компонента / логіки сеансу) координує дії та керує ресурсами взаємодії. Камера та мікрофон моделюються як джерела відео- та аудіоданих, які підключаються до формування медіапотоку. Зв’язки між абстракціями підкреслюють, що медіапотік є результатом узгодженої роботи пристроїв вводу та керуючої логіки, а користувач взаємодіє з системою через цю логіку, а не «напряму» з низькорівневими драйверами. Така діаграма корисна для переходу до проєктування: вона підказує природні межі модулів (керування сеансом, захоплення медіа, транспорт потоку) і допомагає уникнути змішування рівнів відповідальності в одному компоненті.

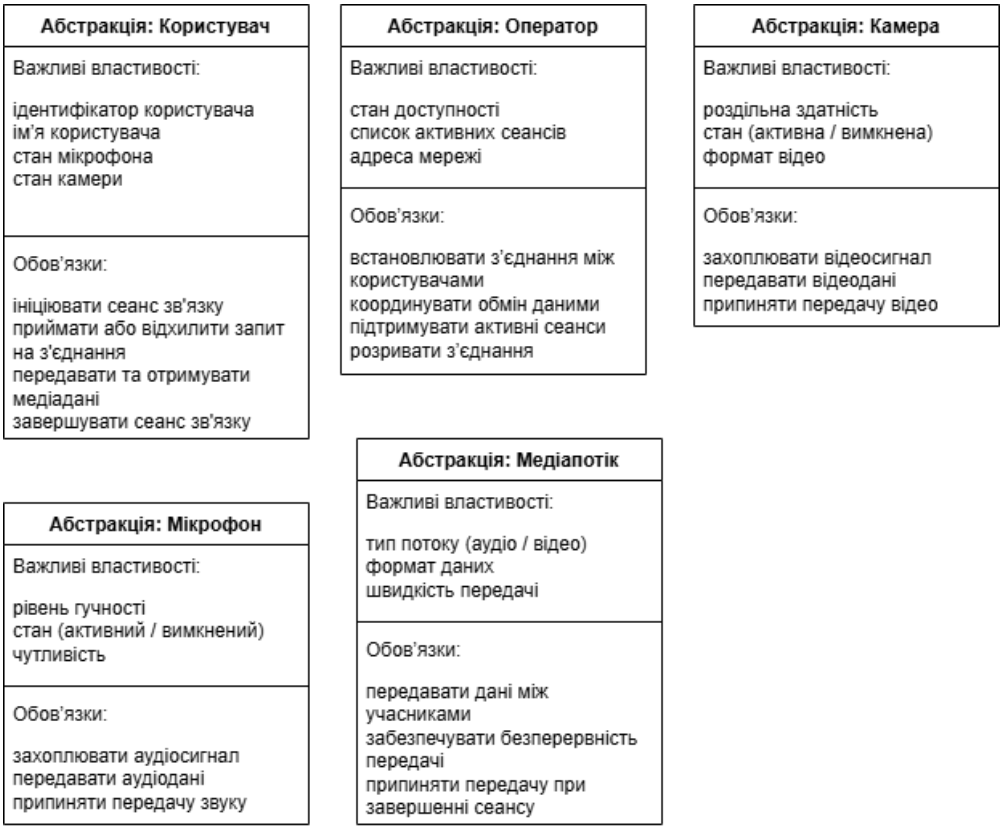

Рис. 4 - Діаграма абстракцій предметної області

У сукупності чотири діаграми формують багаторівневе моделювання: прецеденти задають «що система робить», послідовність — «у якому порядку це відбувається між учасниками», діяльність — «як процес розвивається з урахуванням успіху, повторів і завершення», абстракції — «з яких понять

складається предметна область». Це створює логічний місток до наступних розділів: огляду існуючих рішень, постановки інженерного завдання та проєктування конкретної програмної реалізації, де окремі кроки діаграм відображаються в архітектурі клієнта, сервера та зовнішніх сервісів.

1.4 Огляд існуючих рішень для організації зв’язку онлайн

На сьогодні існує багато програмних рішень для організації онлайн-зв’язку, які використовуються для навчання, роботи, ділових зустрічей, консультацій та повсякденного спілкування. Найвідомішими серед них є Zoom, Google Meet, Microsoft Teams, Skype, Discord, Jitsi Meet та BigBlueButton [32, 33, 34, 35]. Кожне з цих рішень має власну цільову аудиторію, набір функцій, переваги й обмеження. Їхній аналіз дозволяє визначити, які можливості є типовими для сучасних систем відеозв’язку, а також які недоліки можна врахувати під час розробки власного програмного забезпечення.

Одним із найпоширеніших рішень є Zoom [34]. Його перевагами є стабільна якість зв’язку, зручний інтерфейс, підтримка великої кількості учасників, можливість запису зустрічей, демонстрації екрана, створення окремих кімнат і планування конференцій. Zoom активно використовується в бізнесі, освіті та для проведення вебінарів. Водночас цей сервіс має й недоліки: частина важливих функцій доступна лише в платних тарифах, безкоштовні зустрічі можуть мати часові обмеження, а користувач залежить від політики компанії-постачальника. Крім того, для повноцінної роботи часто зручніше встановлювати окремий клієнт, що не завжди підходить для простих або навчальних сценаріїв.

Google Meet є популярним інструментом для відеозустрічей, особливо серед користувачів екосистеми Google. Його головна перевага — простий доступ через браузер, інтеграція з Google Calendar, Gmail та Google Workspace. Користувач може швидко створити зустріч і надіслати посилання іншим учасникам. Сервіс добре підходить для навчальних і робочих нарад. До недоліків

можна віднести залежність від облікового запису Google та обмежену гнучкість налаштування. Для користувачів, які не працюють у середовищі Google, частина переваг сервісу втрачає значення.

Microsoft Teams [35] орієнтований переважно на корпоративну роботу та навчальні організації. Його сильними сторонами є інтеграція з Microsoft 365, підтримка команд, каналів, файлів, календаря, спільної роботи з документами. Teams є не лише засобом відеозв'язку, а й повноцінною платформою для організації робочих процесів. Однак саме через велику кількість можливостей сервіс може бути складним для нового користувача. Для простої відеозустрічі інтерфейс іноді виглядає перевантаженим, а налаштування прав доступу та організаційної структури потребують додаткового адміністрування.

Skype є одним із найстаріших популярних сервісів інтернет-зв'язку. Його перевагами є знайомий багатьом користувачам інтерфейс, підтримка голосових і відеодзвінків, текстових повідомлень і дзвінків на телефонні номери. Водночас за останні роки Skype частково втратив позиції через появу новіших платформ, які краще інтегровані з корпоративними або освітніми сервісами. Для сучасних веб-сценаріїв його можливості можуть бути менш гнучкими, а розвиток продукту не такий активний, як у Teams, Meet або Zoom.

Discord спочатку був орієнтований на ігрові спільноти, але нині активно використовується для навчальних груп, спільнот розробників і неформальної командної роботи. Його перевагами є голосові канали, текстові канали, ролі користувачів, гнучке керування спільнотами та зручна постійна структура серверів. Однак Discord не завжди підходить для офіційних ділових або навчальних зустрічей, оскільки його логіка більше орієнтована на спільноти, а не на разові конференції. Крім того, для деяких організацій питання контролю даних і відповідності внутрішнім політикам можуть бути обмеженням.

Серед відкритих рішень варто виділити Jitsi Meet [32]. Його перевагою є відкритий код, можливість самостійного розгортання та використання без обов'язкової прив'язки до великої комерційної платформи. Jitsi зручно застосовувати там, де важливий контроль над інфраструктурою. Водночас

самостійне розгортання й підтримка такого рішення потребують технічних знань, серверних ресурсів і налаштування безпеки. Для невеликого навчального проєкту повноцінне розгортання складної платформи може бути надмірним.

Ще одним спеціалізованим рішенням є BigBlueButton [33], яке часто використовується в освітньому середовищі. Система підтримує онлайн-заняття, презентації, дошку, чат, керування учасниками та інтеграцію з навчальними платформами. Її перевага полягає у спрямованості саме на освітній процес. Недоліком є відносна складність розгортання та адміністрування, а також те, що для простих відеозустрічей набір функцій може бути надмірним.

Загалом існуючі рішення можна умовно поділити на дві групи. Перша група — це комерційні хмарні сервіси, такі як Zoom, Google Meet і Microsoft Teams. Вони мають високу стабільність, багато готових функцій і підтримку великої кількості користувачів, але обмежують розробника або організацію рамками своєї платформи. Друга група — це відкриті або самостійно розгорнуті рішення, такі як Jitsi Meet і BigBlueButton. Вони дають більше контролю, але потребують складнішої технічної підтримки та окремої інфраструктури.

Розроблюване у цій роботі програмне забезпечення не має на меті повністю замінити великі комерційні платформи. Його перевага полягає в іншому: воно є простішим, прозорішим і краще пристосованим для навчального та демонстраційного використання. На відміну від великих сервісів, система має зрозумілу структуру: веб-клієнт, сервер для координації з’єднання та хмарний сервіс для облікових функцій. Це дозволяє краще простежити, як саме організовується відеозв’язок, як працює кімната, чат, профіль користувача та інші елементи.

Певною перевагою власного програмного забезпечення є також можливість адаптації. У готових платформах користувач здебільшого приймає вже визначені правила роботи, інтерфейс і набір функцій. У власній системі можна змінювати логіку кімнат, зовнішній вигляд, кількість учасників, спосіб авторизації, структуру профілю та інші елементи відповідно до потреб

конкретного проєкту. Крім того, така система не перевантажена зайвими корпоративними функціями, які не потрібні для невеликих групових зустрічей.

1.5 Постановка завдання

На основі аналізу предметної області, вимог до програмної системи та огляду існуючих рішень було визначено завдання розробити веб-додаток для організації аудіо- та відеозв’язку між користувачами в режимі реального часу. Система має забезпечувати створення або вибір кімнати за коротким кодом, підключення кількох учасників, передачу аудіо- та відеопотоку, обмін текстовими повідомленнями, а також базові можливості керування мікрофоном і камерою.

У межах роботи необхідно реалізувати серверну частину, яка відповідатиме за видачу веб-сторінок, обмін службовими повідомленнями між учасниками та роботу чату. Клієнтська частина має забезпечувати зручний інтерфейс головної сторінки, сторінки відеокімнати та особистого кабінету користувача. Для зареєстрованих користувачів потрібно передбачити профіль, аватар, пошук інших користувачів, заявки в друзі та відображення присутності.

Отже, основне завдання роботи полягає у створенні працездатного веб-рішення для невеликих онлайн-зустрічей, яке поєднує аудіо- та відеозв’язок, текстовий чат і базові облікові функції. Результатом виконання завдання має бути програмне забезпечення, придатне для демонстрації, тестування та подальшого розвитку.

Висновки до розділу

У першому розділі було розглянуто предметну область програмних засобів для організації аудіо- та відеозв’язку онлайн. Визначено, що такі системи є актуальними для навчання, роботи, консультацій і повсякденного спілкування, оскільки дозволяють користувачам взаємодіяти на відстані без потреби фізичної присутності.

Було проаналізовано основні вимоги до програмної системи: можливість створення або вибору кімнати, підключення учасників, передача аудіо- та

відеопотоку, текстовий чат, керування мікрофоном і камерою, а також базові облікові функції для зареєстрованих користувачів. Окремо враховано нефункціональні вимоги: зручність інтерфейсу, безпека налаштувань, обмеження щодо кількості учасників і залежність якості зв’язку від мережевих умов.

Для кращого розуміння предметної області було використано моделювання: діаграми показують основні функції системи, послідовність взаємодії користувачів і сервера, загальний процес встановлення та завершення зв’язку, а також основні абстракції, пов’язані з користувачем, пристроями та медіапотоком.

Також було виконано огляд існуючих рішень для онлайн-зв’язку, зокрема Zoom, Google Meet, Microsoft Teams, Skype, Discord, Jitsi Meet і BigBlueButton. Встановлено, що готові платформи мають багато переваг, але часто залежать від комерційної екосистеми, можуть бути перевантажені функціями або потребувати складного розгортання. Це підтверджує доцільність створення власного простішого й керованого веб-рішення для невеликих онлайн-зустрічей.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних є важливою частиною проєктування інформаційного забезпечення, оскільки вона показує, які дані зберігаються в системі, як вони пов’язані між собою та які об’єкти предметної області потрібно врахувати під час подальшої реалізації бази даних. Для програмного забезпечення організації аудіо- та відеозв’язку основними інформаційними об’єктами є користувачі, їхні профілі, кімнати для зв’язку, участь користувачів у кімнатах та заявки на дружбу.

У межах розроблюваної системи можна виділити кілька основних сутностей. Сутність «Кімната» описує логічний простір, у якому відбувається аудіо- та відеозустріч. Кімната може мати власний ідентифікатор або код, за яким користувачі підключаються до неї. Такий підхід дозволяє відокремити одну зустріч від іншої та забезпечити зрозумілий спосіб входу для учасників.

Сутність «Користувачі в кімнаті» відображає зв’язок між конкретними користувачами та конкретною кімнатою. Один користувач може приєднуватися до різних кімнат у різний час, а одна кімната може містити кількох учасників. Тому така сутність фактично виконує роль проміжної таблиці, яка дозволяє описати зв’язок типу «багато до багатьох» між користувачами та кімнатами.

Сутність «Авторизовані користувачі» зберігає основні дані користувачів, які мають обліковий запис у системі. Вона є базою для персоналізованих функцій: входу в систему, відображення імені користувача, зв’язку з профілем, заявками в друзі та іншими даними, що стосуються конкретної особи. Авторизований користувач може брати участь у кімнатах, мати профіль і взаємодіяти з іншими користувачами через механізм дружби.

Сутність «Профілі» містить додаткову інформацію про авторизованого користувача, наприклад відображуване ім’я, аватар або інші персональні налаштування. Винесення профільних даних в окрему сутність є доцільним,

оскільки обліковий запис і публічне представлення користувача виконують різні ролі. Обліковий запис відповідає за ідентифікацію, а профіль — за зручне відображення користувача в інтерфейсі.

Сутність «Запити на дружбу» описує соціальну взаємодію між авторизованими користувачами. Один користувач може надіслати заявку іншому, а стан такої заявки може змінюватися залежно від рішення отримувача. Така сутність дозволяє реалізувати список контактів або друзів, що корисно для швидкого пошуку користувачів і подальшого приєднання до спільних онлайн-зустрічей.

Зв’язки між сутностями відображають логіку роботи системи. Кімната пов’язана з користувачами через сутність «Користувачі в кімнаті». Авторизований користувач має власний профіль. Між авторизованими користувачами можуть існувати запити на дружбу, які показують напрямок соціальної взаємодії: хто надіслав заявку і хто її отримав. Така структура дозволяє зберігати як інформацію про самі відеозустрічі, так і дані, необхідні для персоналізації та взаємодії користувачів.

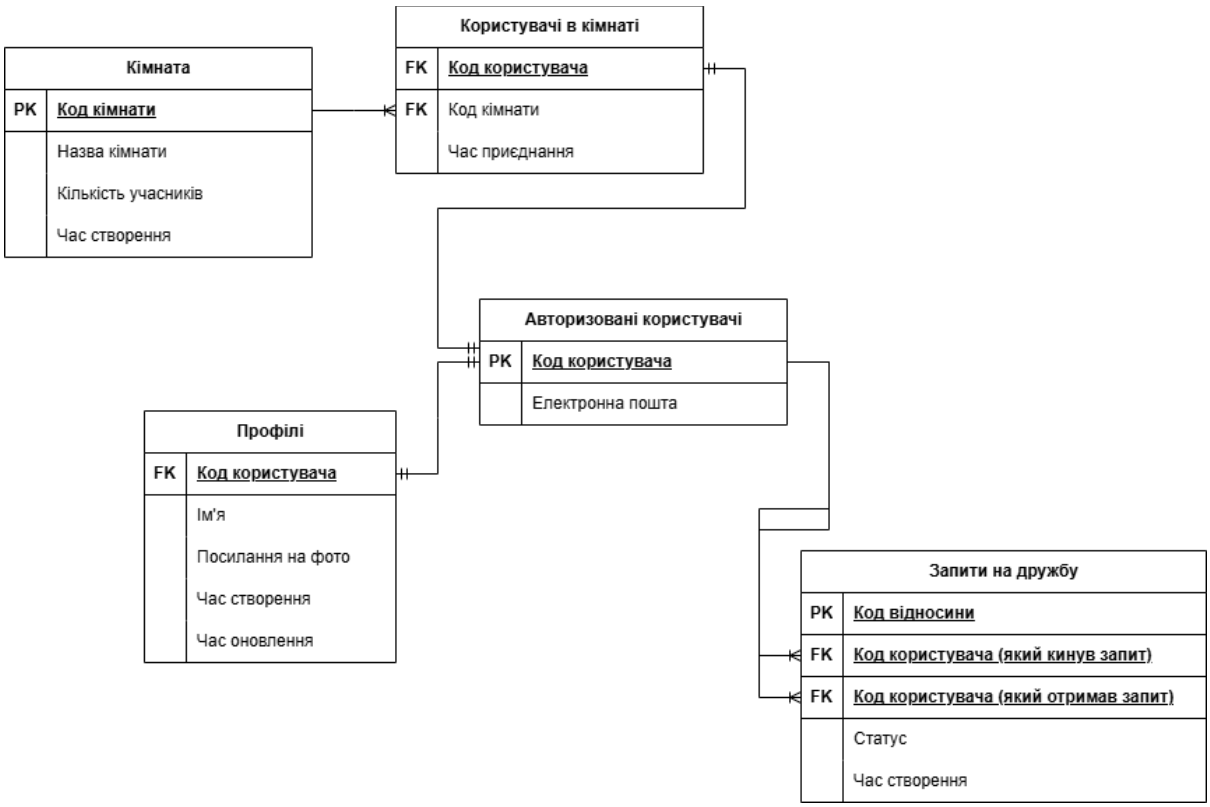

Рис. 5 - ER-діаграма логічної моделі даних

Логічна модель даних, зображена на ER-діаграмі, є основою для подальшого створення фізичної структури бази даних. На її основі можна визначити таблиці, первинні та зовнішні ключі, типи зв’язків і обмеження цілісності. Запропонована модель є достатньою для реалізації основних функцій системи: створення кімнат, участі користувачів у зустрічах, збереження профілів і підтримки запитів на дружбу. Надалі ця модель може бути розширена додатковими сутностями, наприклад історією повідомлень, журналом входів або налаштуваннями користувача.

2.2 Вибір системи управління базами даних

Для розроблюваної системи організації аудіо- та відеозв’язку важливо обрати таку систему управління базами даних, яка забезпечує надійне збереження інформації про користувачів, профілі, заявки в друзі та поточну присутність у системі. Оскільки програмне забезпечення має веб-орієнтований характер і передбачає використання облікових функцій, база даних повинна бути зручною для інтеграції з серверною та клієнтською частиною, підтримувати зв’язки між сутностями й мати достатній рівень безпеки.

У межах роботи доцільно розглянути кілька поширених варіантів: MySQL, MongoDB, PostgreSQL та хмарну платформу Supabase, яка використовує PostgreSQL як основну базу даних [20, 21]. MySQL є популярною реляційною СУБД, має високу швидкодію, велику спільноту та добре підходить для класичних веб-додатків. Однак для реалізації автентифікації, зберігання файлів, політик доступу та оновлень у реальному часі потрібно було б додатково підключати окремі сервіси або писати більше серверної логіки.

MongoDB є документоорієнтованою базою даних і добре підходить для зберігання гнучких JSON-подібних структур. Її зручно використовувати там, де схема даних часто змінюється або дані не мають складних зв’язків. Проте для цієї роботи основні сутності мають чіткі зв’язки: користувачі, профілі, заявки в

друзі, присутність. Тому реляційна модель є більш природною і зрозумілою для проєктування.

PostgreSQL є потужною реляційною СУБД з відкритим кодом [21, 22]. Вона підтримує зовнішні ключі, обмеження цілісності, транзакції, індекси, розширені типи даних і добре підходить для систем, де важливо зберігати зв’язану інформацію. Для розроблюваного веб-додатку PostgreSQL є доцільним вибором, оскільки дозволяє чітко описати структуру даних і забезпечити цілісність зв’язків між таблицями.

У цій роботі обрано Supabase, оскільки він надає не лише PostgreSQL як базу даних, а й додаткові сервіси, корисні для веб-додатку: автентифікацію користувачів, зберігання файлів, політики безпеки доступу до даних та зручний API для взаємодії з клієнтською частиною [20]. Це дозволяє не розробляти з нуля окрему систему реєстрації та входу, а використати готовий і поширений інструмент.

Важливою перевагою Supabase є підтримка Row Level Security (RLS), тобто правил доступу до окремих рядків таблиць [20, 21]. Завдяки цьому можна налаштувати, щоб користувач мав доступ лише до тих даних, які йому дозволено переглядати або змінювати. Наприклад, користувач може редагувати власний профіль, але не профілі інших людей; бачити інформацію про друзів, але не всі приватні дані системи.

Також Supabase зручний для навчального та демонстраційного проєкту, оскільки не потребує складного ручного розгортання бази даних на окремому сервері. Розробник отримує готову панель керування, SQL-редактор, механізм автентифікації та API. Це скорочує час розробки й дозволяє більше уваги приділити основній логіці системи: організації кімнат, аудіо- та відеозв’язку, чату й взаємодії користувачів.

Для реалізації інформаційного забезпечення було обрано Supabase на базі PostgreSQL. Такий вибір є обґрунтованим, оскільки поєднує надійність реляційної бази даних, зручність хмарної платформи, підтримку автентифікації

та можливість гнучкого налаштування доступу до даних. Це відповідає потребам розроблюваної системи та спрощує її подальше тестування й розгортання.

2.3 Розробка структури інформаційної бази

Структура інформаційної бази розроблюваної системи формується на основі логічної моделі даних, описаної в попередніх підрозділах. Її основне призначення — забезпечити збереження даних, необхідних для роботи користувачів із веб-додатком: облікової інформації, профілів, заявок у друзі, присутності користувачів та участі в кімнатах відеозв’язку. Оскільки система орієнтована на веб-середовище й використовує хмарну платформу Supabase, структура інформаційної бази проєктується у вигляді реляційних таблиць PostgreSQL [21, 22, 27].

Основою інформаційної бази є дані про авторизованих користувачів. У Supabase базова інформація про облікові записи зберігається у вбудованій підсистемі автентифікації. До таких даних належать унікальний ідентифікатор користувача, електронна пошта, службова інформація про реєстрацію та вхід. Оскільки ці дані керуються платформою, у власній структурі бази доцільно зберігати лише ті відомості, які потрібні для роботи інтерфейсу та взаємодії між користувачами.

Для цього створюється таблиця profiles, яка містить профіль користувача. Вона пов’язується з авторизованим користувачем за унікальним ідентифікатором. У профілі можуть зберігатися відображуване ім’я, посилання на аватар, дата створення або оновлення запису. Такий підхід дозволяє відокремити технічні облікові дані від публічної інформації, яку бачать інші користувачі системи.

Для реалізації соціальної взаємодії між користувачами передбачається таблиця friend_requests. Вона зберігає заявки в друзі: хто надіслав заявку, кому вона адресована, її поточний статус і дату створення. Статус може мати значення на кшталт pending або accepted, що дозволяє відрізняти нові заявки від уже

підтверджених контактів. Така таблиця є важливою для формування списку друзів і подальшого відображення присутності знайомих користувачів.

Окремою частиною інформаційної бази є таблиця user_presence, яка відповідає за інформацію про поточну активність користувача. У ній може зберігатися ідентифікатор користувача, код кімнати, у якій він перебуває, та час останнього оновлення. Ці дані дозволяють показувати, хто з друзів перебуває онлайн або в якій кімнаті знаходиться користувач. Оскільки інформація про присутність змінюється частіше, ніж профільні дані, її доцільно винести в окрему таблицю.

Для організації відеозустрічей у логічній моделі можуть використовуватися сутності rooms та room_users. Таблиця rooms описує кімнати зв’язку: код кімнати, час створення, статус або інші службові параметри. Таблиця room_users відображає участь користувачів у кімнатах і пов’язує конкретного користувача з конкретною кімнатою. Це дозволяє фіксувати, хто приєднався до певної зустрічі. Якщо в реалізації кімнати створюються тимчасово й не зберігаються постійно в базі даних, ці таблиці можуть розглядатися як логічна частина моделі або напрям подальшого розширення системи.

Зв’язки між таблицями будуються за принципами реляційної моделі. Таблиця profiles має зв’язок один до одного з авторизованим користувачем. Таблиця friend_requests містить два посилання на користувачів: відправника та отримувача заявки. Таблиця user_presence також пов’язана з користувачем і може мати зв’язок із поточною кімнатою. Таблиці rooms і room_users дозволяють описати зв’язок багато до багатьох між користувачами та кімнатами.

Під час розробки структури інформаційної бази важливо враховувати цілісність даних [23, 27]. Для цього використовуються первинні ключі, які однозначно ідентифікують записи, та зовнішні ключі, які забезпечують коректність зв’язків між таблицями. Наприклад, заявка в друзі не повинна посилатися на неіснуючого користувача, а профіль має належати конкретному обліковому запису. Також доцільно використовувати обмеження унікальності,

щоб не створювати дублікати профілів або повторні заявки між одними й тими самими користувачами.

Окрему увагу потрібно приділити безпеці доступу до даних. Оскільки використовується Supabase, для таблиць можна налаштовувати політики Row Level Security, які визначають, які записи користувач має право читати або змінювати. Наприклад, користувач може редагувати лише власний профіль, надсилати заявки від свого імені та переглядати інформацію про присутність тільки тих користувачів, з якими має підтверджений зв’язок.

Отже, розроблена структура інформаційної бази забезпечує збереження основних даних, необхідних для роботи системи: профілів користувачів, соціальних зв’язків, присутності та участі в кімнатах. Така структура є достатньою для реалізації базових функцій веб-додатку й водночас може бути розширена в майбутньому, наприклад для збереження історії повідомлень, налаштувань користувача або журналу активності.

2.4 Схема та фізична структура бази даних

Фізична структура бази даних є практичним продовженням логічної моделі, описаної в попередніх підрозділах. Якщо логічна модель показує основні сутності та зв’язки між ними, то фізична структура визначає, які саме таблиці створюються в базі даних, які поля вони містять, які типи даних використовуються та яким чином забезпечується цілісність інформації.

У межах розроблюваної системи база даних реалізується на основі PostgreSQL у середовищі Supabase [20, 21]. Такий підхід дозволяє поєднати реляційну структуру даних із готовими засобами автентифікації, керування доступом і зручним API для взаємодії з веб-додатком. Основними таблицями фізичної структури є auth_users, profiles, rooms та friend_requests.

Таблиця auth_users відповідає за збереження даних авторизованих користувачів. У Supabase ці дані фактично керуються вбудованою підсистемою автентифікації, однак у загальній схемі бази даних вона розглядається як базова

таблиця користувачів. Вона містить унікальний ідентифікатор користувача, електронну пошту, службові дані реєстрації та входу. Саме з нею пов’язуються інші таблиці, що містять додаткову інформацію про користувача.

Таблиця profiles використовується для збереження відкритої інформації про користувача. До неї можуть входити такі поля, як ідентифікатор профілю, посилання на користувача з auth_users, ім’я, яке відображається в інтерфейсі, посилання на аватар і дата оновлення профілю. Винесення профілю в окрему таблицю дозволяє не змішувати технічні дані входу з даними, які потрібні для відображення користувача в кімнаті або списку друзів.

Таблиця rooms призначена для опису кімнат аудіо- та відеозв’язку. Вона може містити унікальний код кімнати, дату створення, статус кімнати або інші службові параметри. Код кімнати використовується для підключення учасників до однієї зустрічі. Якщо кімнати в системі мають тимчасовий характер, їхні записи можуть зберігатися лише на час активної сесії або використовуватися як частина логічної моделі для подальшого розвитку системи.

Таблиця friend_requests зберігає заявки на дружбу між користувачами. Вона містить ідентифікатор заявки, посилання на користувача-відправника, посилання на користувача-отримувача, статус заявки та дату створення. Така структура дозволяє відстежувати, хто кому надіслав запит, чи був він прийнятий, і на основі цього формувати список друзів або доступних контактів.

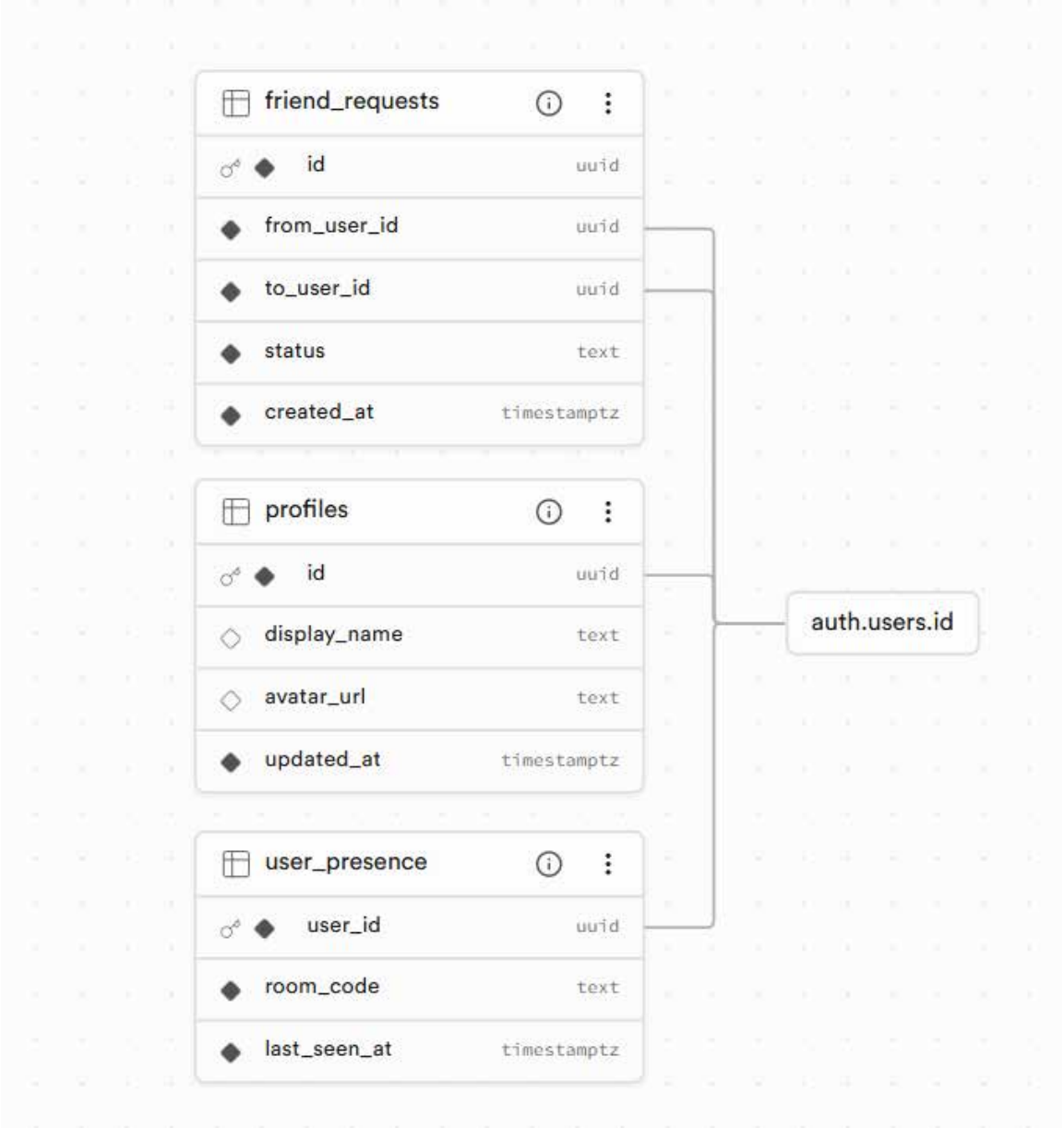

Рис. 6 - Схема фізичної структури бази даних системи

Зв’язки між таблицями реалізуються за допомогою первинних і зовнішніх ключів. Первинні ключі однозначно ідентифікують кожний запис у таблиці, а зовнішні ключі забезпечують правильний зв’язок між сутностями. Наприклад, запис у таблиці profiles має відповідати конкретному користувачу з auth_users, а записи в таблиці friend_requests мають посилатися на двох існуючих користувачів: того, хто надсилає заявку, і того, хто її отримує.

Для забезпечення коректності даних доцільно використовувати обмеження цілісності. Наприклад, один користувач не повинен мати кілька однакових профілів, а між двома користувачами не має створюватися кілька однакових

активних заявок у друзі. Також важливо передбачити перевірку статусів заявок, щоб у таблиці зберігалися лише допустимі значення, наприклад pending або accepted.

Фізична структура бази даних також повинна враховувати питання безпеки доступу. Оскільки система використовує Supabase, для таблиць можна застосовувати політики Row Level Security. Це дозволяє обмежити доступ користувачів до записів: наприклад, редагувати тільки власний профіль, створювати заявки від свого імені та переглядати лише ті дані, до яких користувач має право доступу.

Висновки до розділу 2

Отже, у другому розділі було розроблено інформаційне забезпечення системи організації аудіо- та відеозв’язку. У процесі роботи сформовано логічну модель даних у вигляді ER-діаграми, яка відображає основні сутності системи та зв’язки між ними. Було визначено ключові інформаційні об’єкти: користувачів, профілі, кімнати для зв’язку, участь користувачів у кімнатах і заявки на дружбу. Побудована модель дозволяє забезпечити цілісність даних і підтримку основних функцій веб-додатку.

У межах розділу також виконано аналіз систем управління базами даних і обґрунтовано вибір платформи Supabase на основі PostgreSQL. Обране рішення поєднує можливості реляційної бази даних із готовими сервісами автентифікації, API та механізмами безпеки доступу до даних. Це дозволяє спростити реалізацію серверної частини та зосередитися на функціональних можливостях системи.

Було розроблено структуру інформаційної бази, визначено основні таблиці, їхні поля, зв’язки, первинні та зовнішні ключі. Окрему увагу приділено питанням цілісності інформації та безпеки доступу за допомогою механізмів Row Level Security. Також сформовано фізичну структуру бази даних, яка є основою для подальшої реалізації програмного забезпечення та взаємодії веб-додатку з базою даних.

Результати, отримані в цьому розділі, створюють основу для реалізованої серверної логіки, механізмів взаємодії користувачів і функціонування системи аудіо- та відеозв’язку загалом. Запропонована структура бази даних є достатньою для реалізованої базових функцій системи та може бути розширена в майбутньому відповідно до потреб розвитку проєкту.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ ЗУСТРІЧЕЙ

3.1 Моделювання структури та взаємодії об'єктів

Після визначення вимог і проєктування інформаційної бази наступним етапом є моделювання структури програмного забезпечення. На цьому етапі необхідно показати, з яких основних частин складається система, які об’єкти беруть участь у роботі додатку та як вони взаємодіють між собою під час організації онлайн-зустрічі. Для цього доцільно використати кілька типів діаграм: діаграму пакетів, діаграму класів та діаграму розгортання. Разом вони дозволяють описати систему з різних боків: логічну структуру коду, основні сутності та зв’язки між ними, а також фізичне розміщення компонентів у середовищі виконання.

Розроблюване програмне забезпечення має веб-орієнтовану архітектуру. Основними частинами системи є клієнтський рівень, сервер застосунку та хмарні сервіси. Клієнтський рівень працює у браузері користувача й відповідає за інтерфейс, отримання доступу до камери та мікрофона, відображення локального й віддаленого відеопотоку, керування кнопками дзвінка, чатом і профілем. Сервер застосунку виконує роль проміжного координатора: він обслуговує HTTP-запити, віддає статичні сторінки, забезпечує WebSocket-з’єднання та передає службові повідомлення між учасниками кімнати. Хмарні сервіси відповідають за збереження даних користувачів, автентифікацію, профілі, файли аватарів і, за потреби, обмін оновленнями в режимі реального часу.

Першою доцільно розглянути діаграму пакетів, оскільки вона показує загальну організацію системи та розподіл її частин за відповідальністю. На клієнтському рівні виділяються пакет інтерфейсу користувача та пакет використання браузерних API. Інтерфейс включає HTML-сторінки, CSS-стилі та JavaScript-код, який реагує на дії користувача. Браузерні API забезпечують

роботу WebRTC, WebSocket, локального сховища та інших механізмів, необхідних для роботи відеозустрічі [1, 9, 10, 11]. На серверному рівні розміщується застосунок Node.js, який використовує залежності Express, Socket.IO, dotenv, uuid та інші бібліотеки [16, 17, 18, 19]. Окремо виділено хмарні сервіси: Supabase як платформа для PostgreSQL, Auth, Storage, Realtime та PostgREST, а також хмарний тунель для демонстраційного HTTPS-доступу.

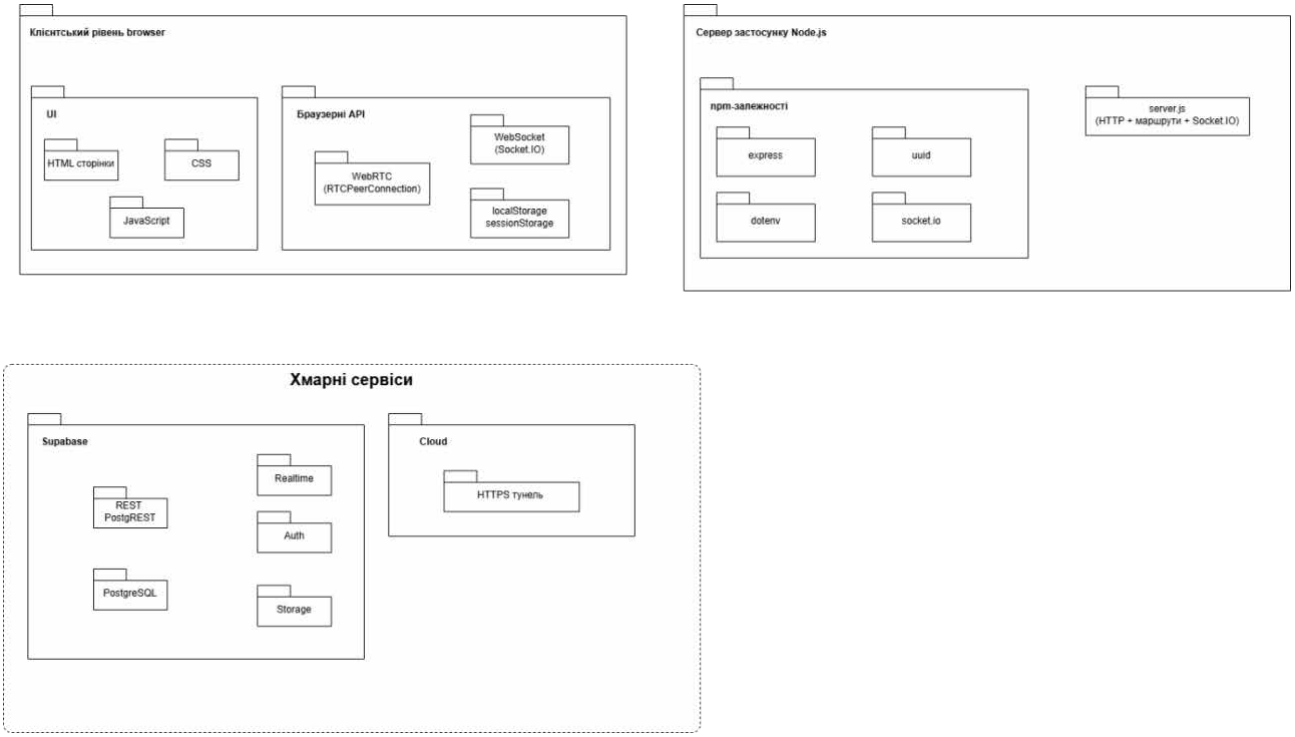

Рис. 7 - Діаграма пакетів програмного забезпечення

Діаграма пакетів демонструє, що система не є монолітною в повному сенсі, а складається з кількох взаємопов’язаних шарів. Клієнтська частина зосереджена на взаємодії з користувачем і медіа-пристроями. Серверна частина відповідає за маршрутизацію, обмін службовими повідомленнями та доставку чату. Хмарна частина забезпечує збереження облікових даних і профільної інформації. Такий поділ є зручним для супроводу, оскільки зміни в інтерфейсі, серверній логіці або базі даних можна виконувати відносно незалежно.

Наступним елементом моделювання є діаграма класів. Вона дозволяє описати основні об’єкти предметної області та взаємозв’язки між ними. У межах системи ключовими сутностями є Користувач, Оператор, Сеанс зв’язку, Повідомлення,

Камера, Мікрофон і Медіапотік. Користувач бере участь у сеансі зв’язку, надсилає повідомлення, використовує камеру та мікрофон. Сеанс зв’язку об’єднує учасників і визначає контекст обміну аудіо-, відео- та текстовими даними. Оператор або керуючий компонент відповідає за модерацію, приєднання до сеансу, керування взаємодією та підтримку порядку в межах зустрічі. Медіапотік є результатом роботи камери й мікрофона та використовується для передачі звуку і відео між учасниками.

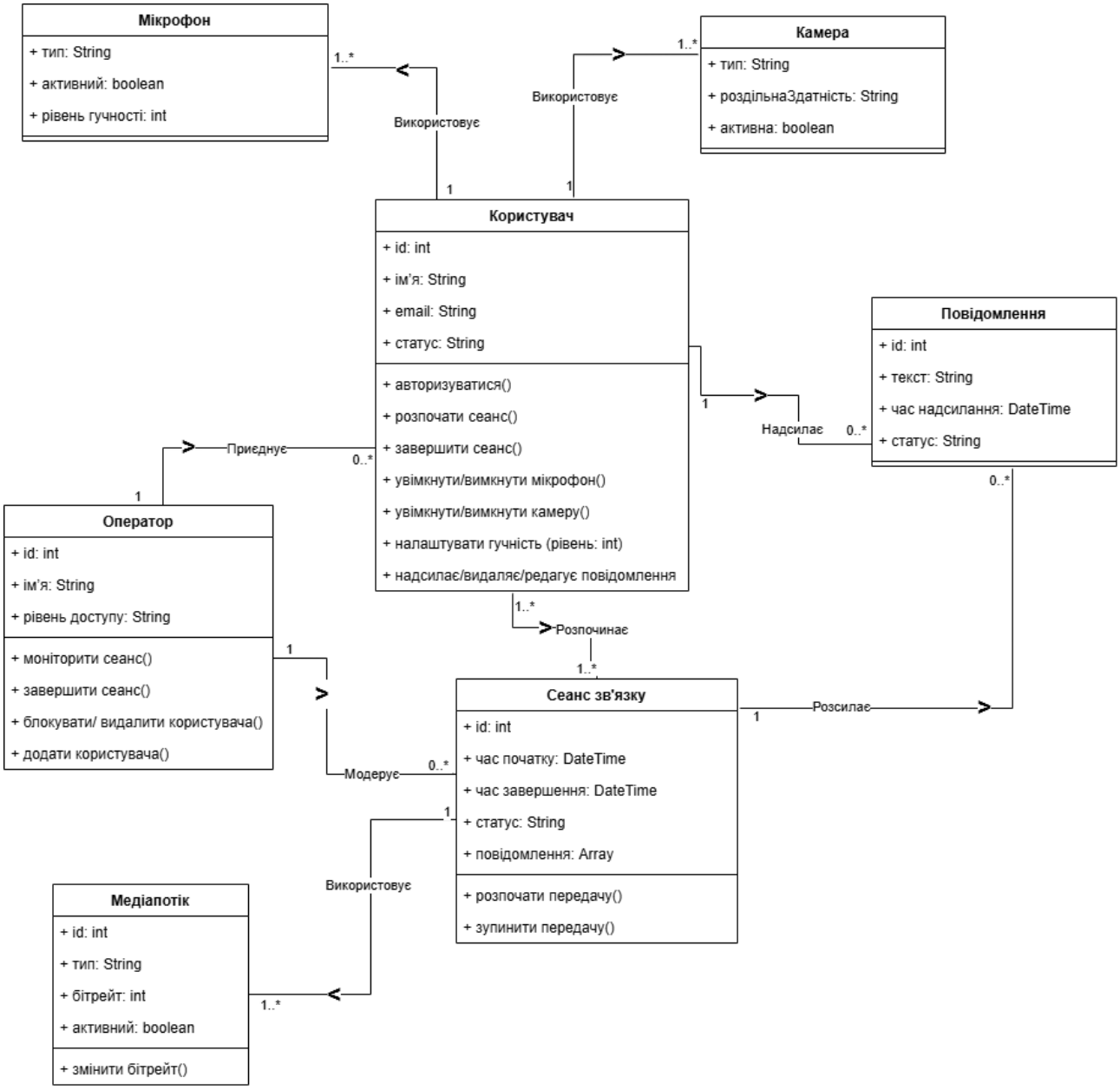

Рис. 8 - Діаграма класів системи організації аудіо- та відеозв’язку

Діаграма класів допомагає краще зрозуміти, які об’єкти повинні бути представлені у програмній логіці. Наприклад, користувач може мати один або кілька пристроїв введення, надсилати багато повідомлень і брати участь у сеансах зв’язку. Повідомлення належать конкретному користувачу та передаються в межах відповідного сеансу. Камера й мікрофон розглядаються як джерела даних, які формують медіапотік. Така модель дозволяє логічно розділити відповідальність: користувач відповідає за дії в системі, пристрої — за отримання медіа, сеанс — за контекст зустрічі, а повідомлення — за текстову взаємодію.

Окремо важливо описати взаємодію об’єктів під час роботи системи. Типовий сценарій починається з того, що користувач відкриває веб-сторінку та переходить до кімнати. Далі браузер запитує дозвіл на доступ до камери й мікрофона. Після надання дозволів створюється локальний медіапотік, який відображається в інтерфейсі та використовується для подальшого встановлення з’єднання з іншими учасниками. Сервер отримує інформацію про приєднання користувача до кімнати та передає службові повідомлення іншим учасникам. У процесі взаємодії клієнти обмінюються параметрами з’єднання, після чого між ними встановлюється передача аудіо- та відеоданих. Текстові повідомлення при цьому надсилаються через сервер, щоб усі учасники кімнати отримували їх у межах спільного чату.

Третім важливим елементом є діаграма розгортання, яка показує, де фізично або логічно розміщуються компоненти системи під час виконання. На ній доцільно відобразити клієнтські пристрої користувачів, сервер застосунку на Node.js, публічні STUN-сервери, хмарну платформу Supabase і, за потреби, проксі або тунель для HTTPS-доступу. Клієнтські пристрої взаємодіють із сервером через HTTP та WebSocket [9]. Для встановлення медіа-з’єднання вони використовують механізми WebRTC та ICE, а для визначення мережевих параметрів можуть звертатися до STUN-серверів [1, 6, 8]. Сервер застосунку не обробляє медіапотік безпосередньо, а забезпечує координацію, маршрутизацію службових повідомлень і доставку текстового чату.

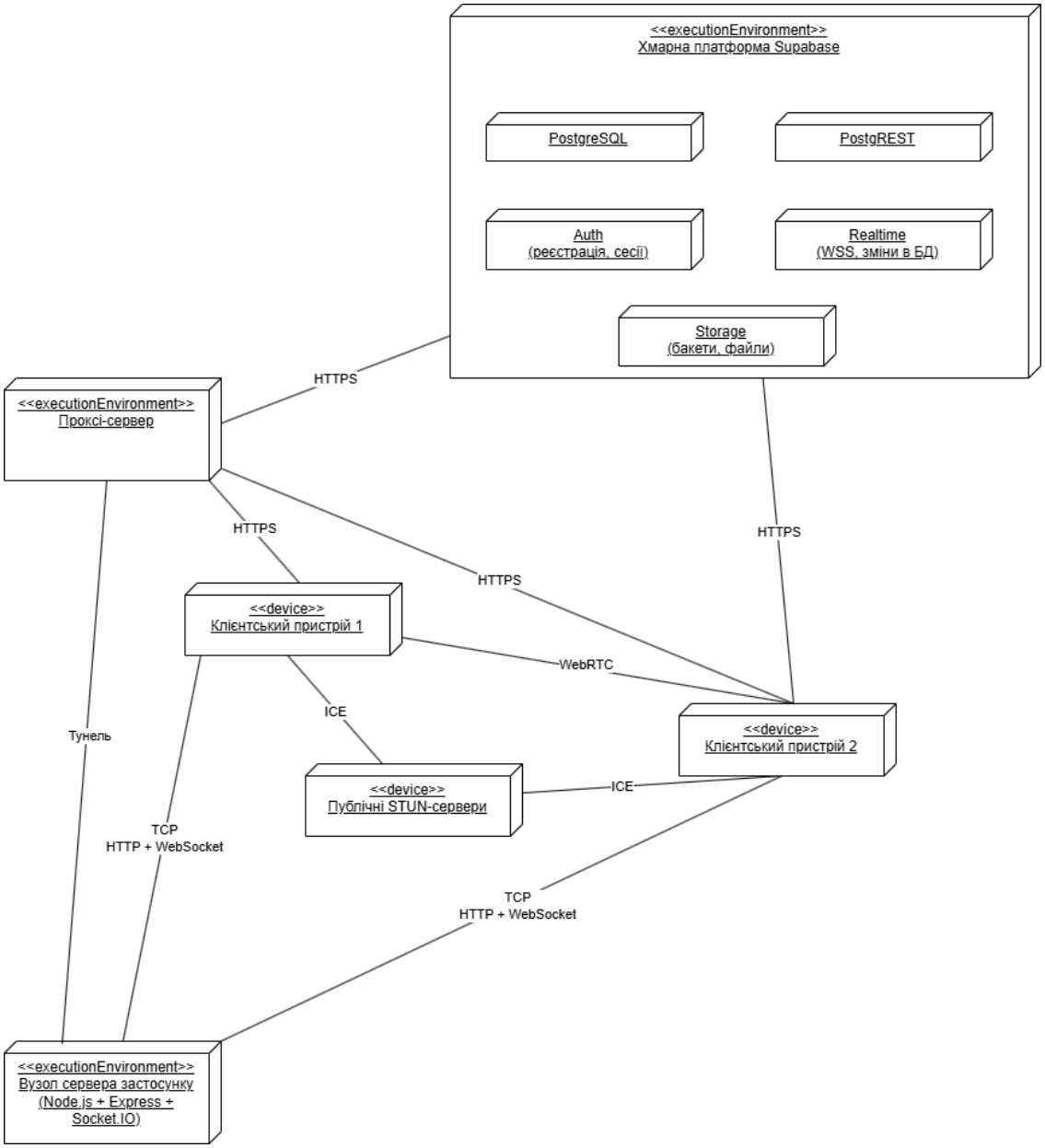

Рис. 9 - Діаграма розгортання програмної системи організації онлайн-зустрічей

Діаграма розгортання підкреслює важливу особливість обраної архітектури: аудіо- та відеопотоки між користувачами можуть передаватися напряму між клієнтськими пристроями через WebRTC, тоді як сервер виконує допоміжну роль. Такий підхід зменшує навантаження на сервер у невеликих групових зустрічах, оскільки він не транслює відео кожного учасника всім іншим. Водночас це накладає певні обмеження: якість зв’язку залежить від мережі учасників, а для складних мережевих умов може знадобитися додаткове використання TURN-сервера. У межах навчального проєкту така архітектура є

обґрунтованою, оскільки дозволяє реалізувати робочий прототип без надмірного ускладнення інфраструктури.

Загалом моделювання структури та взаємодії об’єктів показує, що система побудована за зрозумілим принципом поділу відповідальності. Браузер користувача відповідає за інтерфейс і роботу з медіапристроями, сервер Node.js — за координацію кімнат, сигналізацію та чат, хмарна платформа — за збереження облікових даних і профільної інформації. Таке розділення спрощує розробку, тестування та подальше розширення системи. Наприклад, у майбутньому можна додати запис зустрічей, покращене керування ролями, TURN-сервер або розширені налаштування профілю, не змінюючи повністю всю архітектуру додатку.

3.2 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення визначає, як саме розподілені основні частини системи, за що відповідає кожна з них і яким чином вони взаємодіють між собою. Для веб-додатку організації онлайн-зустрічей така структура є особливо важливою, оскільки система одночасно працює з інтерфейсом користувача, аудіо- та відеопотоками, серверними подіями, текстовими повідомленнями й обліковими даними користувачів.

Розроблюване програмне забезпечення побудоване за клієнт-серверним принципом. Клієнтська частина виконується у браузері користувача та відповідає за відображення сторінок, взаємодію з кнопками й формами, доступ до камери та мікрофона, показ локального й віддаленого відео, а також надсилання текстових повідомлень. Серверна частина працює на базі Node.js і забезпечує обробку HTTP-запитів, видачу статичних файлів, організацію кімнат, обмін службовими повідомленнями між учасниками та доставку повідомлень чату. Окремим рівнем виступає хмарна платформа, яка відповідає за автентифікацію користувачів, збереження профілів, аватарів і пов’язаних даних.

Клієнтська частина програмного забезпечення складається з HTML-сторінок, CSS-стилів і JavaScript-модулів [12, 13, 14]. HTML визначає структуру інтерфейсу: головну сторінку, сторінку кімнати, елементи керування викликом, чат і кабінет користувача. CSS відповідає за зовнішній вигляд, адаптивність і зручність використання. JavaScript реалізує основну логіку на стороні браузера: обробку дій користувача, роботу з WebRTC, підключення до Socket.IO, оновлення інтерфейсу, керування мікрофоном і камерою.

Серверна частина виконує роль координатора. Вона не є основним передавачем відеопотоку, але допомагає користувачам встановити з'єднання між собою. Для цього сервер приймає підключення учасників до кімнати, зберігає інформацію про активні сокети, пересилає службові повідомлення WebRTC та повідомляє інших учасників про приєднання або відключення користувача. Крім того, сервер обробляє текстовий чат: отримує повідомлення від одного учасника й передає його всім користувачам у межах відповідної кімнати.

Хмарна частина системи представлена сервісом Supabase. Вона використовується для роботи з авторизованими користувачами та даними, які мають зберігатися між сесіями. До таких даних належать профілі, аватари, заявки в друзі та інформація про присутність. Завдяки цьому сервер додатку не перевантажується зайвою логікою зберігання облікових даних, а система отримує готові засоби автентифікації та взаємодії з базою даних.

З погляду організації файлів програмне забезпечення можна умовно поділити на кілька груп. Перша група — це серверні файли, які запускають веб-сервер, підключають залежності, налаштовують маршрути й події Socket.IO. Друга група — публічні клієнтські файли, які передаються браузеру: сторінки, стилі, скрипти для кімнати, головної сторінки та кабінету. Третя група — конфігураційні файли, де визначаються залежності проєкту, змінні оточення, параметри підключення до зовнішніх сервісів і допоміжні налаштування запуску.

Основний сценарій взаємодії між частинами системи виглядає так: користувач відкриває веб-додаток у браузері, сервер передає йому потрібні

HTML, CSS і JavaScript-файли, після чого клієнтський код ініціалізує інтерфейс. Якщо користувач входить у кімнату, браузер встановлює WebSocket-з’єднання із сервером і повідомляє про приєднання. Сервер фіксує учасника в кімнаті та пересилає іншим користувачам події, необхідні для встановлення аудіо- та відеозв’язку. Після цього браузери учасників обмінюються службовими даними та формують медіа-з’єднання. Паралельно користувачі можуть надсилати текстові повідомлення, які проходять через сервер і відображаються в чаті.

Особливістю організаційної структури є те, що відео- та аудіопотоки не потребують постійної обробки сервером застосунку. Сервер переважно відповідає за сигналізацію, тобто обмін службовими даними для встановлення зв’язку. Такий підхід є зручним для невеликих групових зустрічей, оскільки зменшує навантаження на сервер і робить архітектуру простішою. Водночас він має обмеження: для дуже великих конференцій або складних мережевих умов можуть знадобитися додаткові серверні компоненти, наприклад TURN або медіа-сервер.

Організаційна структура програмного забезпечення складається з трьох основних рівнів: клієнтського, серверного та хмарного. Клієнтський рівень забезпечує інтерфейс і роботу з медіа, серверний — координацію кімнат, сигналізацію та чат, а хмарний — автентифікацію й збереження даних користувачів. Такий поділ робить систему зрозумілою, достатньо гнучкою та придатною для подальшого розвитку.

3.3 Компонентна структура системи

Компонентна структура системи показує, з яких великих функціональних частин складається програмне забезпечення та як ці частини взаємодіють між собою. На відміну від опису окремих класів або файлів, компонентна модель розглядає систему на вищому рівні: як набір незалежних або частково незалежних блоків, кожен з яких виконує свою роль. Для веб-додатку організації онлайн-зустрічей це особливо важливо, оскільки система одночасно включає

інтерфейс користувача, серверну логіку, WebRTC-зв’язок, обмін повідомленнями та хмарні сервіси для зберігання даних і автентифікації.

У розроблюваній системі можна виділити такі основні компоненти: веб-інтерфейс, WebRTC, сервер Node.js з Express і Socket.IO, Supabase Database та API, а також Supabase Auth. Кожен із цих компонентів відповідає за окремий напрям роботи системи, але разом вони формують єдиний процес організації аудіо- та відеозустрічі в браузері. Загальна компонентна структура наведена на діаграмі.

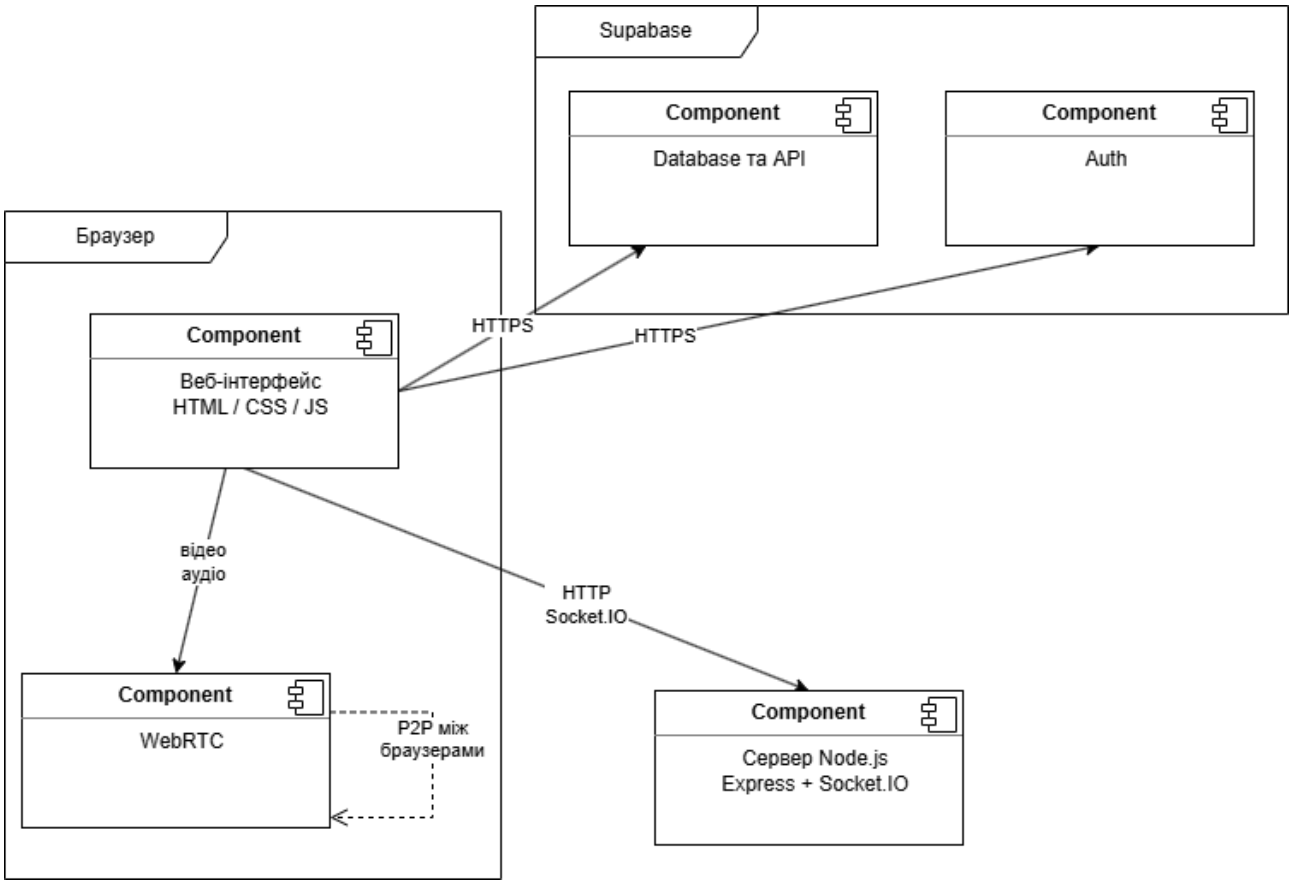

Рис. 10 - Діаграма компонентів програмного забезпечення

Компонент «Веб-інтерфейс HTML / CSS / JS» працює на стороні браузера та є основною частиною, з якою безпосередньо взаємодіє користувач. Він відповідає за відображення сторінок, кнопок керування дзвінком, відеоелементів, чату, форм входу та профілю. HTML задає структуру сторінок, CSS відповідає за зовнішній вигляд і адаптивність, а JavaScript реалізує поведінку інтерфейсу: реакцію на натискання кнопок, підключення до кімнати,

керування камерою та мікрофоном, відправлення повідомлень і оновлення інформації на екрані.

Окремим компонентом виділено WebRTC, який відповідає за передачу аудіо- та відеопотоків між браузерами користувачів [1, 2, 4]. Саме цей компонент забезпечує можливість спілкування в режимі реального часу без необхідності передавати весь медіапотік через сервер застосунку. Браузер отримує доступ до камери та мікрофона користувача, формує медіапотік і передає його іншому учаснику або учасникам зустрічі. Такий підхід є ефективним для невеликих групових дзвінків, оскільки зменшує навантаження на сервер.

Важливо зазначити, що WebRTC не працює повністю самостійно. Для того щоб два браузери встановили з’єднання, їм потрібно обмінятися службовими даними: інформацією про параметри сесії, мережеві адреси та можливі шляхи передачі даних. Цей процес називається сигналізацією. У розроблюваній системі сигналізація реалізується через серверний компонент Node.js Express + Socket.IO [16, 17, 18]. Тобто медіапотік передається між браузерами, а сервер допомагає організувати початкове встановлення зв’язку.

Компонент «Сервер Node.js Express + Socket.IO» виконує роль центрального координатора системи. За допомогою Express сервер обслуговує HTTP-запити, надає клієнту HTML, CSS і JavaScript-файли та забезпечує маршрутизацію сторінок. За допомогою Socket.IO реалізується двосторонній обмін подіями між браузером і сервером. Саме через цей канал передаються повідомлення про приєднання до кімнати, відключення користувача, службові WebRTC-дані та текстові повідомлення чату.

Також серверний компонент контролює логіку кімнат. Коли користувач переходить у кімнату, сервер отримує інформацію про код кімнати та додає учасника до відповідної групи з’єднань. Якщо в кімнаті вже є інші учасники, сервер повідомляє сторони про необхідність встановлення зв’язку. Якщо користувач відправляє повідомлення в чат, сервер приймає його та пересилає всім учасникам цієї кімнати. У разі виходу користувача сервер повідомляє інших учасників, щоб інтерфейс міг оновитися та прибрати відключений відеопотік.

Компонент Supabase Database та API відповідає за збереження даних, які мають існувати довше, ніж одна відеосесія. До таких даних належать профілі користувачів, аватари, заявки на дружбу, статуси присутності та інші облікові відомості. Через API клієнтська або серверна частина може отримувати й оновлювати ці дані. Завдяки цьому система не потребує окремо розробленого backend-модуля для кожної операції з базою даних, а використовує можливості хмарної платформи.

Компонент Supabase Auth забезпечує автентифікацію користувачів [20]. Він відповідає за реєстрацію, вхід, вихід і підтримку сесії користувача. Це важливо для функцій, які не можуть бути повністю анонімними: редагування профілю, завантаження аватара, надсилання заявок у друзі або відображення присутності. Завдяки використанню готового компонента автентифікації зменшується складність власної реалізації та підвищується надійність роботи з обліковими записами.

Між компонентами існують різні типи взаємодії. Браузер отримує веб-інтерфейс від сервера та виконує його на стороні користувача. Веб-інтерфейс взаємодіє з компонентом WebRTC для отримання й передачі аудіо- та відео. Через HTTP і Socket.IO браузер обмінюється даними із сервером Node.js. Через HTTPS веб-інтерфейс звертається до Supabase для роботи з автентифікацією та даними користувача. При цьому медіа-з’єднання через WebRTC може встановлюватися напряму між браузерами, що позначається як P2P-взаємодія.

Перевагою такої компонентної структури є чіткий розподіл відповідальності. Веб-інтерфейс відповідає за взаємодію з користувачем, WebRTC — за медіапотоки, сервер Node.js — за сигналізацію та чат, Supabase — за облікові функції та збереження даних. Завдяки цьому кожен компонент можна розробляти, тестувати й удосконалювати окремо. Наприклад, можна змінити дизайн інтерфейсу без зміни серверної логіки, покращити роботу чату без зміни бази даних або розширити профіль користувача без втручання в механізм WebRTC.

Крім того, така структура є зручною для подальшого розвитку системи. У майбутньому можна додати нові компоненти: модуль запису зустрічей, сервер ретрансляції медіа, систему ролей, сповіщення, календар зустрічей або розширену аналітику. При цьому базова архітектура залишиться зрозумілою: клієнт, сервер координації, медіа-взаємодія та хмарне збереження даних.

Компонентна структура розроблюваного програмного забезпечення відображає логічне розділення системи на функціональні блоки. Вона поєднує браузерний інтерфейс, технологію WebRTC, сервер Node.js з Express і Socket.IO та хмарні сервіси Supabase. Такий підхід забезпечує достатню гнучкість, простоту супроводу й можливість розширення системи відповідно до майбутніх потреб.

3.4 Вибір інструментарію для створення прикладного програмного забезпечення

Вибір інструментарію для створення прикладного програмного забезпечення є важливим етапом розробки, оскільки саме від обраних технологій залежить зручність реалізації, стабільність роботи системи, можливість подальшого розширення та простота супроводу. Для програмного забезпечення організації онлайн-зустрічей потрібно підібрати такі засоби, які дозволяють реалізувати веб-інтерфейс, аудіо- та відеозв’язок у режимі реального часу, серверну координацію учасників, текстовий чат, а також роботу з обліковими записами користувачів.

Під час вибору інструментарію враховувались такі критерії: підтримка сучасними браузерами, доступність документації, поширеність у веб-розробці, можливість швидкої інтеграції між клієнтською та серверною частинами, безкоштовність або наявність безкоштовного тарифу, а також відповідність темі бакалаврської роботи. Оскільки система має працювати у браузері, основою клієнтської частини обрано стандартні веб-технології: HTML, CSS та JavaScript [12, 13, 14, 15].

HTML використовується для створення структури сторінок веб-додатку. За його допомогою формуються основні елементи інтерфейсу: головна сторінка, кнопки входу до кімнати, блоки відео, поле чату, форми авторизації та профілю користувача. Перевагою HTML є його універсальність і підтримка всіма сучасними браузерами.

CSS використовується для оформлення зовнішнього вигляду веб-додатку. За допомогою стилів задаються кольори, розміри, розміщення блоків, адаптивність інтерфейсу та зручність роботи користувача. Це важливо для системи відеозв'язку, оскільки користувач має швидко знаходити основні елементи керування: увімкнення або вимкнення мікрофона, камери, чат, вихід із кімнати.

Основною мовою програмування клієнтської частини є JavaScript. Вона використовується для обробки дій користувача, роботи з браузерними API, встановлення з'єднання із сервером, керування медіапотоками та оновлення інтерфейсу без перезавантаження сторінки. JavaScript є природним вибором для такого типу системи, оскільки він виконується безпосередньо у браузері й має доступ до необхідних веб-можливостей.

Для реалізації аудіо- та відеозв'язку обрано технологію WebRTC [1, 2, 4]. Вона дозволяє браузерам передавати аудіо-, відео- та інші дані в режимі реального часу [5, 10]. Основною перевагою WebRTC є те, що користувачу не потрібно встановлювати окрему програму: достатньо сучасного браузера. WebRTC також дає змогу організувати P2P-з'єднання між клієнтами, тобто передавати медіапотік напряму між браузерами, зменшуючи навантаження на сервер застосунку.

Водночас WebRTC потребує допоміжного механізму для встановлення початкового з'єднання між учасниками. Для цього потрібна сигналізація: обмін службовими повідомленнями між браузерами. У роботі для цього використовується Socket.IO, який забезпечує зручний двосторонній обмін подіями між клієнтом і сервером [9, 18, 19]. Socket.IO добре підходить для чатів,

кімнат, повідомлень про підключення або відключення користувачів, а також для передачі службових даних WebRTC.

Серверна частина реалізується на основі Node.js [16]. Ця платформа дозволяє виконувати JavaScript на сервері, що спрощує розробку, оскільки одна мова використовується і на клієнті, і на сервері [14]. Node.js добре підходить для систем реального часу, де потрібно обробляти багато подій, підключень і повідомлень. Для розроблюваної системи це особливо важливо, оскільки сервер має працювати з кімнатами, учасниками, сигналізацією та чатом.

Для спрощення створення серверної частини використовується фреймворк Express.js [17]. Він дозволяє швидко налаштувати HTTP-сервер, маршрути, віддачу статичних файлів і обробку запитів. Express є одним із найпоширеніших інструментів у Node.js-екосистемі, має простий синтаксис і велику кількість прикладів. У межах роботи Express використовується для обслуговування веб-сторінок і підключення клієнтської частини до серверної логіки.

Для керування залежностями використовується npm — стандартний менеджер пакетів для Node.js. За його допомогою підключаються необхідні бібліотеки: Express, Socket.IO, dotenv, uuid та інші. Використання npm спрощує встановлення, оновлення та відтворення середовища розробки на іншому комп’ютері.

Бібліотека dotenv використовується для роботи зі змінними оточення. Це важливо з погляду безпеки, оскільки конфіденційні параметри, ключі доступу та службові налаштування не повинні зберігатися безпосередньо в коді. Змінні оточення дозволяють розділити програмну логіку та конфігурацію.

Бібліотека uuid може використовуватись для генерації унікальних ідентифікаторів. У веб-додатку для організації зустрічей це може бути корисно для створення службових ідентифікаторів сесій, користувачів або інших об’єктів, які мають бути унікальними в межах системи.

Для зберігання даних і реалізації облікових функцій обрано Supabase [20]. Це хмарна платформа, яка надає базу даних PostgreSQL, автентифікацію користувачів, API для роботи з даними, сховище файлів і механізми безпеки.

Використання Supabase дозволяє не створювати з нуля складну систему реєстрації, входу та збереження профілів, а скористатися готовими засобами.

Основою Supabase є PostgreSQL — потужна реляційна система управління базами даних [21, 22]. Вона добре підходить для зберігання структурованих даних: профілів користувачів, заявок у друзі, інформації про присутність і зв'язків між сутностями. PostgreSQL підтримує зовнішні ключі, обмеження цілісності, транзакції та інші механізми, необхідні для надійної роботи інформаційної бази.

Компонент Supabase Auth використовується для автентифікації користувачів. Він забезпечує реєстрацію, вхід, вихід і підтримку користувацької сесії. Це значно спрощує розробку облікових функцій і дозволяє зосередитися на логіці самого додатку: кімнатах, відеозв'язку, чаті та взаємодії користувачів.

Компонент Supabase Storage може використовуватись для зберігання файлів, зокрема аватарів користувачів. Це дає змогу не зберігати файли безпосередньо на сервері застосунку, а використовувати окреме хмарне сховище. Такий підхід є зручним для масштабування й підтримки.

Для забезпечення доступу до локально розгорнутого застосунку з інтернету може використовуватись ngrok. Він створює тимчасовий публічний HTTPS-тунель до локального сервера. Це корисно для демонстрації роботи додатку, тестування на різних пристроях або перевірки підключення користувачів без повноцінного розгортання на хостингу.

Під час розробки також використовуються стандартні засоби браузера: DevTools, консоль, інспектор елементів і мережевий монітор. Вони дають змогу перевіряти роботу інтерфейсу, помилки JavaScript, WebSocket-з'єднання, HTTP-запити та поведінку сторінки під час виконання.

Отже, для створення прикладного програмного забезпечення було обрано такий інструментарій: HTML, CSS і JavaScript для клієнтської частини; WebRTC для аудіо- та відеозв'язку; Node.js, Express і Socket.IO для серверної частини, сигналізації та чату; Supabase і PostgreSQL для збереження даних, профілів та

автентифікації; npm, dotenv, uuid і ngrok як допоміжні засоби розробки, конфігурації та демонстрації.

Обраний набір технологій є доцільним для бакалаврської роботи, оскільки він дозволяє створити повноцінний веб-додаток реального часу без надмірного ускладнення архітектури. Система залишається зрозумілою, гнучкою та придатною для подальшого розвитку: у майбутньому до неї можна додати TURN-сервер, запис зустрічей, розширені ролі користувачів, календар подій або покращену систему сповіщень.

3.5 Алгоритмізація та програмування програмних модулів

Алгоритмізація програмних модулів є етапом, на якому загальна архітектура системи перетворюється на послідовність конкретних дій, що виконуються клієнтською та серверною частинами. Для програмного забезпечення організації онлайн-зустрічей основними модулями є: модуль головної сторінки, модуль кімнати відеозв’язку, модуль WebRTC-з’єднання, модуль сигналізації, модуль текстового чату, модуль автентифікації та модуль профілю користувача. Кожен із них виконує окрему функцію, але разом вони забезпечують повний сценарій роботи системи — від відкриття сайту до проведення аудіо- та відеозустрічі.

Серверна частина системи реалізується на основі Node.js, Express та Socket.IO. Її алгоритм можна описати як послідовність запуску веб-сервера, підключення необхідних залежностей, налаштування маршрутизації, видачі статичних файлів і обробки подій у реальному часі. Після запуску сервер очікує HTTP-запити від браузера та WebSocket-підключення від клієнтів, які входять до кімнат. Коли користувач відкриває головну сторінку або сторінку кімнати, сервер передає відповідні HTML, CSS і JavaScript-файли. Якщо користувач приєднується до кімнати, сервер додає його сокет до відповідної групи та повідомляє інших учасників про нове підключення.

Алгоритм роботи серверного модуля можна подати так: спочатку виконується ініціалізація Express-застосунку, потім підключається HTTP-сервер і Socket.IO. Далі налаштовується роздача статичних файлів із клієнтської частини. Після цього сервер переходить у режим очікування підключень. При отриманні події приєднання до кімнати сервер перевіряє код кімнати, додає користувача до кімнати, перевіряє кількість учасників і розсилає іншим клієнтам службові повідомлення. Якщо користувач надсилає WebRTC-пропозицію, відповідь або ICE-кандидат, сервер не обробляє зміст цих даних, а пересилає їх потрібному учаснику. Якщо надходить текстове повідомлення, сервер передає його всім користувачам у межах кімнати. При відключенні користувача сервер повідомляє інших учасників і звільняє пов’язані з ним дані.

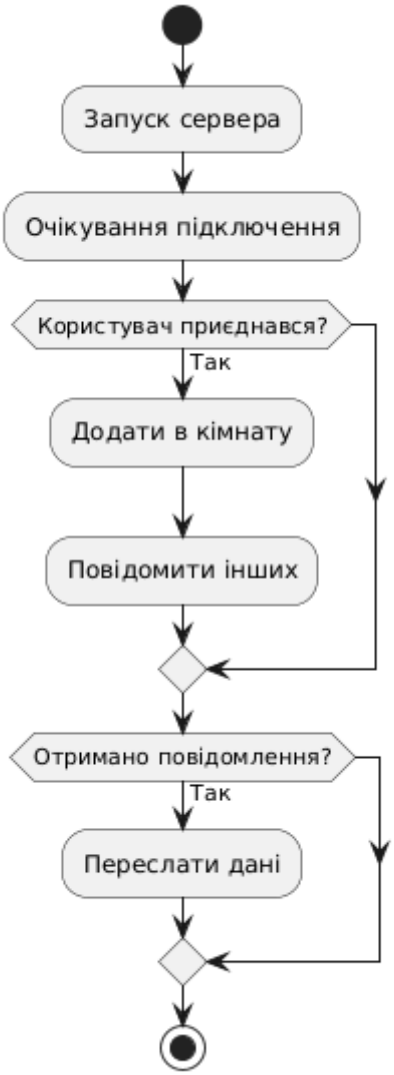

Рис. 11 - Блок-схема роботи серверного модуля

Клієнтський модуль кімнати є одним із найважливіших у системі. Його завдання — підготувати інтерфейс зустрічі, отримати доступ до камери та мікрофона, встановити зв’язок із сервером, створити WebRTC-з’єднання з іншими учасниками та відображати медіапотоки на сторінці. Алгоритм роботи цього модуля починається з завантаження сторінки кімнати. Після цього JavaScript-код визначає код кімнати з адреси сторінки, ініціалізує підключення до Socket.IO, запитує дозвіл у браузера на використання камери й мікрофона та створює локальний медіапотік. Отриманий потік відображається у відеоелементі користувача й додається до майбутніх WebRTC-з’єднань.

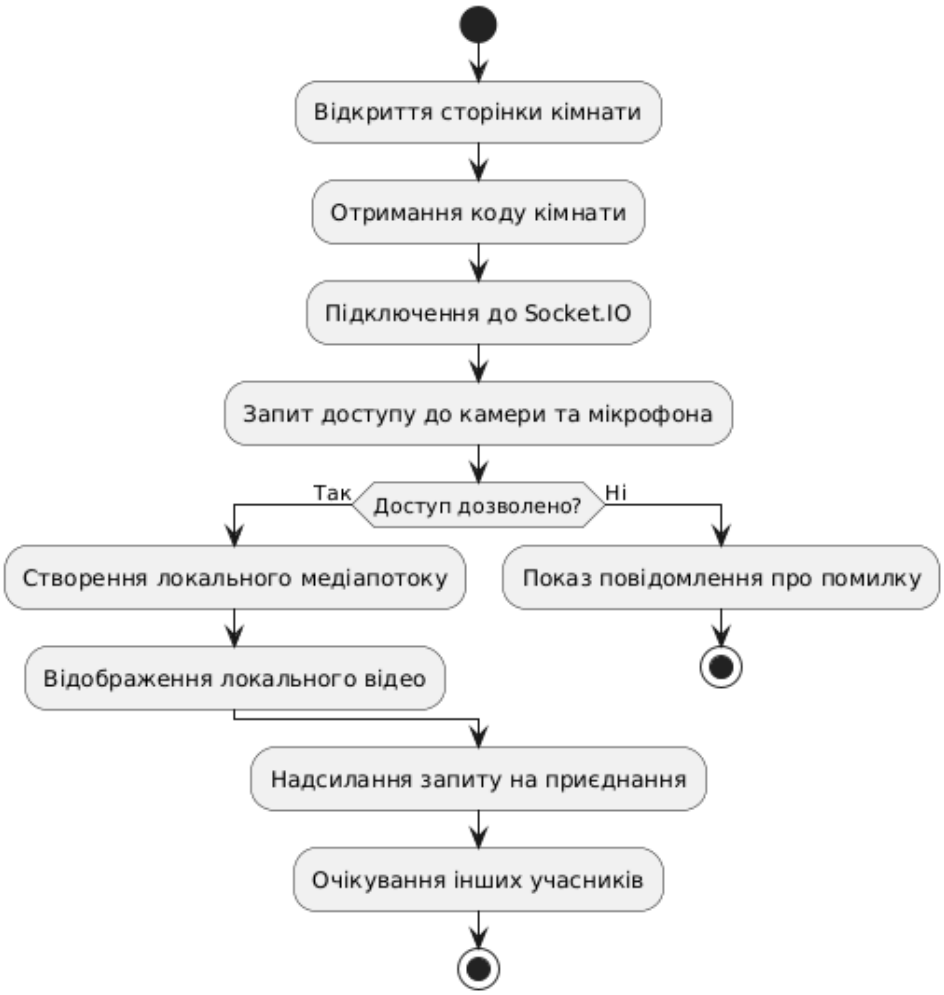

Рис. 12 - Блок-схема підключення до кімнати

Для встановлення зв’язку між учасниками використовується алгоритм WebRTC-сигналізації [1, 5, 8]. Один клієнт створює об’єкт RTCPeerConnection, додає до нього локальні аудіо- та відеотреки, формує пропозицію з’єднання (offer) і передає її іншому учаснику через сервер [10]. Інший клієнт приймає цю пропозицію, створює власний RTCPeerConnection, додає локальні треки, формує відповідь (answer) і надсилає її назад. Паралельно обидва клієнти обмінюються ICE-кандидатами, які допомагають браузерам знайти можливий мережевий шлях для передачі даних. Після завершення цього процесу між браузерами встановлюється медіа-з’єднання, і користувачі можуть чути та бачити один одного.

Рис. 13 - Блок-схема встановлення WebRTC-з’єднання

Важливою частиною клієнтської логіки є керування мікрофоном і камерою. Для цього використовуються методи роботи з медіатреків. Якщо користувач натискає кнопку вимкнення мікрофона, відповідний аудіотрек вимикається, але саме з’єднання не розривається. Аналогічно працює вимкнення відео: відеотрек стає неактивним, а інтерфейс оновлює стан кнопки. Такий підхід дозволяє швидко керувати приватністю під час зустрічі без повторного встановлення всього з’єднання.

Модуль текстового чату працює паралельно з відеозв’язком. Його алгоритм простіший: користувач вводить повідомлення в поле чату й натискає кнопку надсилання або клавішу Enter. Клієнтська частина перевіряє, що повідомлення не є порожнім, після чого надсилає його на сервер через Socket.IO. Сервер приймає повідомлення, додає до нього службову інформацію (наприклад, ім’я користувача або час) і розсилає всім учасникам кімнати. На клієнтській стороні повідомлення додається до списку чату без перезавантаження сторінки.

Рис. 14 - Блок-схема текстового чату

Окремим модулем є головна сторінка веб-додатку. Вона забезпечує перший контакт користувача із системою. На цьому етапі користувач може створити нову кімнату або ввести відомий код існуючої кімнати. Алгоритм роботи модуля передбачає перевірку введеного значення, формування правильного посилання та перехід на сторінку кімнати. Якщо користувач створює нову кімнату, система генерує короткий числовий код або інший ідентифікатор, після чого відкриває відповідну адресу.

Модуль автентифікації реалізує вхід і реєстрацію користувачів через Supabase Auth. Його алгоритм складається з відправлення даних форми до сервісу автентифікації, отримання результату, збереження сесії та оновлення інтерфейсу. Якщо користувач успішно входить у систему, він отримує доступ до

особистого кабінету та додаткових функцій. Якщо дані введено неправильно, система має показати повідомлення про помилку й дозволити повторити спробу.

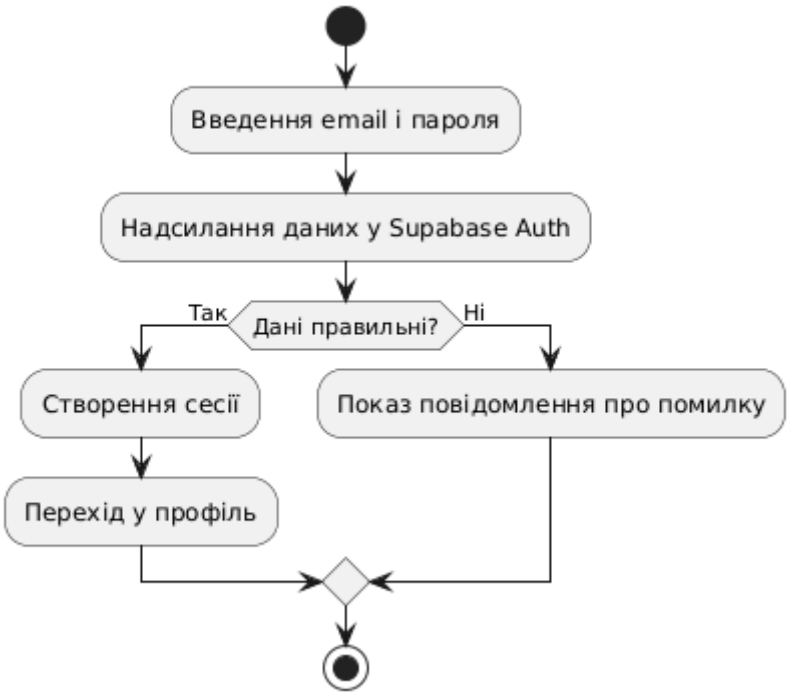

Рис. 15 - Блок-схема автентифікації

Модуль профілю користувача відповідає за відображення й редагування персональних даних. Після входу в систему клієнтська частина звертається до бази даних Supabase, отримує профіль користувача та показує його в інтерфейсі. Користувач може змінити ім’я, аватар або інші доступні поля. Після збереження змін дані оновлюються в базі. Якщо використовується завантаження аватара, файл передається до Supabase Storage, а в таблиці профілю зберігається посилання на нього.

Модуль заявок у друзі забезпечує пошук користувачів, надсилання запитів і прийняття або відхилення заявок. Алгоритм його роботи включає отримання списку користувачів або результатів пошуку, перевірку наявних заявок, створення нового запису в таблиці friend_requests і оновлення статусу заявки. Після прийняття заявки користувачі можуть відображатися один одному як друзі. Це корисно для подальшого показу присутності та швидкого переходу до спільної кімнати.

Модуль присутності відповідає за оновлення інформації про активність користувача. Коли авторизований користувач входить у кімнату, система може записати поточний код кімнати та час оновлення. Коли користувач залишає кімнату або закриває сторінку, інформація має бути очищена або змінена. Це дозволяє друзям бачити, чи активний користувач і де він перебуває. Такий модуль є допоміжним, але підвищує зручність користування системою.

Під час програмування модулів важливо дотримуватися принципу розділення відповідальності. Сервер не повинен виконувати зайву обробку медіапотоків, якщо система побудована на P2P WebRTC. Його завдання — координувати кімнати, передавати службові повідомлення та підтримувати чат. Клієнтська частина повинна відповідати за медіа, інтерфейс і взаємодію з користувачем. Supabase має відповідати за збереження довготривалих даних і автентифікацію.

У процесі реалізації також потрібно враховувати помилки та граничні ситуації. Наприклад, користувач може заборонити доступ до камери або мікрофона, ввести неправильний код кімнати, втратити інтернет-з’єднання, спробувати приєднатися до переповненої кімнати або надіслати порожнє повідомлення в чат. Для таких ситуацій програмні модулі повинні передбачати повідомлення користувачу або коректне завершення операції.

Алгоритмізація та програмування модулів забезпечують практичну реалізацію архітектури системи. Основними алгоритмами є створення кімнати, приєднання до кімнати, отримання медіапотоку, встановлення WebRTC-з’єднання, обмін службовими повідомленнями, надсилання текстових повідомлень, робота з профілем і оновлення присутності. Сукупність цих модулів формує працездатне програмне забезпечення для організації аудіо- та відеозустрічей у браузері.

Висновки до розділу 3

Отже, у третьому розділі було виконано проєктування та практичну реалізацію програмного забезпечення для організації аудіо- та відеозустрічей у браузері. У процесі роботи сформовано структуру системи, визначено основні програмні компоненти, їхню взаємодію та принципи організації клієнтської, серверної й хмарної частин додатку.

У межах розділу було побудовано діаграми пакетів, класів, компонентів і розгортання, які дозволили описати систему з різних рівнів: логічної структури, взаємодії об’єктів і фізичного розміщення компонентів. Це дало змогу чітко визначити ролі браузера користувача, серверного застосунку Node.js та хмарної платформи Supabase у загальній архітектурі програмного забезпечення.

Також було розглянуто організаційну та компонентну структуру системи. Визначено, що клієнтська частина відповідає за інтерфейс, роботу з камерою та мікрофоном, відображення медіапотоків і взаємодію з користувачем. Серверна частина реалізує сигналізацію WebRTC, керування кімнатами та текстовий чат, а Supabase забезпечує автентифікацію, збереження профілів, файлів і даних користувачів. Такий поділ відповідальності зробив систему зрозумілою, гнучкою та зручною для подальшого супроводу.

У результаті вибору інструментарію було обґрунтовано використання HTML, CSS і JavaScript для клієнтської частини, WebRTC для передачі аудіо- та відеоданих, Node.js, Express і Socket.IO для серверної логіки та сигналізації, а також Supabase і PostgreSQL для роботи з даними й автентифікацією. Обраний набір технологій дозволив реалізувати повноцінний веб-додаток реального часу без надмірного ускладнення архітектури.

Під час алгоритмізації та програмування модулів було реалізовано основні функції системи: створення та підключення до кімнат, встановлення WebRTC-з’єднання, передачу аудіо- та відеопотоків, текстовий чат, автентифікацію користувачів, роботу з профілем, заявки в друзі та механізм присутності

користувачів. Також було враховано обробку типових помилок і граничних ситуацій, що підвищує стабільність та зручність роботи системи.

У результаті виконання розділу розроблено працездатне програмне забезпечення для організації онлайн-зустрічей, яке забезпечує аудіо- та відеозв’язок у режимі реального часу, підтримує взаємодію користувачів і може бути використане як основа для подальшого розвитку системи. Архітектура додатку дозволяє в майбутньому додати нові можливості, зокрема запис зустрічей, систему ролей, TURN-сервер, календар подій або розширені засоби керування користувачами.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування системи є важливим етапом розробки програмного забезпечення, оскільки дозволяє перевірити правильність роботи основних функцій, виявити помилки та оцінити готовність додатку до демонстрації й подальшої експлуатації [31]. Для програмного забезпечення організації аудіо- та відеозустрічей тестування має охоплювати не лише звичайну перевірку інтерфейсу, а й роботу з камерою, мікрофоном, сервером сигналізації, текстовим чатом, кімнатами та обліковими функціями користувачів.

Метою тестування є підтвердження того, що система виконує основні вимоги, сформульовані в попередніх розділах: дозволяє створити або відкрити кімнату за кодом, підключити кількох учасників, встановити аудіо- та відеозв’язок, обмінюватися текстовими повідомленнями, керувати мікрофоном і камерою, а також використовувати профіль і авторизацію для зареєстрованих користувачів.

Тестування проводиться у кілька етапів. Спочатку перевіряється запуск серверної частини та доступність головної сторінки веб-додатку. Після цього тестуються основні сценарії роботи з кімнатами: створення нової кімнати, перехід за кодом, підключення другого користувача та відображення учасників. Далі перевіряється робота медіафункцій: доступ до камери та мікрофона, поява локального відео, встановлення з’єднання з іншим учасником і передача аудіо- та відеопотоку.

Окремо тестується сигналізація WebRTC, оскільки саме вона забезпечує обмін службовими повідомленнями між учасниками. Під час перевірки необхідно впевнитися, що сервер коректно передає повідомлення про приєднання до кімнати, WebRTC-пропозиції, відповіді та ICE-кандидати. Якщо

хоча б один із цих елементів працює неправильно, з’єднання між браузерами може не встановитися.

Важливим напрямом є тестування текстового чату. Перевіряється введення повідомлення, надсилання через сервер, отримання повідомлення іншим учасником і відображення його в інтерфейсі. Також доцільно перевірити обмеження на порожні повідомлення або надмірно довгі тексти, якщо такі обмеження передбачені реалізацією.

Функції керування медіа перевіряються окремо. Потрібно переконатися, що кнопки вимкнення мікрофона та вимкнення відео змінюють стан відповідних медіатреків без розриву з’єднання. Також перевіряється, чи змінюється візуальний стан кнопок або індикаторів у інтерфейсі, щоб користувач розумів, у якому стані перебуває камера чи мікрофон.

Для авторизованих користувачів тестуються облікові можливості: реєстрація, вхід, вихід із системи, відображення профілю, зміна імені або аватара, пошук інших користувачів, надсилання заявки в друзі та прийняття заявки. Якщо система відображає присутність користувача, потрібно перевірити оновлення поточної кімнати та коректне очищення або зміну статусу після виходу.

Також необхідно перевірити поведінку системи в граничних і помилкових ситуаціях. До таких ситуацій належать: заборона доступу до камери або мікрофона, закриття вкладки одним з учасників, відключення інтернету, спроба приєднатися до переповненої кімнати, введення некоректного коду кімнати або спроба надіслати порожнє повідомлення. Система повинна або коректно обробляти такі ситуації, або принаймні не завершувати роботу аварійно [30, 31].

Під час тестування доцільно використовувати кілька браузерів або кілька пристроїв, щоб змоделювати реальну взаємодію різних користувачів. Наприклад, один користувач може бути відкритий у Google Chrome, інший — у Microsoft Edge або Firefox. Також можна перевірити роботу системи в локальній мережі та через HTTPS-тунель, якщо використовується демонстраційний доступ ззовні.

Результати тестування показують, чи відповідає система основним вимогам і чи може вона використовуватись для демонстрації роботи онлайн-зустрічі. Якщо виявлено обмеження, їх необхідно зазначити як напрями подальшого вдосконалення. Наприклад, у майбутньому можна покращити автоматичне повторне підключення після втрати мережі, додати TURN-сервер для складних мережевих умов, розширити систему ролей або додати запис зустрічей.

Тестування системи підтверджує працездатність основних модулів: кімнат, аудіо- та відеозв'язку, сигналізації, чату, керування медіа та облікових функцій. Проведення таких перевірок дозволяє оцінити якість реалізації, виявити слабкі місця та підготувати систему до впровадження й подальшої експлуатації.

4.2 Вимоги до апаратного та програмного забезпечення

Оскільки розроблювана система є веб-застосунком, основні вимоги поділяються на дві групи: вимоги до серверного середовища, де запускається серверна частина, та вимоги до клієнтського пристрою, з якого користувач підключається до онлайн-зустрічі. У межах цієї роботи сервер не виконує повну обробку аудіо- та відеопотоків, а переважно відповідає за видачу веб-сторінок, сигналізацію WebRTC, обмін повідомленнями чату та взаємодію з хмарними сервісами. Тому вимоги до апаратного сервера є помірними порівняно з системами, де весь відеопотік проходить через медіа-сервер.

Для запуску серверної частини необхідний комп'ютер або віртуальний сервер із встановленим середовищем Node.js. Мінімальними апаратними характеристиками для демонстраційного або навчального запуску можна вважати: процесор із 2 ядрами, 2 ГБ оперативної пам'яті, 1–2 ГБ вільного місця на диску для файлів проєкту та залежностей, а також стабільне підключення до мережі Інтернет. Для більш комфортної роботи бажано використовувати процесор із 4 ядрами, 4 ГБ оперативної пам'яті та швидке мережеве з'єднання. Якщо кількість одночасних користувачів збільшується, вимоги до сервера також

зростають, насамперед через кількість WebSocket-підключень і обмін службовими повідомленнями.

Оскільки аудіо- та відеопотоки передаються за допомогою WebRTC переважно між браузерами користувачів, сервер не потребує значних ресурсів для кодування або декодування відео. Це є перевагою обраної архітектури. Водночас сервер повинен мати стабільний канал зв'язку, оскільки через нього проходять службові повідомлення, повідомлення чату та події приєднання або відключення учасників. Для демонстраційного режиму достатньо локального запуску на персональному комп'ютері, але для постійної експлуатації доцільно використовувати VPS або хмарний сервер із публічною адресою та підтримкою HTTPS.

До програмного забезпечення серверного середовища належать: операційна система Windows, Linux або macOS; середовище виконання Node.js; менеджер пакетів npm; встановлені залежності проєкту, зокрема Express, Socket.IO, dotenv та інші бібліотеки, зазначені у файлі залежностей [16, 17, 18]. Також необхідно налаштувати змінні оточення для підключення до зовнішніх сервісів, зокрема Supabase та, за потреби, ngrok.

Для роботи з базою даних окремий фізичний сервер у межах цієї системи не є обов'язковим, оскільки використовується хмарна платформа Supabase. Вона надає PostgreSQL, автентифікацію користувачів, сховище файлів і API. Тому вимоги до власного сервера зменшуються: він не зберігає всю базу даних локально, а взаємодіє з хмарним сервісом через захищені запити. Водночас потрібне стабільне інтернет-з'єднання між сервером застосунку, клієнтами та Supabase.

Для клієнтської частини потрібен сучасний пристрій із браузером, який підтримує WebRTC, доступ до камери й мікрофона та стабільне підключення до Інтернету. Це може бути персональний комп'ютер, ноутбук або інший пристрій із підтримкою сучасних браузерів. Рекомендовано використовувати Google Chrome, Microsoft Edge, Mozilla Firefox або інший актуальний браузер. Для участі у відеозустрічі необхідні камера, мікрофон і пристрій відтворення звуку —

динаміки або навушники. Якщо користувач планує лише слухати або писати в чат, вимоги до пристроїв можуть бути нижчими.

Якість роботи клієнтської частини значною мірою залежить від швидкості й стабільності мережі. Для одного відеодзвінка бажано мати стабільне широкосмугове підключення. При слабкому інтернеті можливі затримки, зниження якості відео або переривання зв’язку. Також потрібно враховувати, що браузер має надати дозвіл на використання камери та мікрофона; без цього повноцінна участь у відеозустрічі буде неможливою.

Для демонстрації системи з доступом ззовні може використовуватися ngrok або інший інструмент тунелювання. У такому випадку локальний сервер отримує тимчасову публічну HTTPS-адресу, за якою до нього можуть підключитися інші користувачі. Це зручно для навчальної роботи, захисту проєкту або тестування на різних пристроях без повноцінного розгортання на хостингу. Для промислової експлуатації краще використовувати постійний домен, SSL-сертифікат і стабільний хостинг.

4.3 Склад інсталяційного пакету

Інсталяційний пакет програмного забезпечення призначений для передавання, розгортання та запуску веб-додатку на локальному комп’ютері або сервері. Оскільки розроблювана система є веб-застосунком на основі Node.js, інсталяційний пакет має містити вихідний код серверної та клієнтської частин, файли конфігурації, опис залежностей, інструкції щодо запуску, а також додаткові матеріали, необхідні для налаштування бази даних і зовнішніх сервісів.

До складу інсталяційного пакету входить серверна частина програмного забезпечення. Вона містить головний файл запуску сервера, у якому налаштовується HTTP-сервер, маршрути Express, підключення Socket.IO та обробка подій у реальному часі. Саме серверна частина забезпечує видачу веб-

сторінок, роботу кімнат, пересилання службових повідомлень WebRTC і повідомлень чату.

Окремою частиною пакету є клієнтська частина, яка складається з HTML-, CSS- та JavaScript-файлів. HTML-файли відповідають за структуру сторінок: головної сторінки, кімнати відеозв'язку та кабінету користувача. CSS-файли визначають зовнішній вигляд і адаптивність інтерфейсу. JavaScript-файли реалізують логіку роботи у браузері: підключення до кімнати, роботу з камерою та мікрофоном, WebRTC-з'єднання, текстовий чат, авторизацію та взаємодію з Supabase.

Важливим елементом інсталяційного пакету є файл package.json. У ньому зазначаються назва проєкту, команда запуску, список залежностей і службова інформація про застосунок. Саме цей файл використовується менеджером пакетів npm для встановлення потрібних бібліотек. Після отримання пакету користувач або адміністратор має виконати встановлення залежностей командою npm install.

Разом із package.json може постачатися файл package-lock.json, який фіксує конкретні версії встановлених залежностей. Це дозволяє відтворити однакове середовище запуску на різних комп'ютерах і зменшує ризик помилок через зміну версій бібліотек.

До пакету також входить файл прикладу змінних оточення, наприклад .env.example. У ньому можуть бути перелічені параметри, які потрібно заповнити перед запуском: порт сервера, адреса Supabase, ключі доступу, токен ngrok або інші конфігураційні значення. Реальний файл .env не слід публікувати або передавати разом із відкритим кодом, якщо він містить секретні ключі.

Для роботи з інформаційною базою доцільно додати SQL-скрипти або окрему інструкцію зі створення таблиць у Supabase. Такі матеріали можуть містити створення таблиць profiles, friend_requests, user_presence або інших потрібних структур, а також налаштування політик доступу Row Level Security. Це спрощує повторне розгортання системи та дозволяє швидко підготувати базу даних до роботи.

Також до інсталяційного пакету варто включити інструкцію користувача або адміністратора. У ній потрібно описати порядок встановлення Node.js, встановлення залежностей, налаштування змінних оточення, запуск сервера, відкриття застосунку в браузері та перевірку основних функцій. Якщо система може запускатися через ngrok для демонстрації ззовні, це також бажано описати окремим пунктом.

До додаткових матеріалів можуть входити діаграми, зображення, документація, тестові сценарії та результати тестування. Вони не є обов'язковими для запуску системи, але корисні для пояснення архітектури, захисту роботи та подальшого супроводу програмного забезпечення.

Після розпакування інсталяційного пакету користувач має встановити залежності, налаштувати змінні оточення, підготувати базу даних у Supabase і запустити серверну частину. Після цього веб-додаток стає доступним у браузері за локальною адресою або за публічним посиланням, якщо використовується тунелювання чи хостинг.

Інсталяційний пакет має містити всі необхідні файли для запуску й перевірки програмного забезпечення: серверний код, клієнтський інтерфейс, опис залежностей, конфігурацію, матеріали для бази даних та інструкцію з розгортання. Такий склад забезпечує можливість відтворити роботу системи на іншому комп'ютері або сервері без додаткового пошуку потрібних компонентів.

Висновки до розділу 4

Отже, у четвертому розділі було розглянуто питання тестування, вимог до середовища виконання та підготовки інсталяційного пакету розробленого програмного забезпечення для організації аудіо- та відеозустрічей. Проведені перевірки дозволили оцінити працездатність системи, правильність реалізації основних функцій і готовність веб-додатку до демонстрації та подальшого використання.

У процесі тестування було перевірено роботу ключових модулів системи: створення та підключення до кімнат, встановлення WebRTC-з’єднання, передачу аудіо- та відеопотоків, текстовий чат, керування камерою й мікрофоном, а також функції авторизації, профілів користувачів і заявок у друзі. Окрему увагу приділено перевірці сигналізації WebRTC, роботі Socket.IO та обробці типових помилкових ситуацій, зокрема втрати з’єднання, неправильного коду кімнати або відсутності доступу до медіапристроїв. Результати тестування підтвердили працездатність основних функцій системи та її відповідність поставленим вимогам.

Також було визначено вимоги до апаратного та програмного забезпечення. Встановлено, що для роботи серверної частини достатньо середовища Node.js із помірними апаратними ресурсами, оскільки обрана архітектура WebRTC дозволяє передавати медіапотоки напряму між браузерами користувачів без значного навантаження на сервер. Для клієнтської частини достатньо сучасного браузера з підтримкою WebRTC, камери, мікрофона та стабільного підключення до Інтернету. Використання Supabase дозволило зменшити вимоги до локального серверного середовища завдяки перенесенню функцій збереження даних і автентифікації до хмарної платформи.

У межах розділу також було сформовано склад інсталяційного пакету програмного забезпечення. До нього включено серверну та клієнтську частини веб-додатку, файли конфігурації, опис залежностей, SQL-матеріали для підготовки бази даних і документацію щодо запуску системи. Такий підхід забезпечує можливість швидкого розгортання й відтворення роботи програмного забезпечення на іншому комп’ютері або сервері.

У результаті виконання розділу підтверджено, що розроблена система є працездатною, може бути розгорнута у веб-середовищі та забезпечує виконання основних функцій організації онлайн-зустрічей. Запропоноване програмне забезпечення має достатній рівень готовності для демонстрації, тестової експлуатації та подальшого розвитку відповідно до майбутніх потреб користувачів і можливого розширення функціональності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WebRTC 1.0: Real-Time Communication Between Browsers : W3C Recommendation. URL: https://www.w3.org/TR/webrtc/

2. Loreto S., Romano S. P. Real-Time Communication with WebRTC: Peer-to-Peer in the Browser. Sebastopol : O'Reilly Media, 2014. 164 p.

3. Johnston A. B., Burnett D. C. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web. 3rd ed. St. Louis : Digital Codex LLC, 2014. 380 p.

4. Sergiienko A. WebRTC Cookbook. Birmingham : Packt Publishing, 2015. 230 p.

5. Alvestrand H. Overview: Real-Time Protocols for Browser-Based Applications : RFC 8825. IETF, 2021. URL: https://www.rfc-editor.org/rfc/rfc8825

6. Petit-Huguenin M., Salgueiro G., Rosenberg J. et al. Session Traversal Utilities for NAT (STUN) : RFC 8489. IETF, 2020. URL: https://www.rfc-editor.org/rfc/rfc8489

7. Reddy T., Johnston A., Matthews P., Rosenberg J. Traversal Using Relays around NAT (TURN) : RFC 8656. IETF, 2020. URL: https://www.rfc-editor.org/rfc/rfc8656

8. Keranen A., Holmberg C., Rosenberg J. Interactive Connectivity Establishment (ICE) : RFC 8445. IETF, 2018. URL: https://www.rfc-editor.org/rfc/rfc8445

9. Fette I., Melnikov A. The WebSocket Protocol : RFC 6455. IETF, 2011. URL: https://www.rfc-editor.org/rfc/rfc6455

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

11. Media Capture and Streams API : MDN Web Docs. Mozilla Foundation. URL:https://developer.mozilla.org/enUS/docs/Web/API/Media_Capture_and_Streams_API

12. HTML Living Standard : WHATWG. URL: https://html.spec.whatwg.org/

13. CSS Snapshot 2023 : W3C Working Group Note. URL: https://www.w3.org/TR/CSS/

14. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 706 p.

15. Crockford D. JavaScript: The Good Parts. Sebastopol : O'Reilly Media, 2008. 172 p.

16. Node.js Documentation : OpenJS Foundation. URL: https://nodejs.org/en/docs/

17. Hahn E. Express in Action: Writing, building, and testing Node.js applications. Shelter Island : Manning Publications, 2016. 256 p.

18. Rai R. Socket.IO Real-time Web Application Development. Birmingham : Packt Publishing, 2013. 140 p.

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

20. Supabase Documentation. URL: https://supabase.com/docs

21. PostgreSQL 16 Documentation : The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/

22. Obe R. O., Hsu L. S. PostgreSQL: Up and Running. 3rd ed. Sebastopol : O'Reilly Media, 2017. 314 p.

23. Дейт К. Дж. Введення в системи баз даних. 8-ме вид. Київ : Діалектика, 2019. 1328 с.

24. Буч Г., Рамбо Дж., Якобсон А. Мова UML. Керівництво користувача. 2-ге вид. Київ : ДМК Прес, 2017. 496 с.

25. Фаулер М. UML. Основи. 3-тє вид. Київ : Символ-Плюс, 2018. 192 с.

26. Глибовець М. М., Стрюк А. М. Веб-програмування : навчальний посібник. Київ : НаУКМА, 2018. 240 с.

27. Пасічник В. В., Резніченко В. А. Організація баз даних та знань : підручник. Київ : Видавнича група BHV, 2019. 384 с.

28. Дакетт Дж. HTML і CSS. Розробка та дизайн веб-сайтів. Київ : Ескімо, 2020. 480 с.

29. OWASP Top 10:2021 — The Ten Most Critical Web Application Security Risks : OWASP Foundation. URL: https://owasp.org/Top10/

30. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva : ISO, 2011. 25 p.

31. Майєрс Г., Сендлер К., Беджет Т. Мистецтво тестування програм. 3-тє вид. Київ : Діалектика-Вільямс, 2019. 272 с.

32. Jitsi Meet — Open Source Video Conferencing : Documentation. 8x8, Inc. URL: https://jitsi.github.io/handbook/

33. BigBlueButton Documentation : BigBlueButton Inc. URL: https://docs.bigbluebutton.org/

34. Zoom Help Center : Zoom Video Communications, Inc. URL: https://support.zoom.us/

35. Microsoft Teams Documentation : Microsoft Corporation. URL: https://learn.microsoft.com/en-us/microsoftteams/

ДОДАТОК А

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ДИПЛОМ

ІІІ ступеня

НАГОРОДЖУЄТЬСЯ

Максим САУХ

студент факультету інформаційних технологій

ПЕРЕМОЖЕЦЬ І ТУРУ
Всеукраїнського конкурсу студентських
наукових робіт з галузей знань і спеціальностей
у 2025/2026 навчальному році
зі спеціальності «Інженерія програмного забезпечення»

Ректор,
доктор економічних наук,
професор **Вадим ТКАЧУК**

м. Київ, 2026 р.

ДОДАТОК Б

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Саух Максим Андрійович , здобувач(ка) НУБіП України,
усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему
«Програмне забезпечення для організації онлайн аудіо- та відеозв'язку» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
   - o [ ] інструменти генеративного ШІ не використовувалися.
   - o [✓] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
     - ▪ [ ] перекладу іншомовних джерел;
     - ▪ [ ] стилістичного редагування та перевірки граматики;
     - ▪ [✓] допомоги у написанні/відлагодженні програмного коду;
     - ▪ [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___»_____ 2026 р. _____ **Максим САУХ**

(підпис)